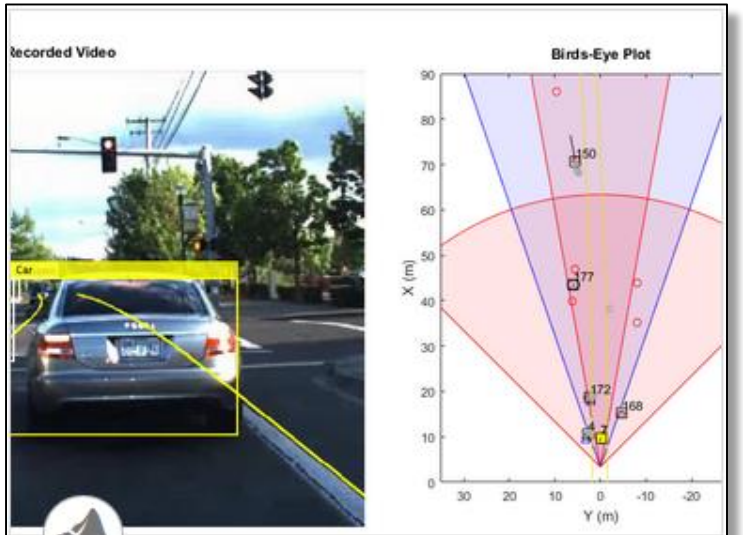
Automated Driving System Toolbox R2017a

Case study:
Vision and radar-based sensor fusion

Seo-Wook Park
Principal Application Engineer
The MathWorks, Inc.

Automated Driving System Toolbox R2017a

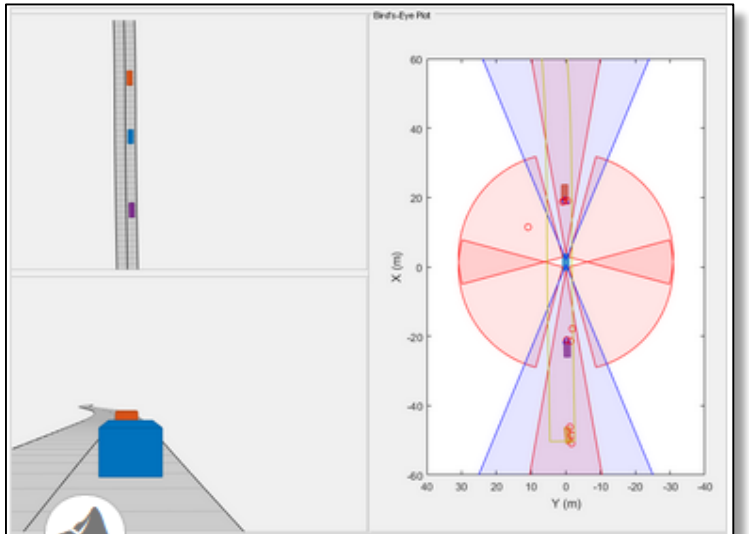
Examples for vision and radar-based sensor fusion



Forward Collision Warning Using Sensor Fusion

Perform forward collision warning by fusing data from vision and radar sensors to track objects in front of the vehicle.

Open Script



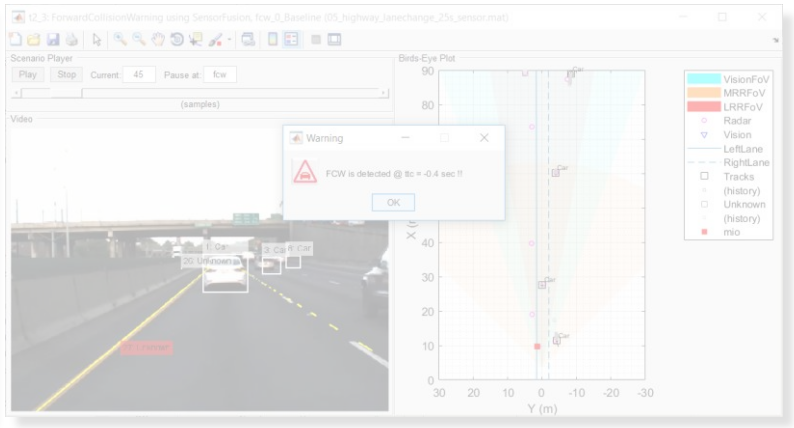
Sensor Fusion Using Synthetic Radar and Vision Data

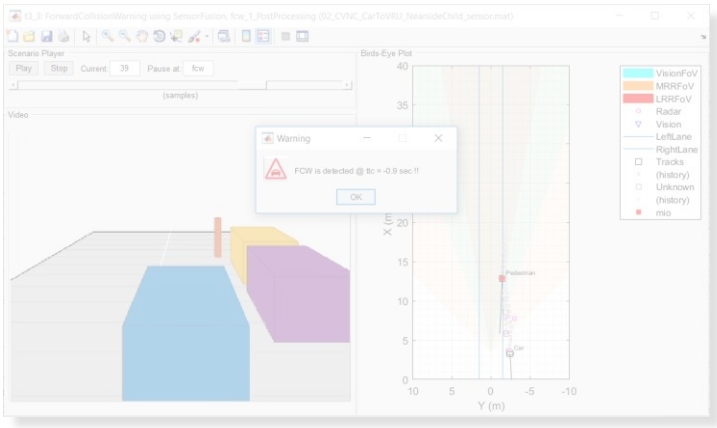
Generate a scenario, simulate sensor detections, and use sensor fusion to track simulated vehicles. The main benefit of using scenario

Open Script

Automated Driving System Toolbox R2017a

Case study: Vision and radar-based sensor fusion





Explore a baseline sensor fusion algorithm

- Read logged vehicle data
- Visualize vehicle data
- Create multi-object tracker
- Implement forward collision warning

Test the algorithm with new data set

- Test the baseline algorithm with new data set
- Tune algorithm parameters
- Customize the algorithm

Synthesize data to further test the algorithm

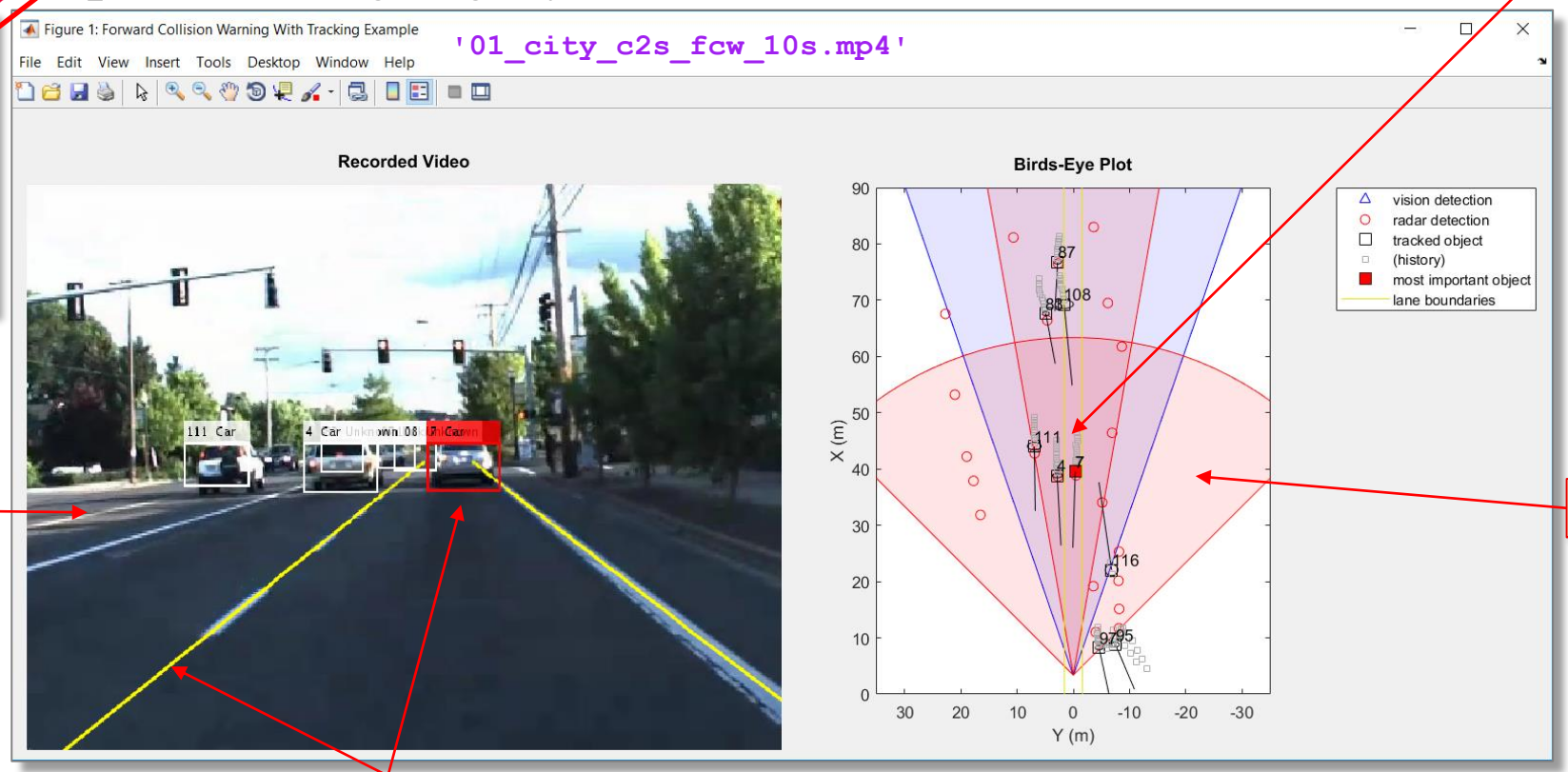
- Create driving scenario
- Add sensor detections
- Generate synthetic data
- Test the algorithm

Explore a baseline sensor fusion algorithm

R2017a

DEMO

t1_ForwardCollisionWarningTrackingExample.m



multiObjectTracker

- objectDetection
- trackingEKF
- trackingUKF
- trackingKF
- updateTracks

Video Display

- VideoReader
- readFrame
- imshow

birdsEyePlot

- coverageAreaPlotter
- plotCoverageArea
- detectionPlotter
- plotDetection
- laneBoundaryPlotter
- plotLaneBoundary
- trackPlotter
- plotTrack

Video Annotation

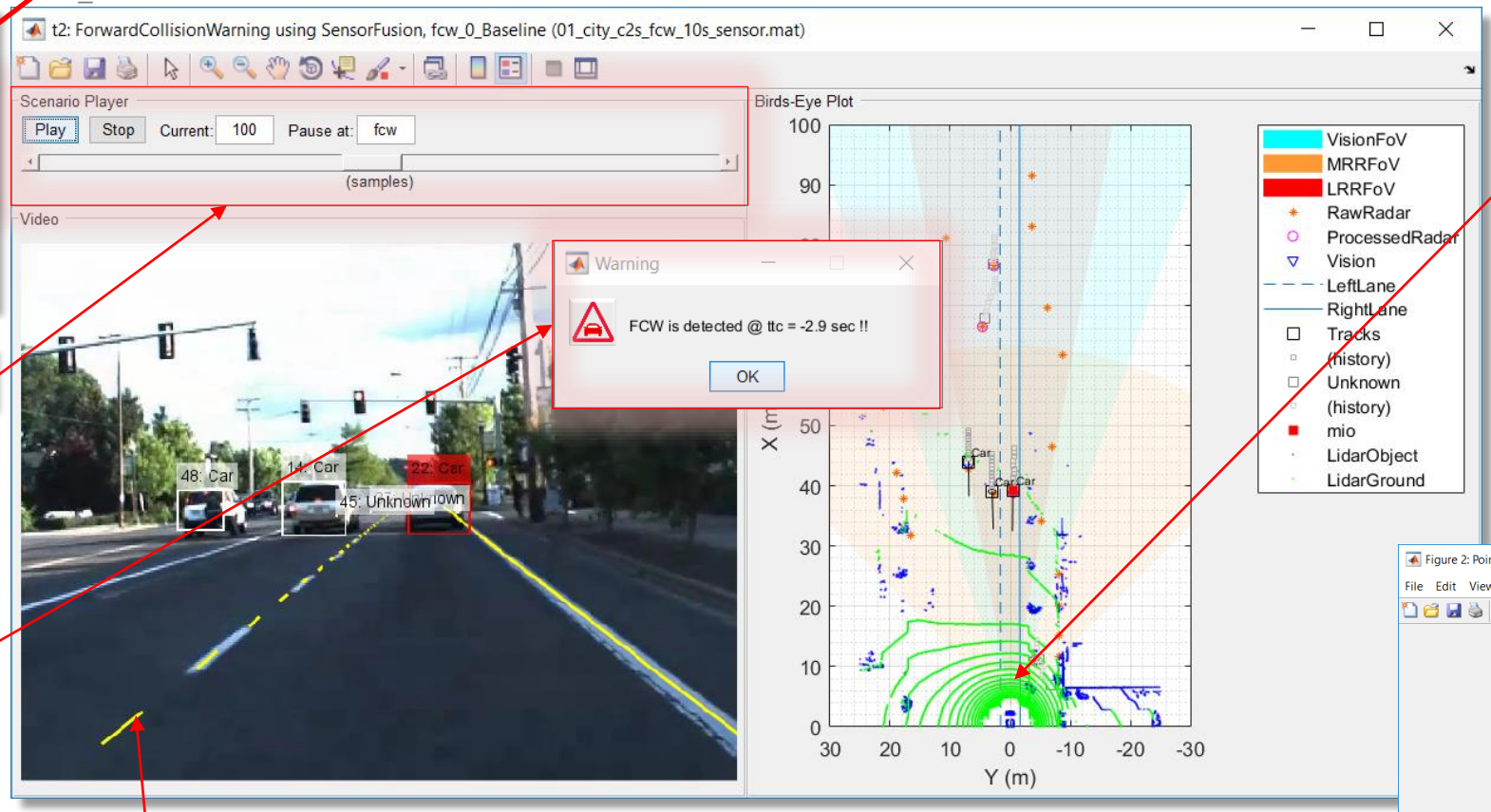
- parabolicLaneBoundary
- insertLaneBoundary
- vehicleToImage
- insertObjectAnnotation

Explore a baseline sensor fusion algorithm + ScenarioPlayer + PointCloud

R2017a

DEMO

t2_fcwSensorFusion.m



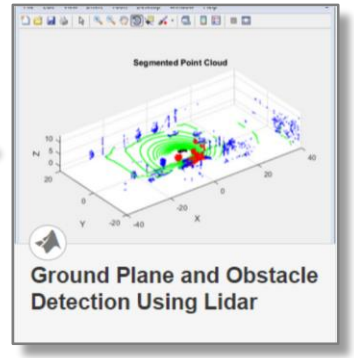
ScenarioPlayer

- play/pause
- stop
- pause at
- slider

Pause at FCW

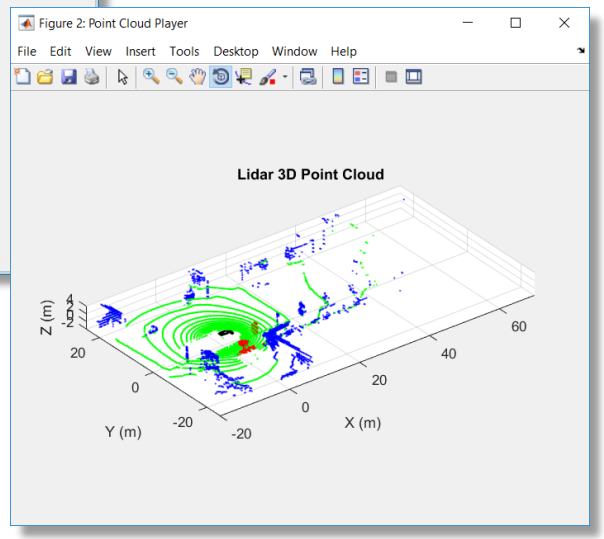
Display LaneBoundaryType

+

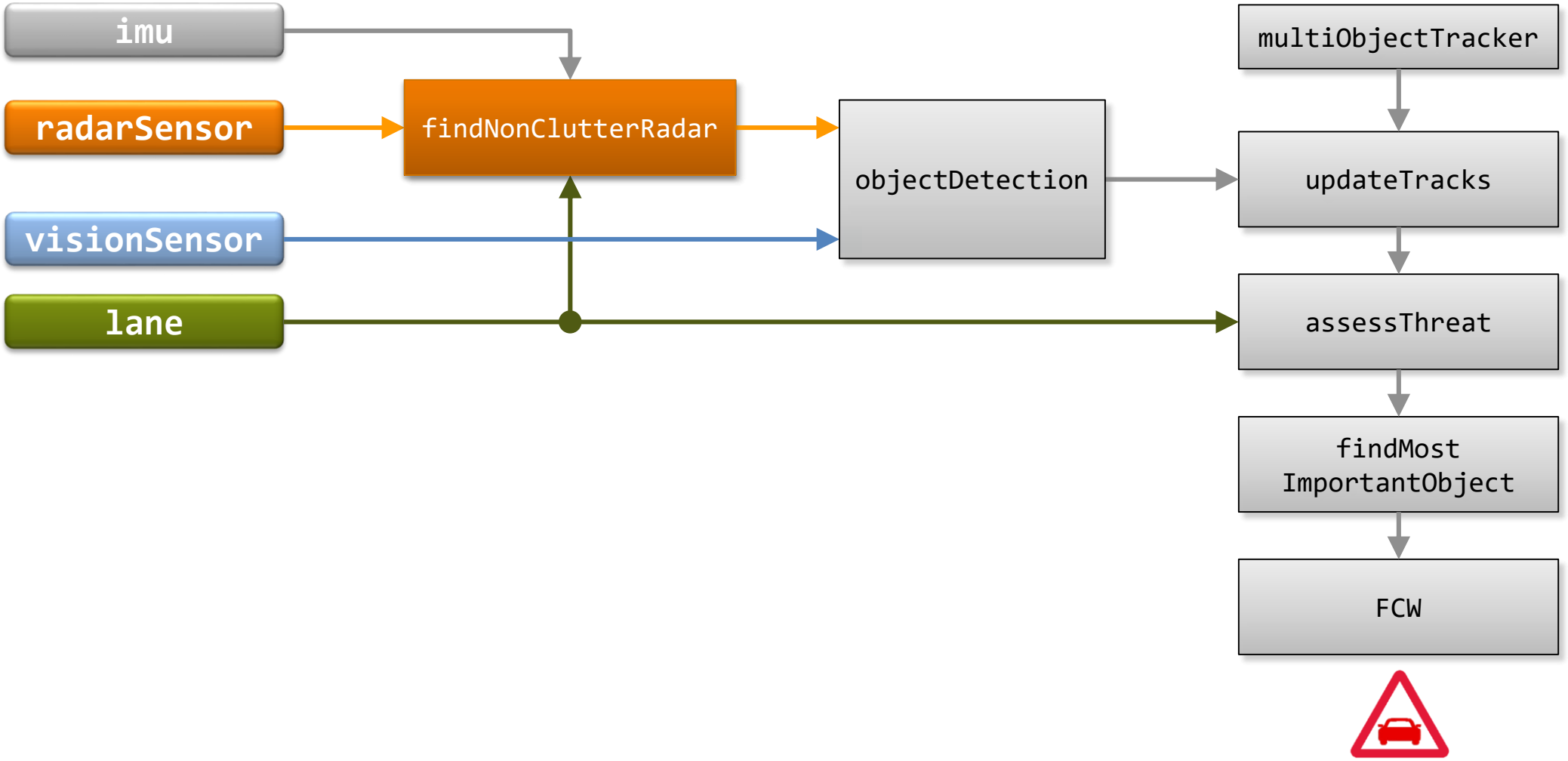


Lidar Point Cloud

- pcplayer
- pointCloud
- findPointsInROI
- findNeighborsInRadius
- select
- pcdenoise/pcdownsample
- pcfitplane

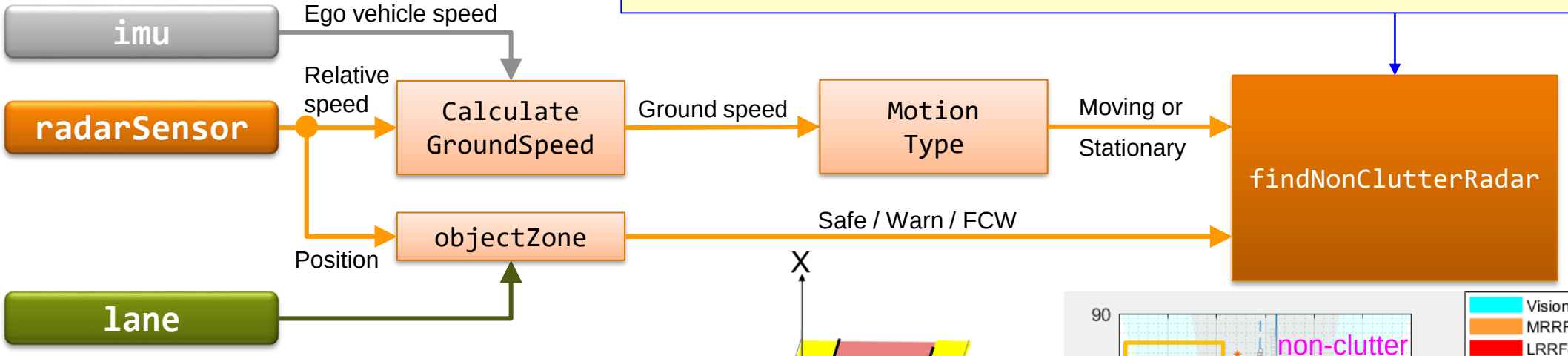


Baseline sensor fusion algorithm

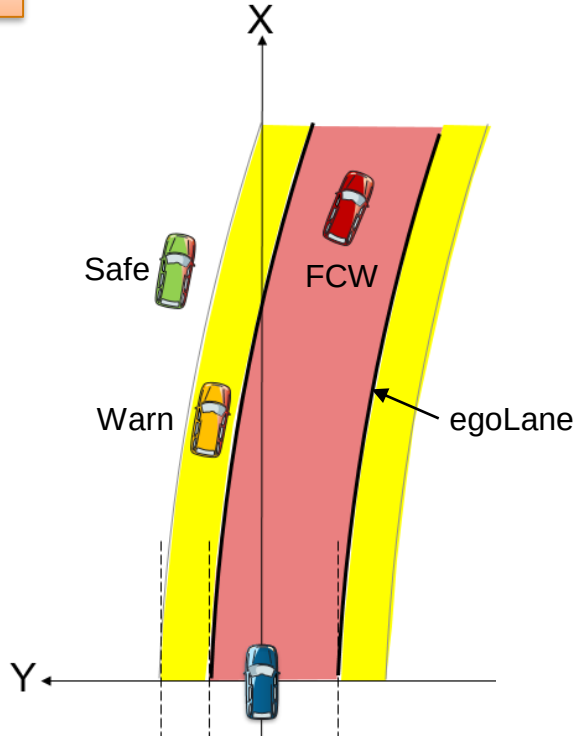


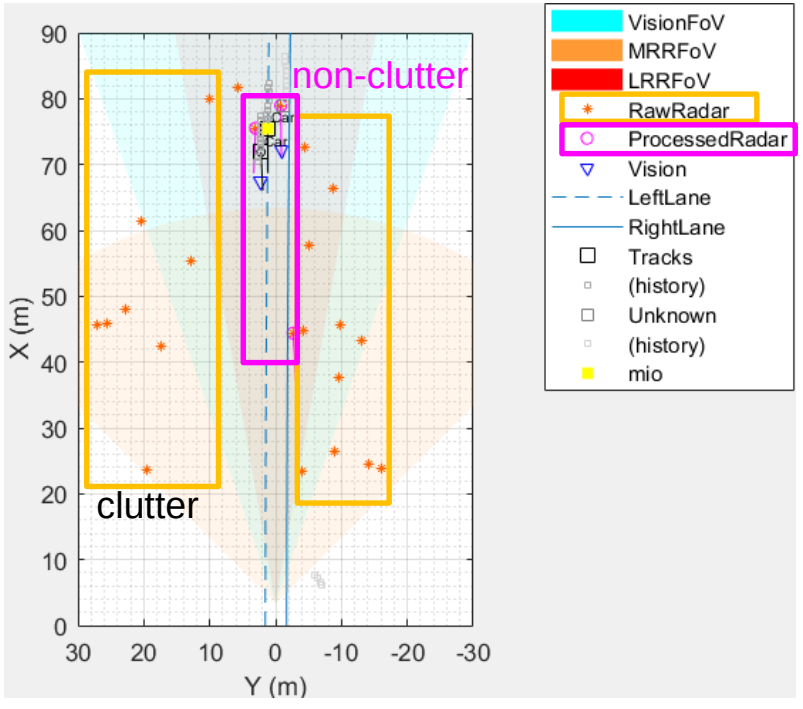
findNonClutterRadar

```
if Zone == zoneEnum.FCW || (Zone ~= zoneEnum.Safe
&& MotionType == objectMotionEnum.Moving)
```

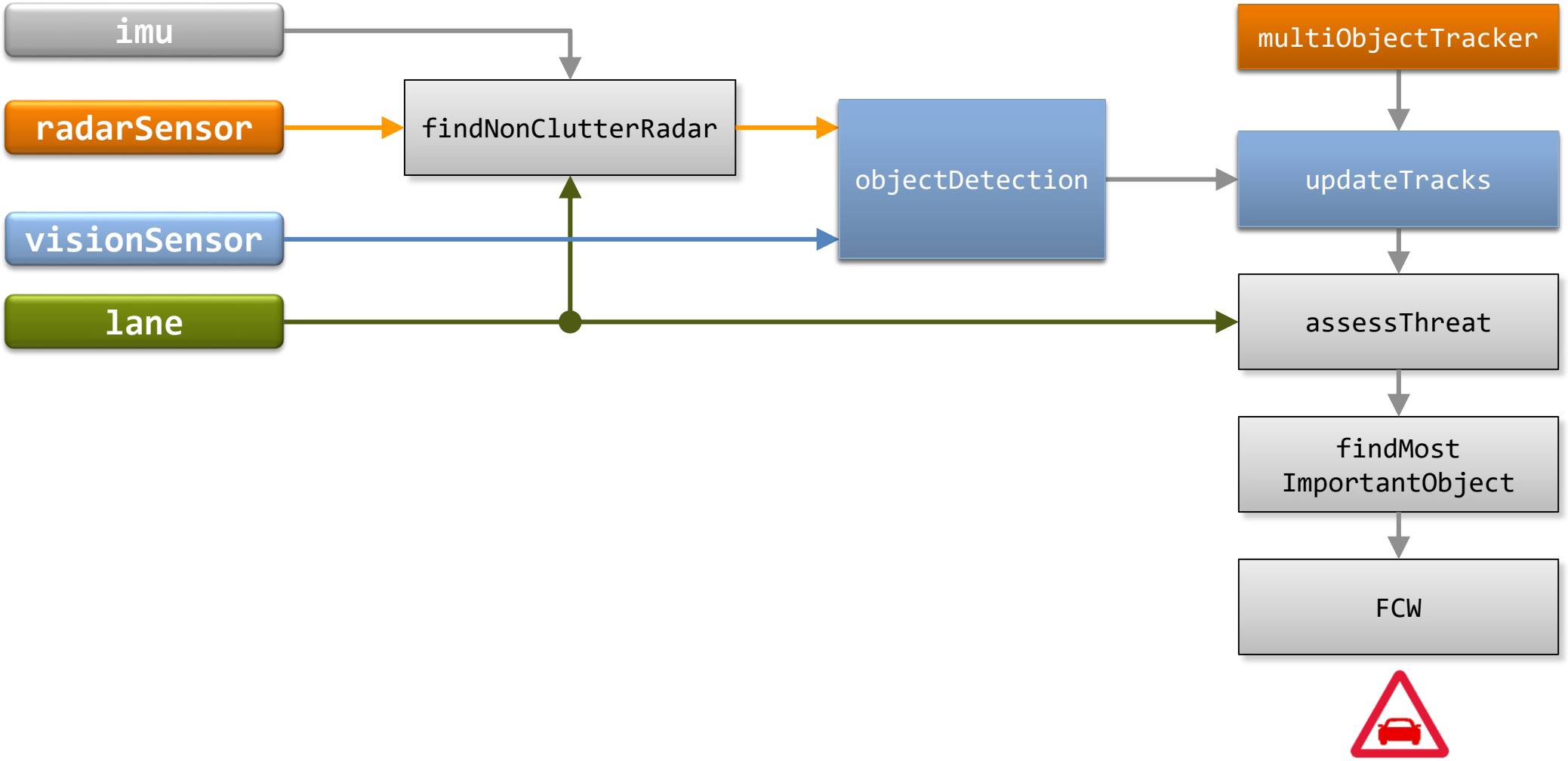



```
enumeration
    Safe  (0)
    Warn  (1)
    FCW   (2)
end
```



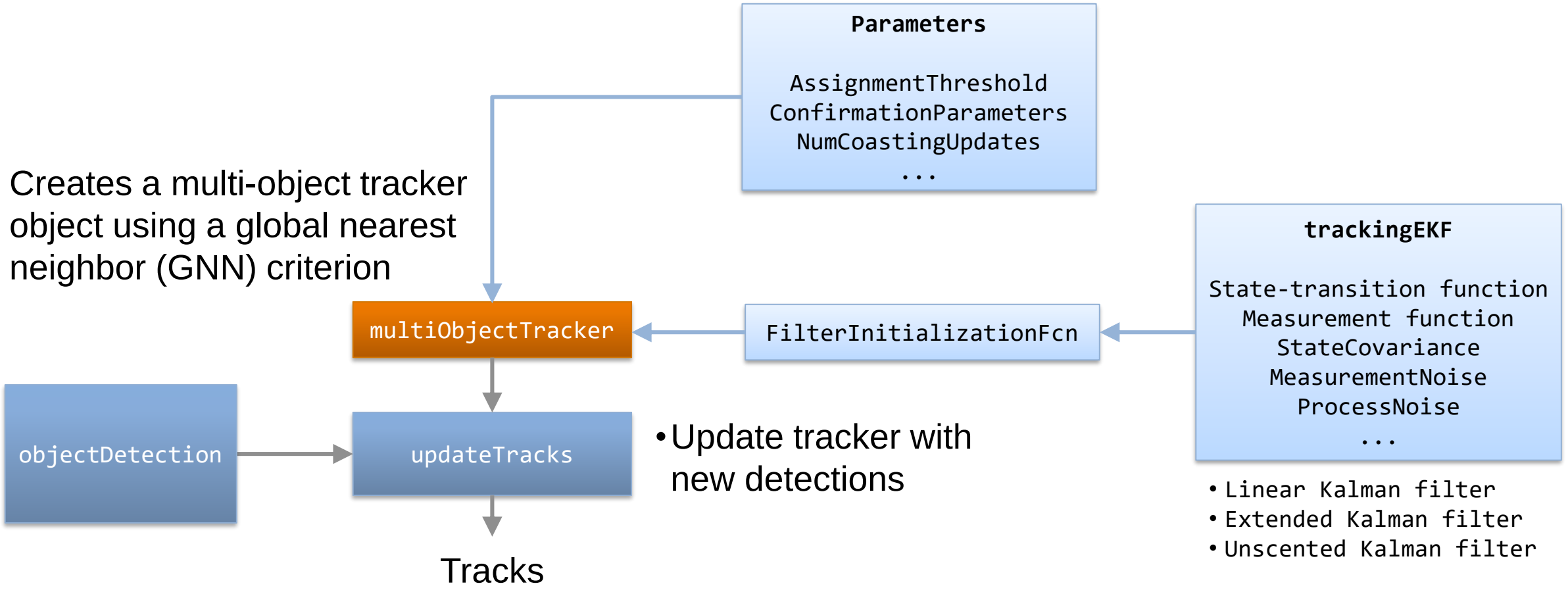


Forward Collision Warning using Sensor Fusion



Multi-Object Tracker

- Creates a multi-object tracker object using a global nearest neighbor (GNN) criterion

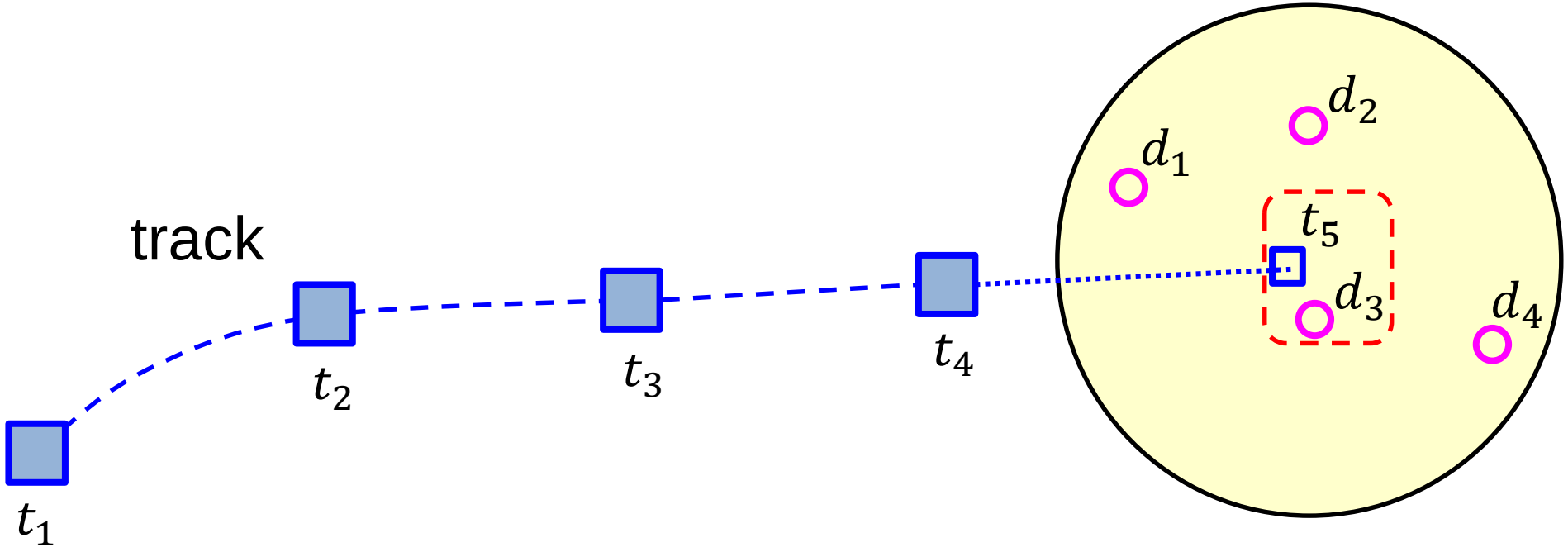


- Update tracker with new detections

- Linear Kalman filter
- Extended Kalman filter
- Unscented Kalman filter

Global Nearest Neighbor (GNN)

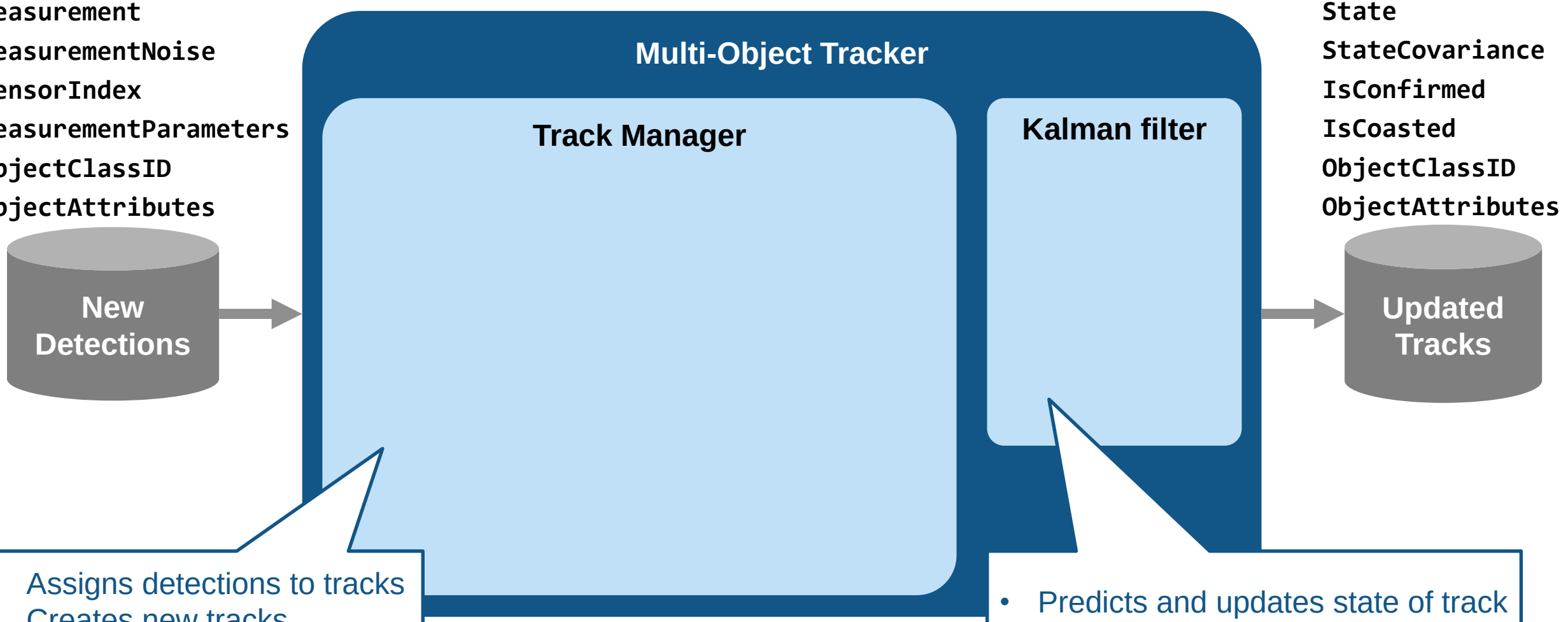
- Evaluate each detection in predicted track region and find “best” one to associate with the track based on Mahalanobis distance and `assignDetectionsToTracks` [1)]



1) James Munkres's variant of the Hungarian assignment algorithm

Multi-Object Tracker

```
Time
Measurement
MeasurementNoise
SensorIndex
MeasurementParameters
ObjectClassID
ObjectAttributes
```



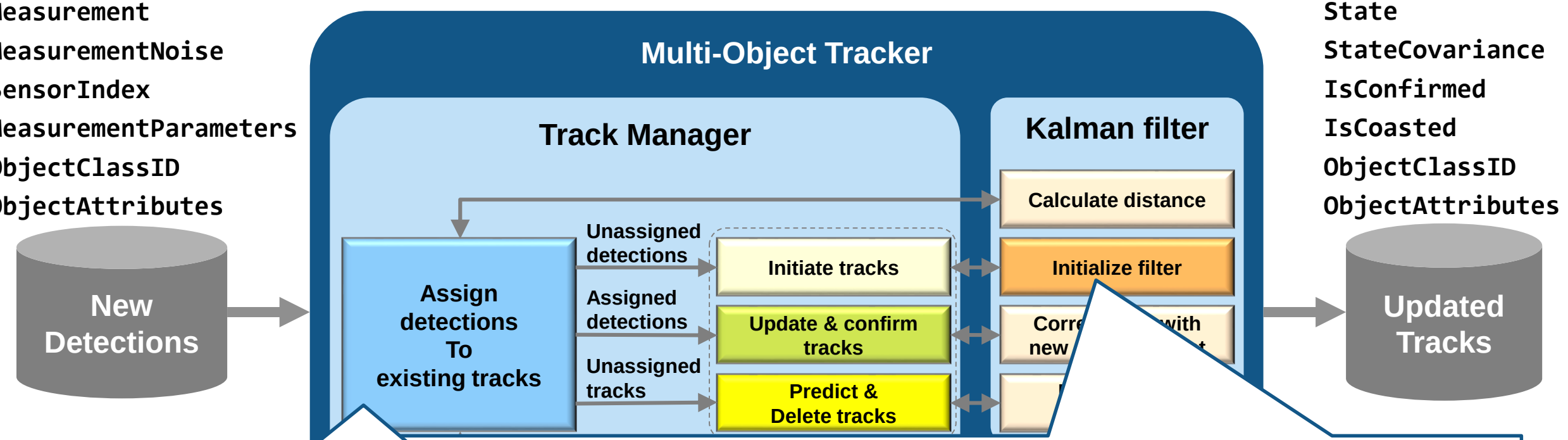

```
TrackID
Time
Age
State
StateCovariance
IsConfirmed
IsCoasted
ObjectClassID
ObjectAttributes
```

- Assigns detections to tracks
- Creates new tracks
- Updates existing tracks
- Removes old tracks

- Predicts and updates state of track
- Supports linear, extended, and unscented Kalman filters

Multi-Object Tracker

Time
Measurement
MeasurementNoise
SensorIndex
MeasurementParameters
ObjectClassID
ObjectAttributes



TrackID
Time
Age
State
StateCovariance
IsConfirmed
IsCoasted
ObjectClassID
ObjectAttributes

trackingKF, trackingUKF

```
ction filter = initcaekf(detection)

trackingEKF(@constacc, ... % State-transition function (const-accel model)
```

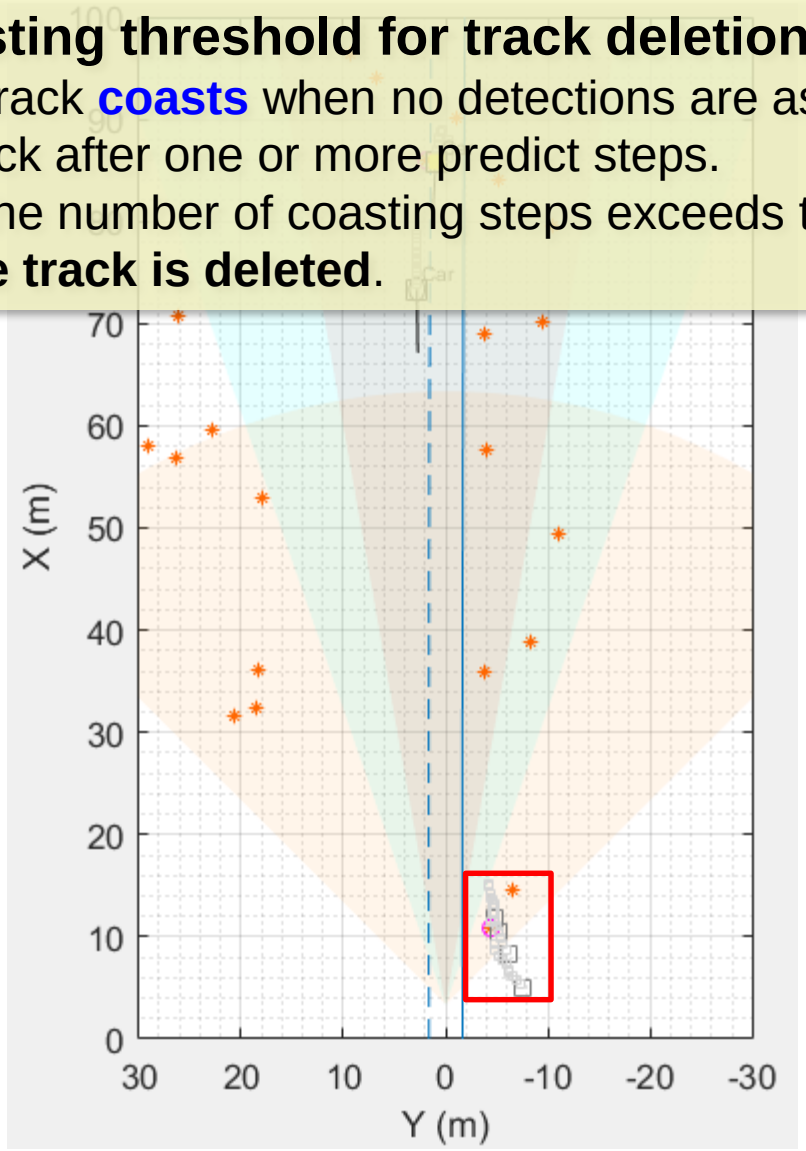
```
tracker = multiObjectTracker(...
    'FilterInitializationFcn', @initcaekf, ... % Handle to tracking Kalman filter
    'AssignmentThreshold', 35, ... % Normalized distance from track for assignment
    'ConfirmationParameters', [2 3], ... % Confirmation parameters for track creation
    'NumCoastingUpdates', 5); % Coasting threshold for track deletion
```

```
easurement noise covar
on function Jacobian
nction
ce
```

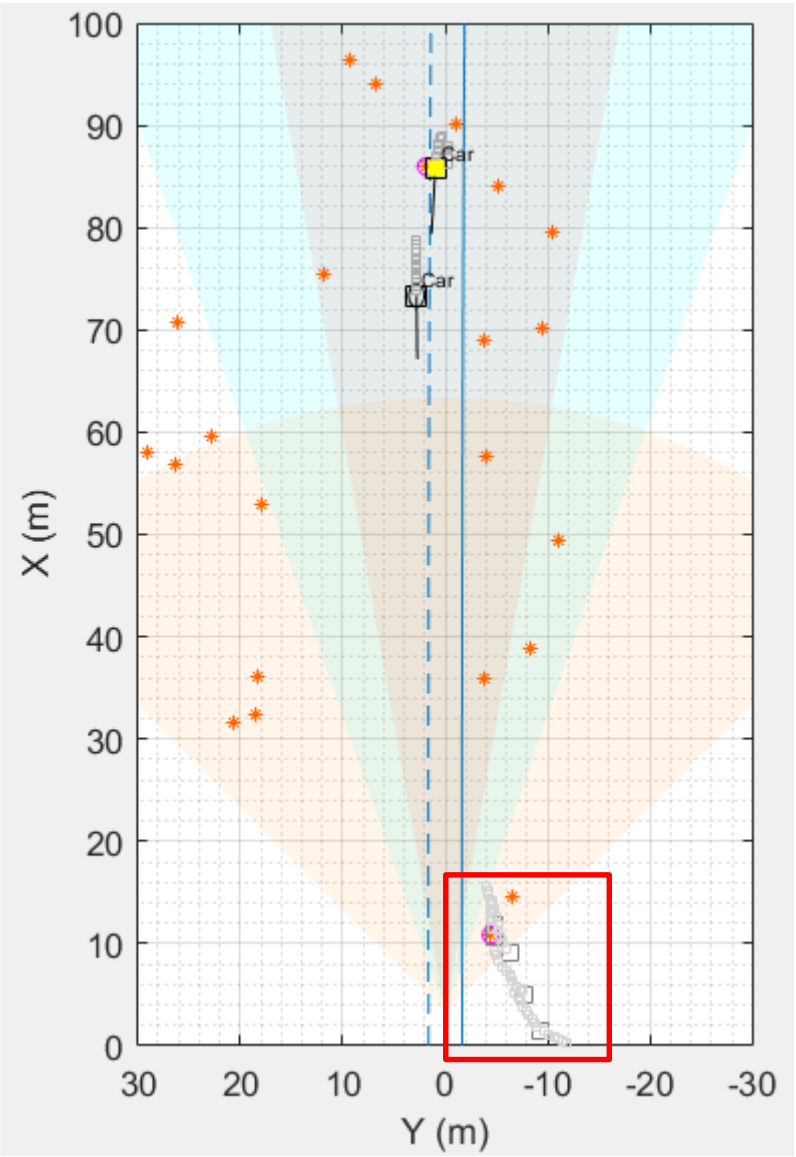
NumCoastingUpdates

Coasting threshold for track deletion.

- A track **coasts** when no detections are assigned to the track after one or more predict steps.
- If the number of coasting steps exceeds this threshold, **the track is deleted**.



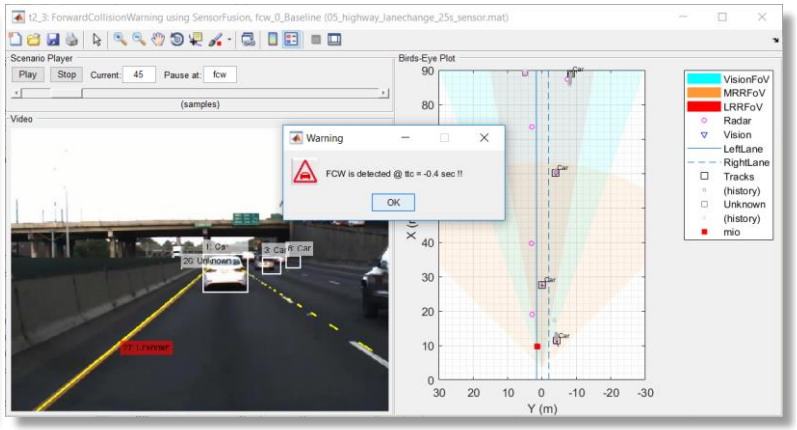
'NumCoastingUpdates', 5

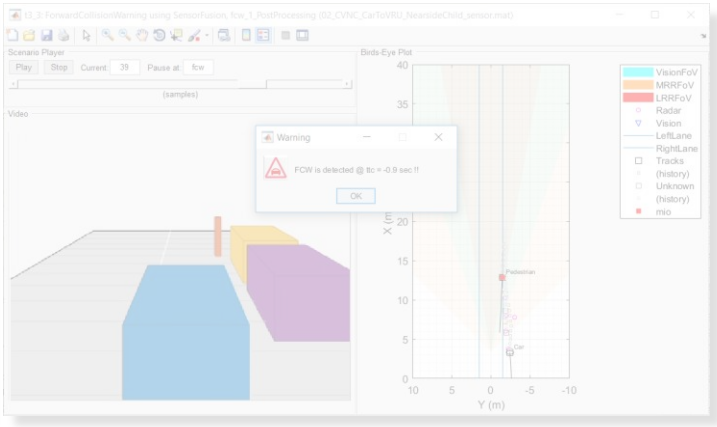


'NumCoastingUpdates', 20

Automated Driving System Toolbox R2017a

Case study: Vision and radar-based sensor fusion





Explore a baseline sensor fusion algorithm

- Read logged vehicle data
- Visualize vehicle data
- Create multi-object tracker
- Implement forward collision warning

Test the algorithm with new data set

- Test the baseline algorithm with new data set
- Tune algorithm parameters
- Customize the algorithm

Synthesize data to further test the algorithm

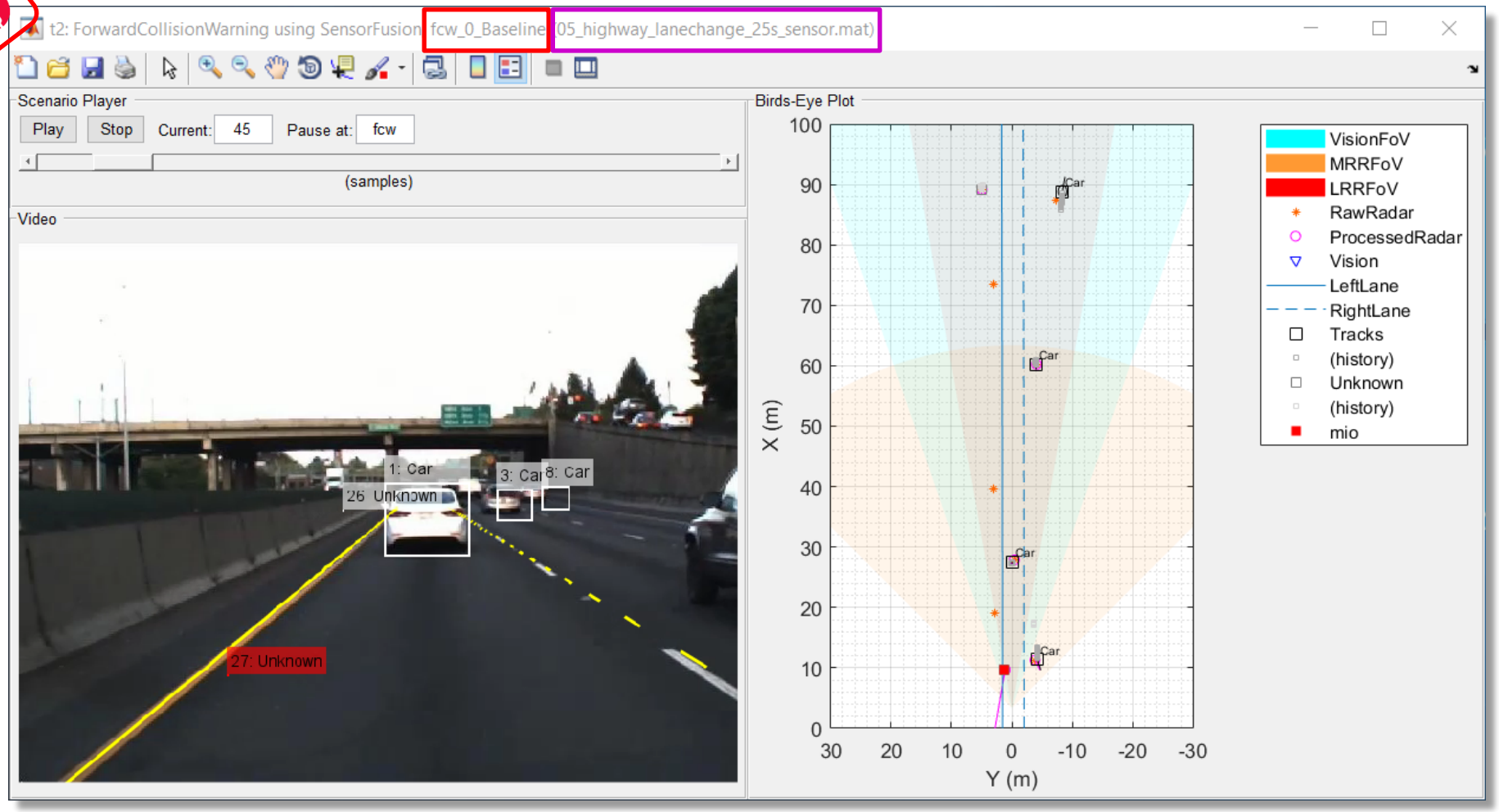
- Create driving scenario
- Add sensor detections
- Generate synthetic data
- Test the algorithm
- Generate C code

Test the baseline algorithm for new data set

DEMO

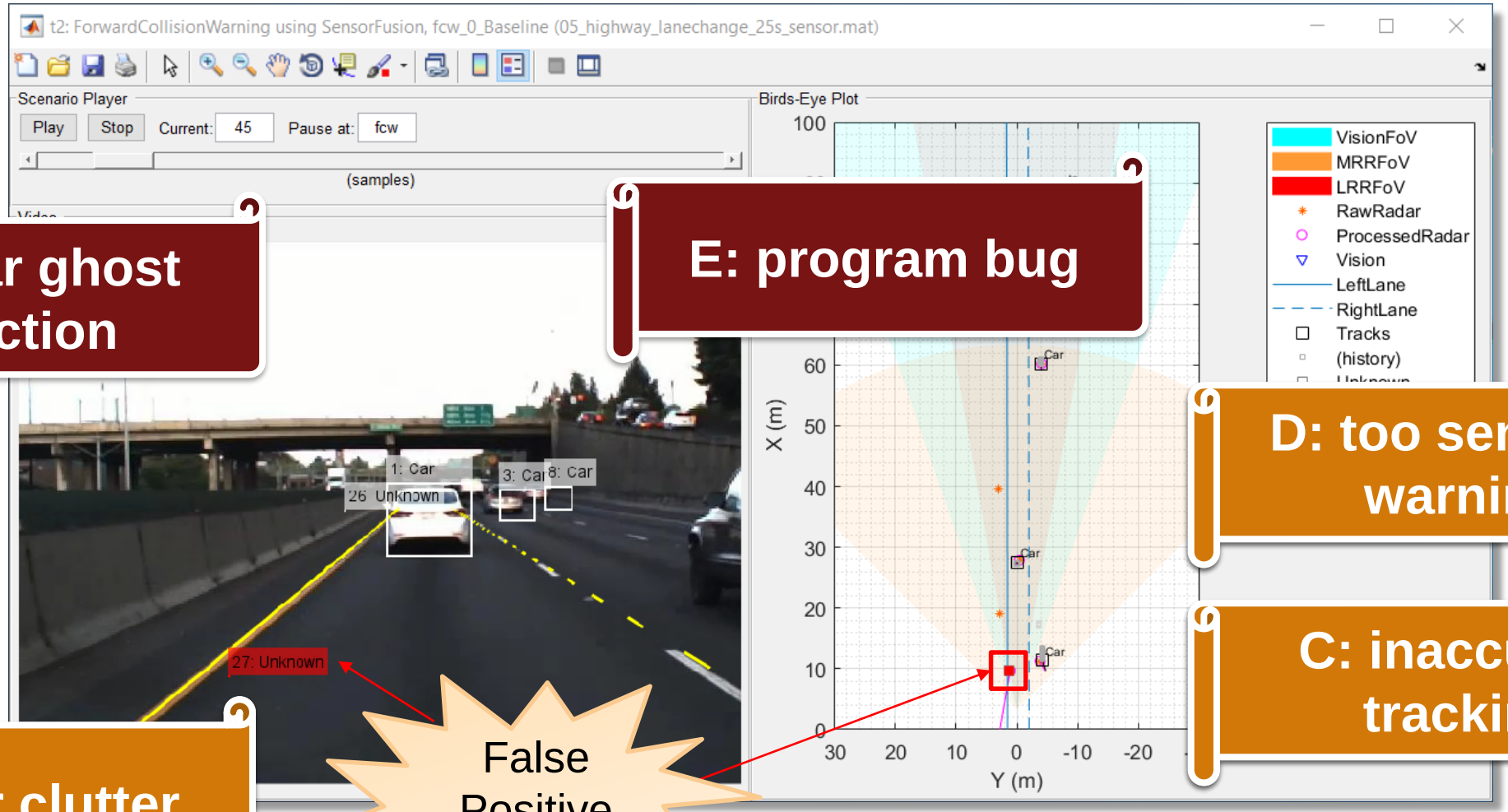
t2_fcwSensorFusion.m

'fcw_0_Baseline' '05_highway_lanechange_25s.mp4'



Pop Quiz

What causes the false positive?



A: radar ghost detection

E: program bug

D: too sensitive warning

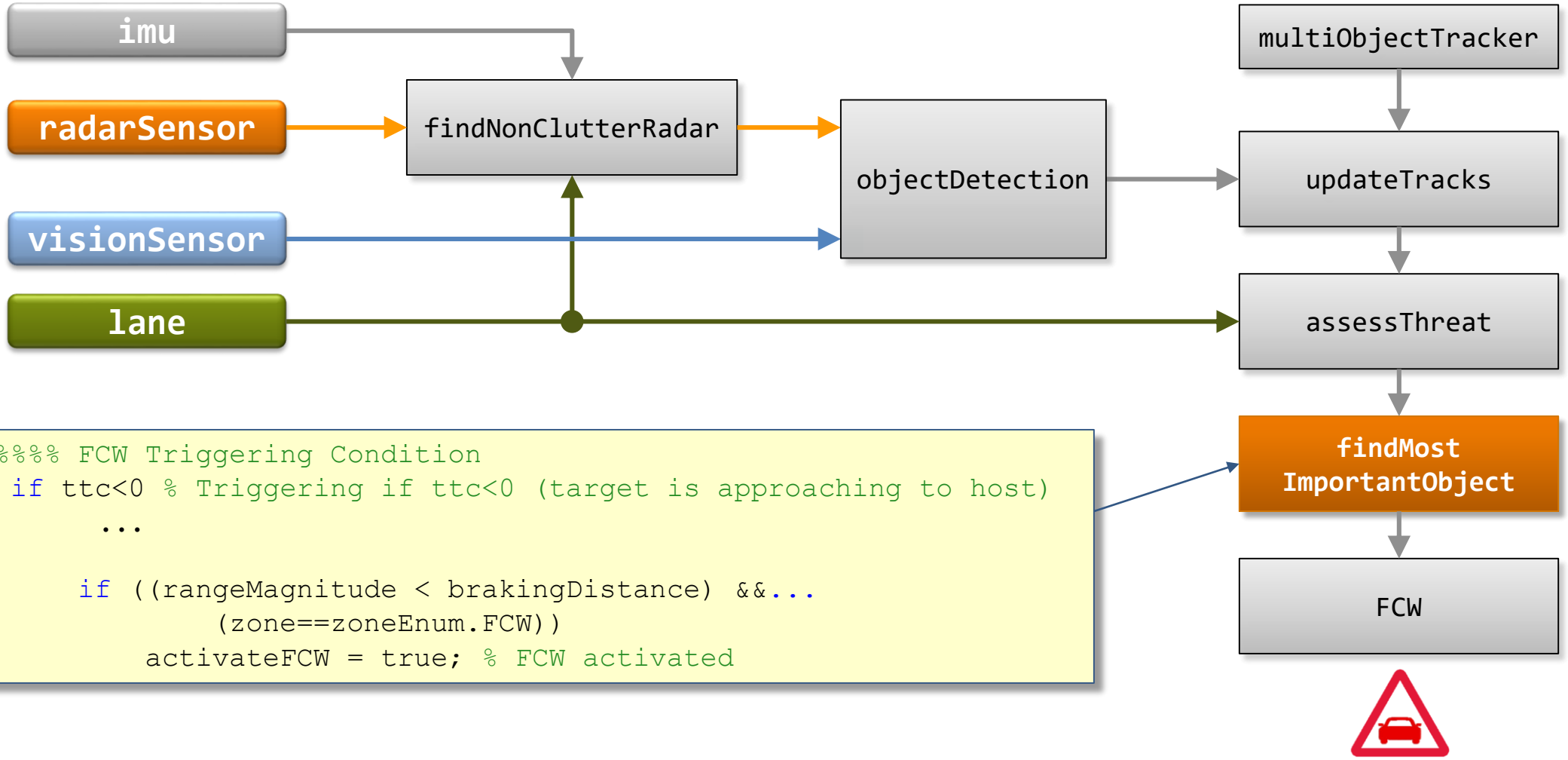
C: inaccurate tracking

B: radar clutter

False Positive

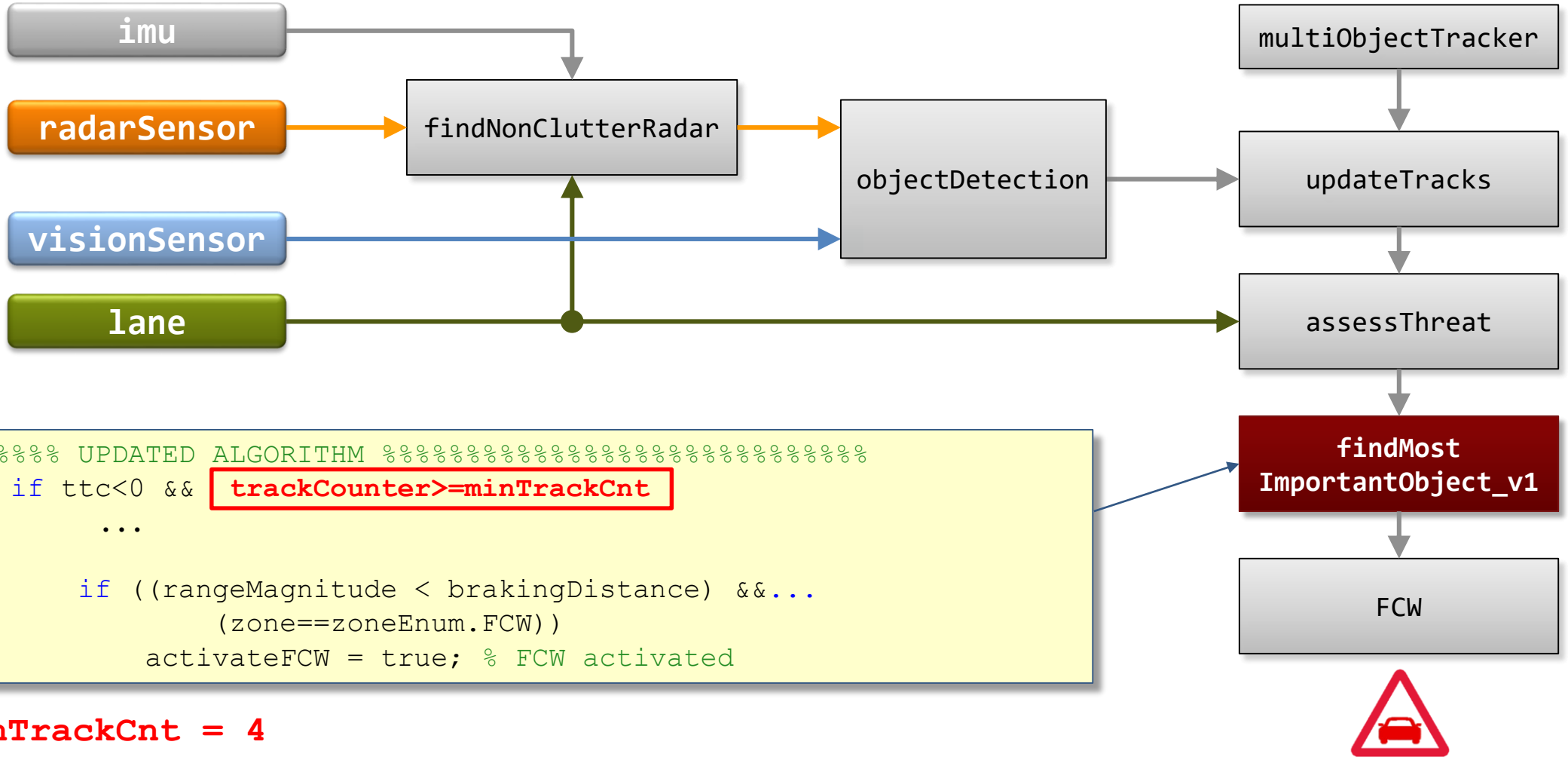
Baseline sensor fusion algorithm (*fcw_0_Baseline*)

D: too sensitive warning




```
%%%%%% FCW Triggering Condition
   if ttc<0 % Triggering if ttc<0 (target is approaching to host)
        ...

       if ((rangeMagnitude < brakingDistance) &&...
             (zone==zoneEnum.FCW))
          activateFCW = true; % FCW activated
```

Modified algorithm (*fcw_1_PostProcessing*)




```
%%%%%% UPDATED ALGORITHM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   if ttc<0 && trackCounter>=minTrackCnt
         ...

       if ((rangeMagnitude < brakingDistance) &&...
             (zone==zoneEnum.FCW))
          activateFCW = true; % FCW activated
```

minTrackCnt = 4

fcw_0_Baseline vs. *fcw_1_PostProcessing*

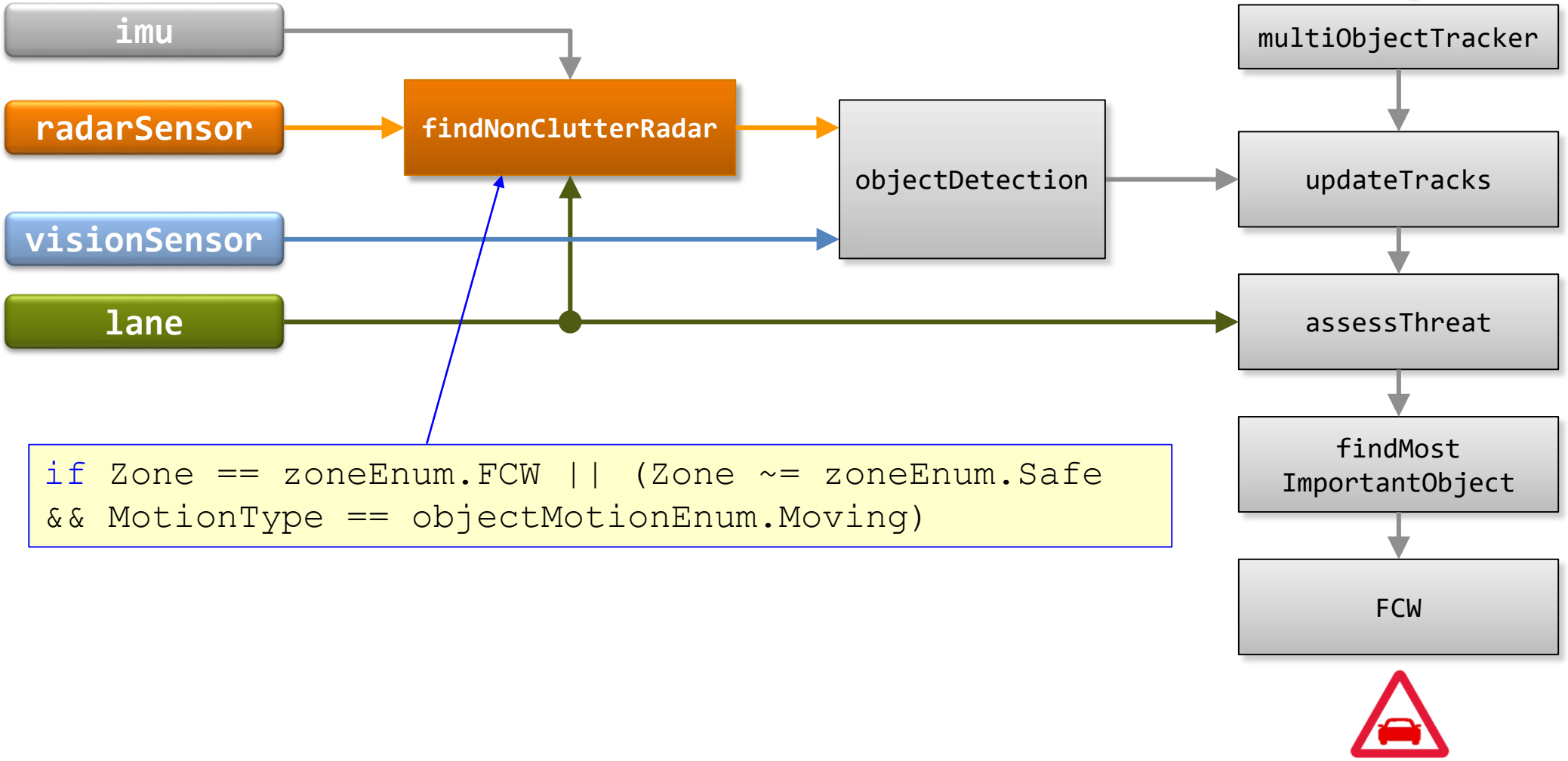
fcw_0_Baseline

fcw_1_PostProcessing

Baseline sensor fusion algorithm (*fcw_0_Baseline*)

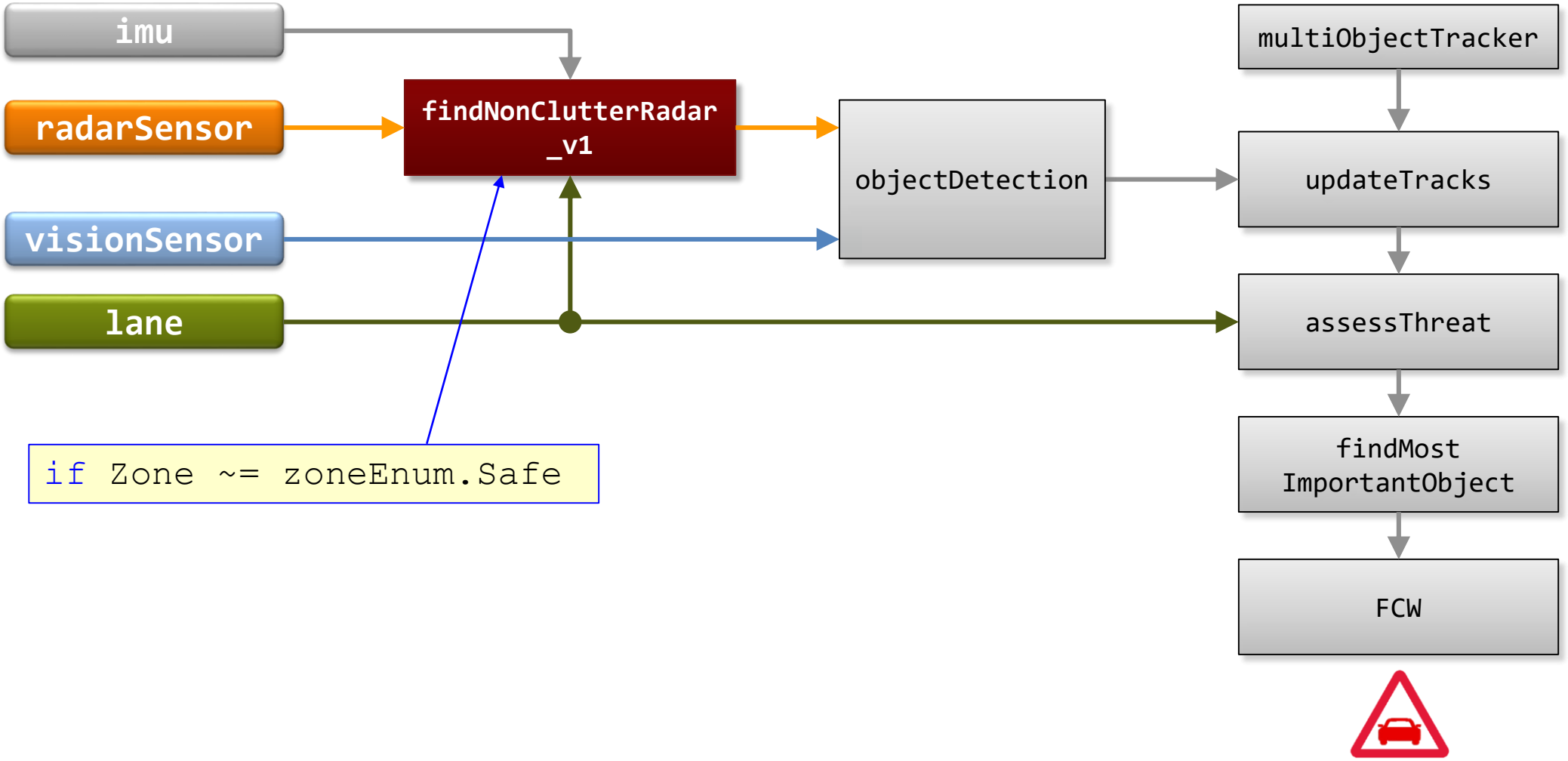
B: radar clutter

C: inaccurate tracking




```
if Zone == zoneEnum.FCW || (Zone ~= zoneEnum.Safe
&& MotionType == objectMotionEnum.Moving)
```

Modified algorithm (fcw_2_PreProcessing)



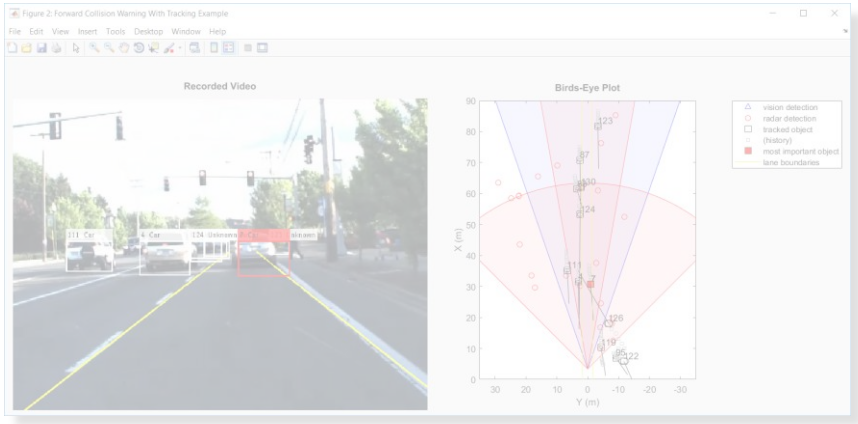
fcw_0_Baseline vs. *fcw_2_PreProcessing*

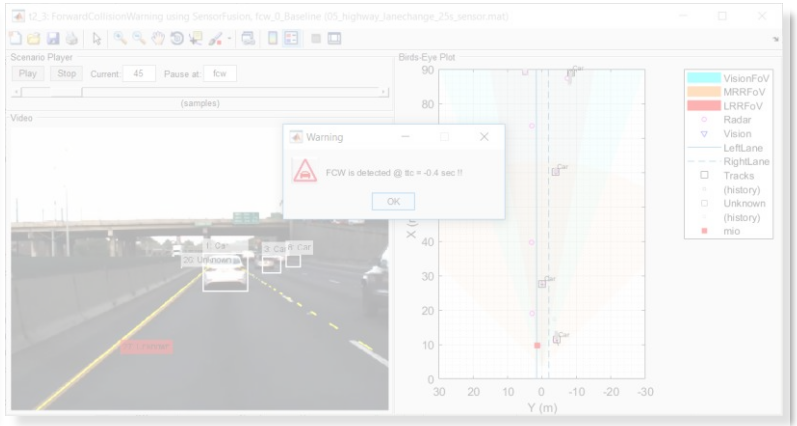
fcw_0_Baseline

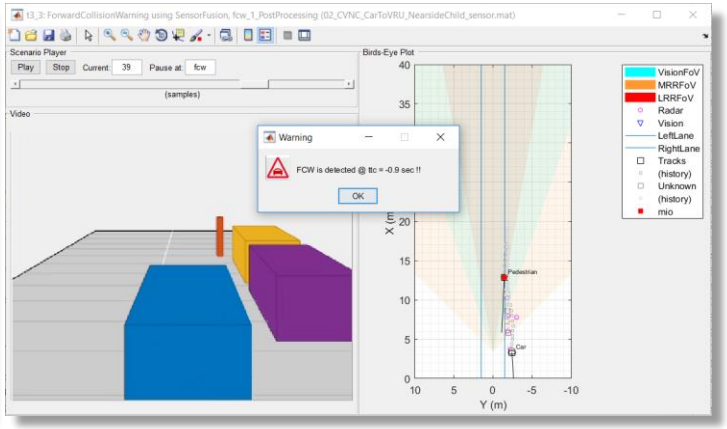
fcw_2_PreProcessing

Automated Driving System Toolbox R2017a

Case study: Vision and radar-based sensor fusion







Explore a baseline sensor fusion algorithm

- Read logged vehicle data
- Visualize vehicle data
- Create multi-object tracker
- Implement forward collision warning

Test the algorithm with new data set

- Test the baseline algorithm with new data set
- Tune algorithm parameters
- Customize the algorithm

Synthesize data to further test the algorithm

- Create driving scenario
- Add sensor detections
- Generate synthetic data
- Test the algorithm

Euro NCAP TEST PROTOCOL – AEB VRU systems
Car-to-VRU Nearside Child (CVNC)

- a collision in which a vehicle travels forwards towards a child pedestrian crossing it's path running from behind and obstruction from the nearside and the frontal structure of the vehicle strikes the pedestrian at 50% of the vehicle's width when no braking action is applied.

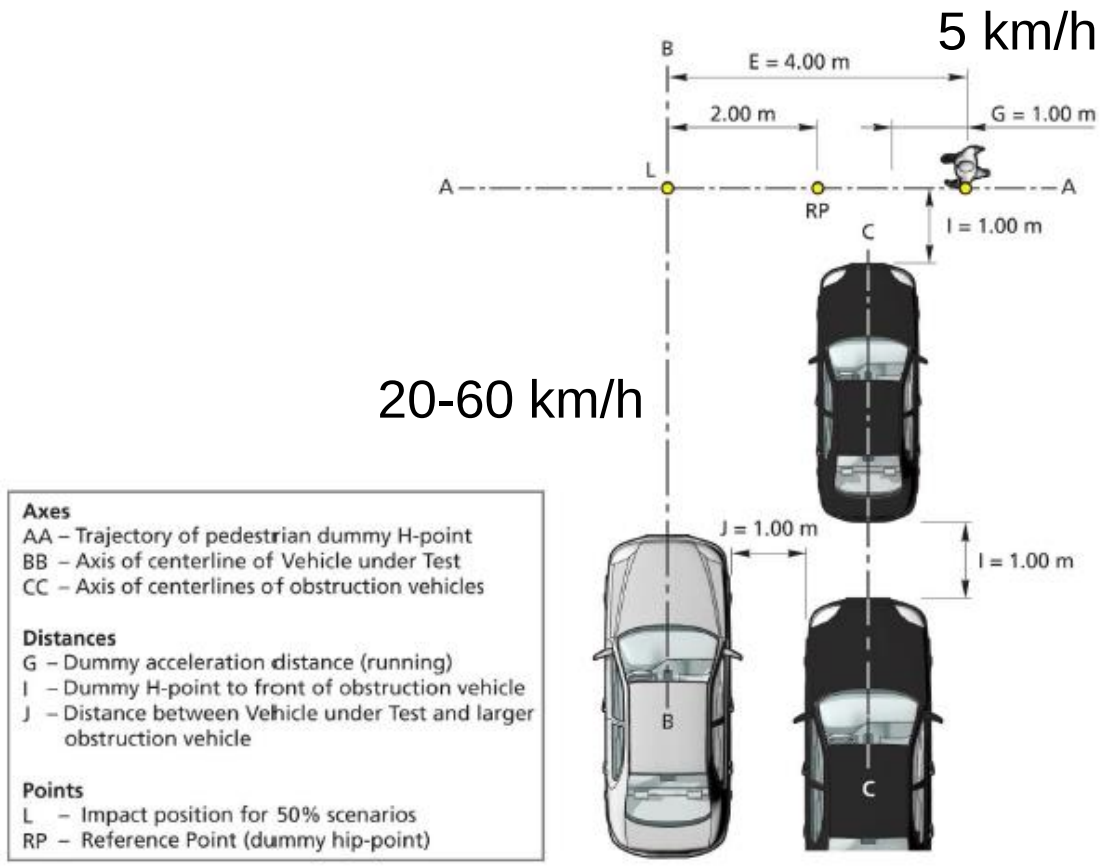


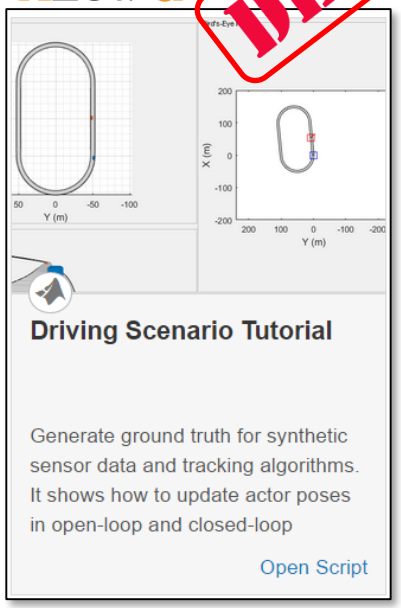
Figure 8c: CVNC scenario, Running Child from Nearside from Obstruction vehicles (see Annex B)

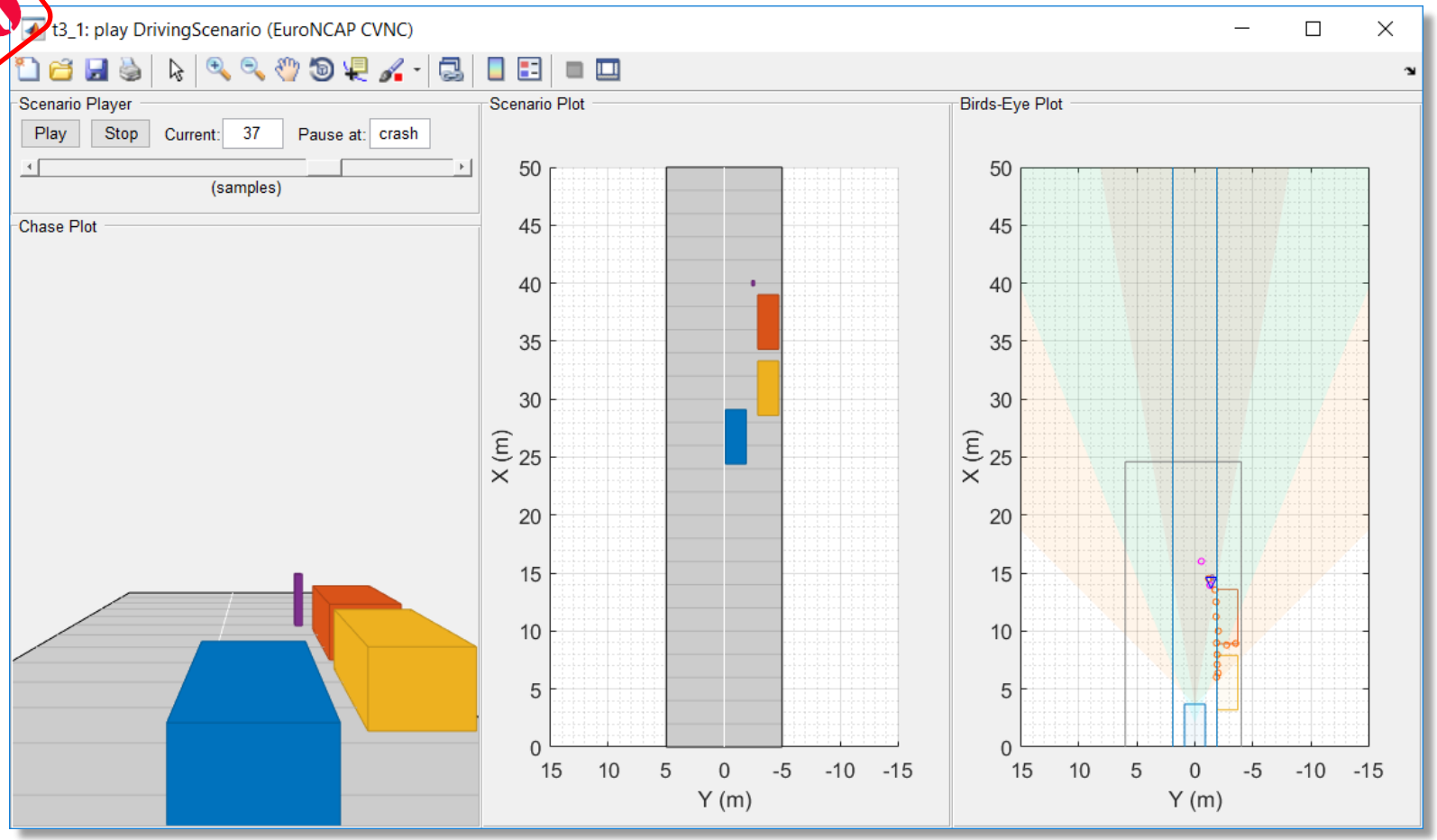
Create Driving Scenario

t3_1_playDrivingScenario.m

DEMO

R2017a





birdsEyePlot

- coverageAreaPlotter
- plotCoverageArea
- detectionPlotter
- plotDetection
- laneBoundaryPlotter
- plotLaneBoundary
- trackPlotter
- plotTrack

drivingScenario

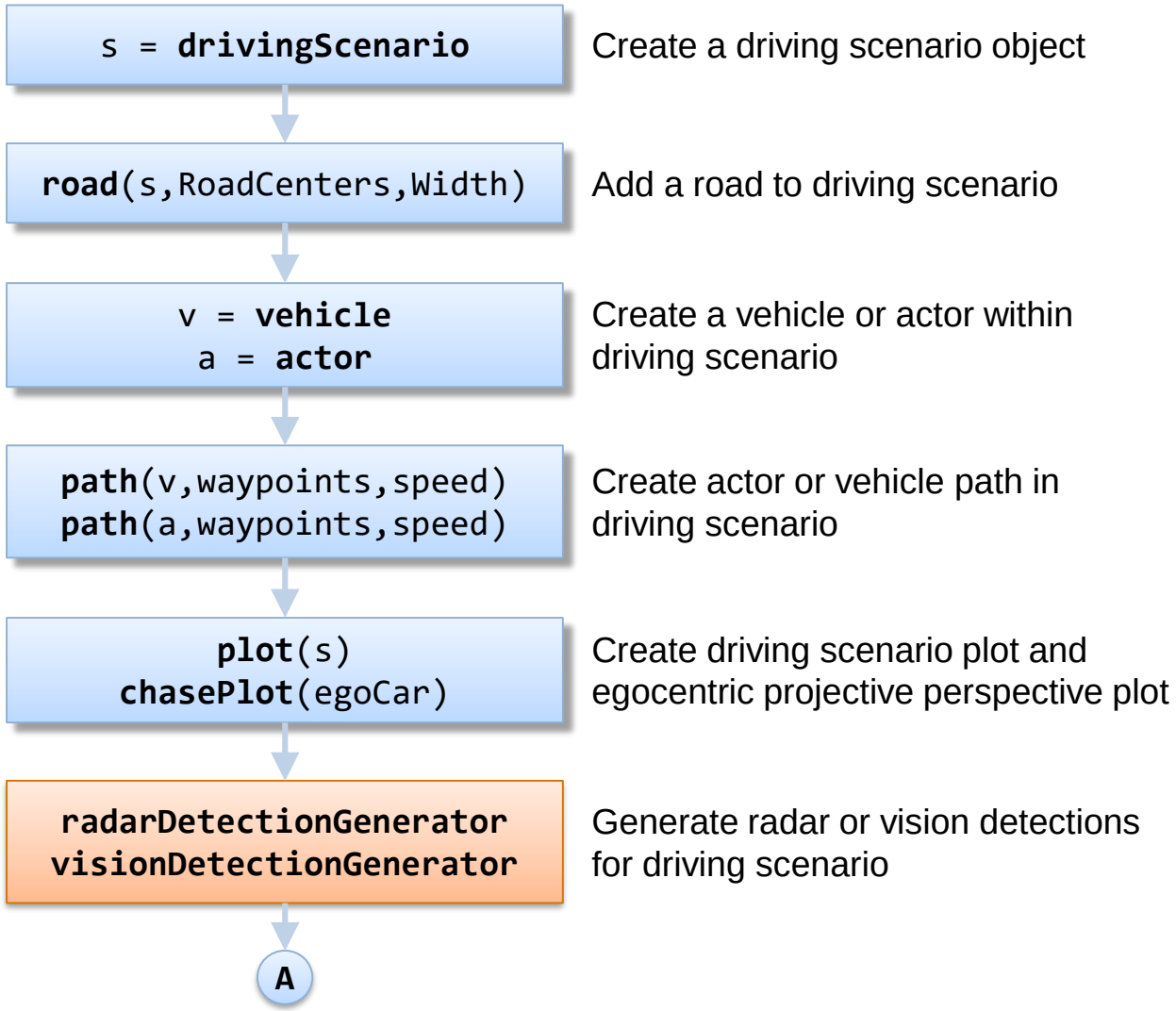
- chasePlot
- road
- vehicle
- actor
- path
- plot
- radarDetectionGenerator
- visionDetectionGenerator

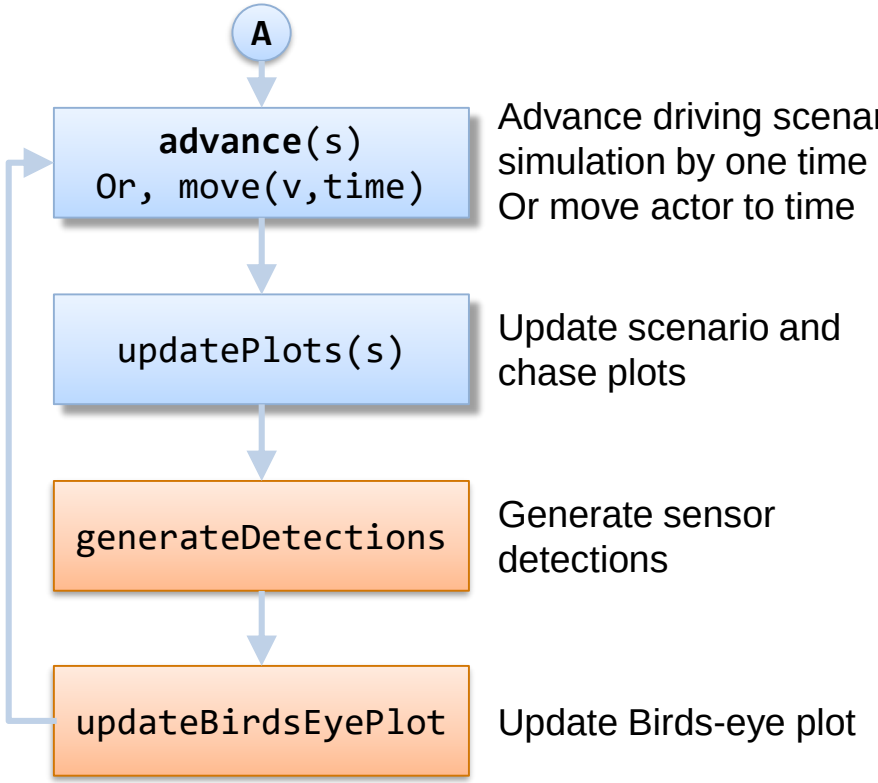
Egocentric projective perspective plot

driving scenario plot

Generate radar or vision detections for driving scenario

Workflow for driving scenario





Sensor model for vision detection

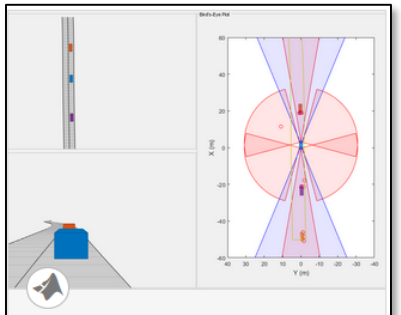
```
visionSensor = visionDetectionGenerator(...
    'SensorIndex',       3, ...                % Unique sensor identifier
    'UpdateInterval',    0.1, ...              % Required interval between sensor updates (sec)
    'SensorLocation',    [2.1,0], ...          % Location of sensor center in vehicle coord. [x,y](meters)
    'Height',            1.1, ...              % Sensor's mounting height from the ground (meters)
    'Yaw',               0, ...                % Yaw angle of sensor mounted on ego vehicle (deg)
    'Pitch',             0, ...                % Pitch angle of sensor mounted on ego vehicle (deg)
    'Roll',              0, ...                % Roll angle of sensor mounted on ego vehicle (deg)
    'Intrinsics',        cameraIntrinsics( ... % Sensor's intrinsic camera parameters
                           [800,800], ...        % [fx, fy] in pixels
                           [320,240], ...        % [cx, cy] in pixels
                           [480,640]), ...       % [numRows, numColumns] in pixels
    'MaxRange',          150, ...              % Maximum detection range (meters)
    'MaxSpeed',          50, ...               % Maximum detectable object speed (m/s)
    'MaxAllowedOcclusion', 0.5, ...            % Maximum allowed occlusion [0 1]
    'DetectionProbability',  0.9, ...          % Probability of detecting a target (fraction)
    'FalsePositivesPerImage',0.1);             % Number of false positives per image (fraction)
```

Sensor model for radar detection

```
radarSensor = radarDetectionGenerator(...
    'SensorIndex',          1, ...          % Unique sensor identifier
    'UpdateInterval',       0.05, ...       % Required interval between sensor updates (sec)
    'SensorLocation',       [3.7,0], ...    % Location of sensor center in vehicle coord. [x,y] in meters
    'Height',               0.2, ...        % Sensor's mounting height from the ground (meters)
    'Yaw',                  0, ...          % Yaw angle of sensor mounted on ego vehicle (deg)
    'Pitch',                0, ...          % Pitch angle of sensor mounted on ego vehicle (deg)
    'Roll',                 0, ...          % Roll angle of sensor mounted on ego vehicle (deg)
    'FieldOfView',          [90,4.5], ...   % Total angular field of view for radar [az el] in deg
    'MaxRange',             60, ...         % Maximum detection range (meters)
    'RangeRateLimits',      [-100,40], ...  % Minimum and maximum range rates [min,max] in m/s
    'DetectionProbability',0.9, ...         % Detection probability
    'FalseAlarmRate',       1e-6, ...       % Rate at which false alarms are reported
    'AzimuthResolution',    12, ...         % Azimuthal resolution of radar (deg)
    'RangeResolution',      1.25, ...       % Range resolution of radar (meters)
    'RangeRateResolution', 0.5, ...         % Range rate resolution of radar (m/s)
    'RangeBiasFraction',    0.05);          % Fractional range bias component of radar
```

Generate Synthetic Data from Driving Scenario

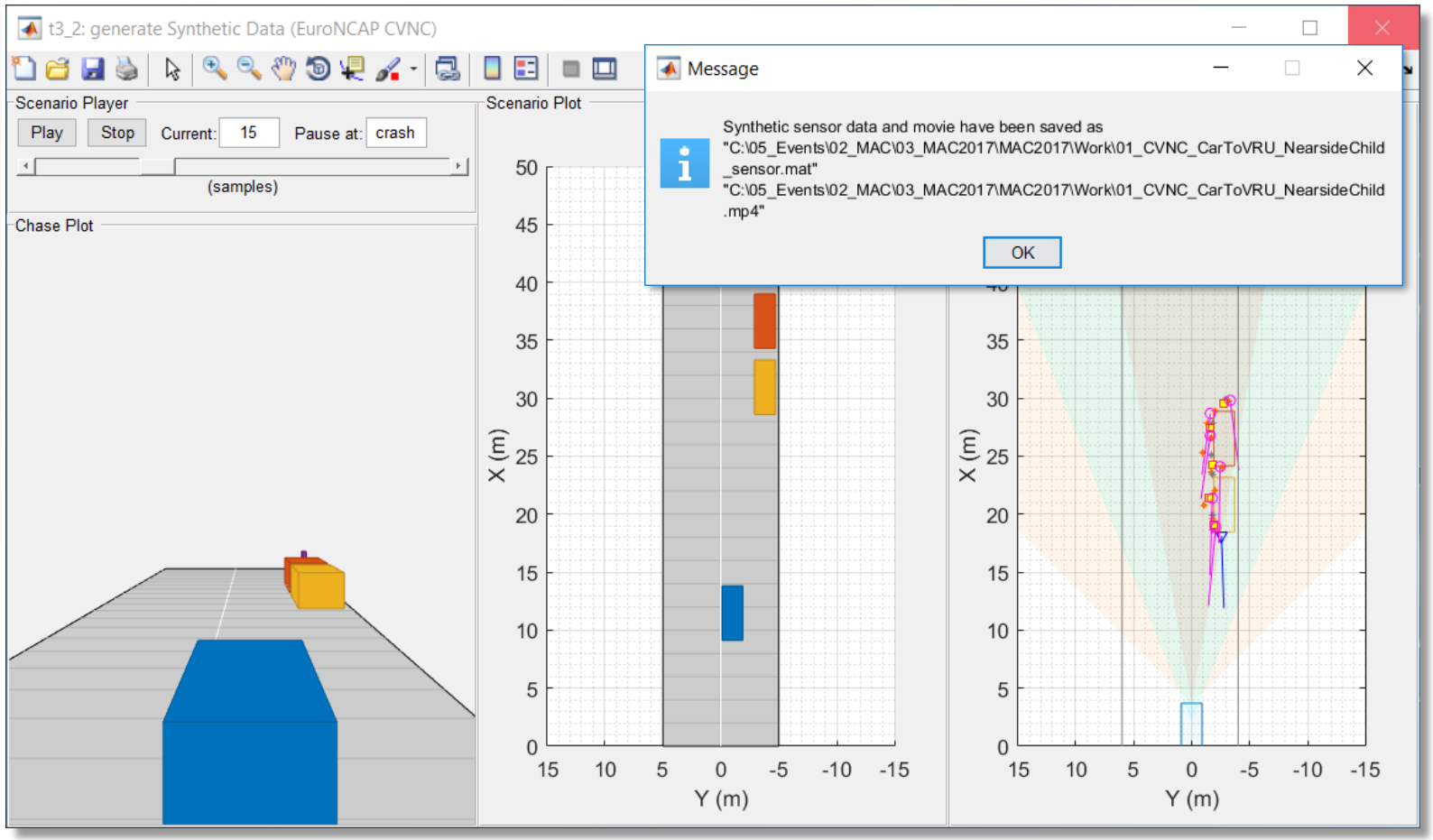
R2017a



Sensor Fusion Using Synthetic Radar and Vision Data

Generate a scenario, simulate sensor detections, and use sensor fusion to track simulated vehicles. The main benefit of using scenario

Open Script



drivingScenario

- road
- vehicle
- actor
- actorProfiles
- path
- plot
- chasePlot
- radarDetectionGenerator
- visionDetectionGenerator

generateSyntheic

- radarClusterDetection
- trackRadarDetections
- dataPacking

birdsEyePlot

- coverageAreaPlotter
- plotCoverageArea
- detectionPlotter
- plotDetection
- laneBoundaryPlotter
- plotLaneBoundary
- trackPlotter
- plotTrack

Generate Synthetic Data from Driving Scenario

```
radarSensor = radarDetectionGenerator
visionSensor = visionDetectionGenerator
```

```
advance(s)
Or, move(v,time)
```

```
updatePlots(s)
```

```
generateDetections
```

```
updateBirdsEyePlot
```

```
targets = targetPoses(egoCar)
```

Get target poses (positions & orientations) w.r.t. ego vehicle

```
rDets = radarSensor(targets,time)
vDets = visionSensor(targets,time)
```

- Radar detection (MRR)
- Radar detection (LRR)
- Vision detection

```
radarClusterDetections(rDets)
```

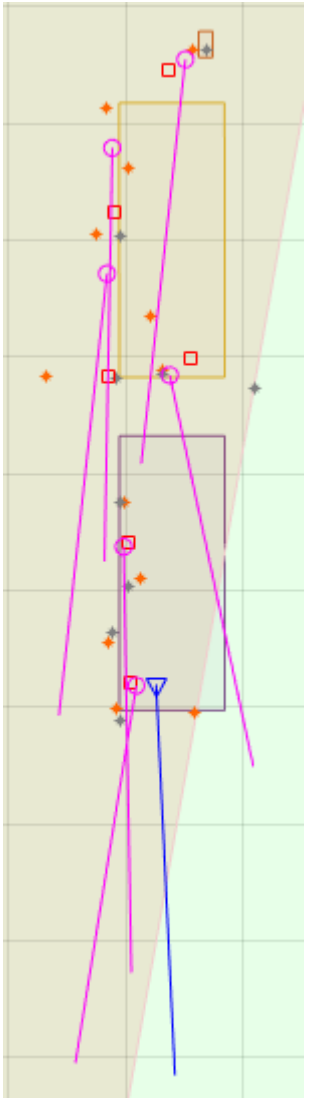
- Radar cluster

```
trackRadarDetections
```

- Radar tracks by multi-object tracker

```
dataPacking
```

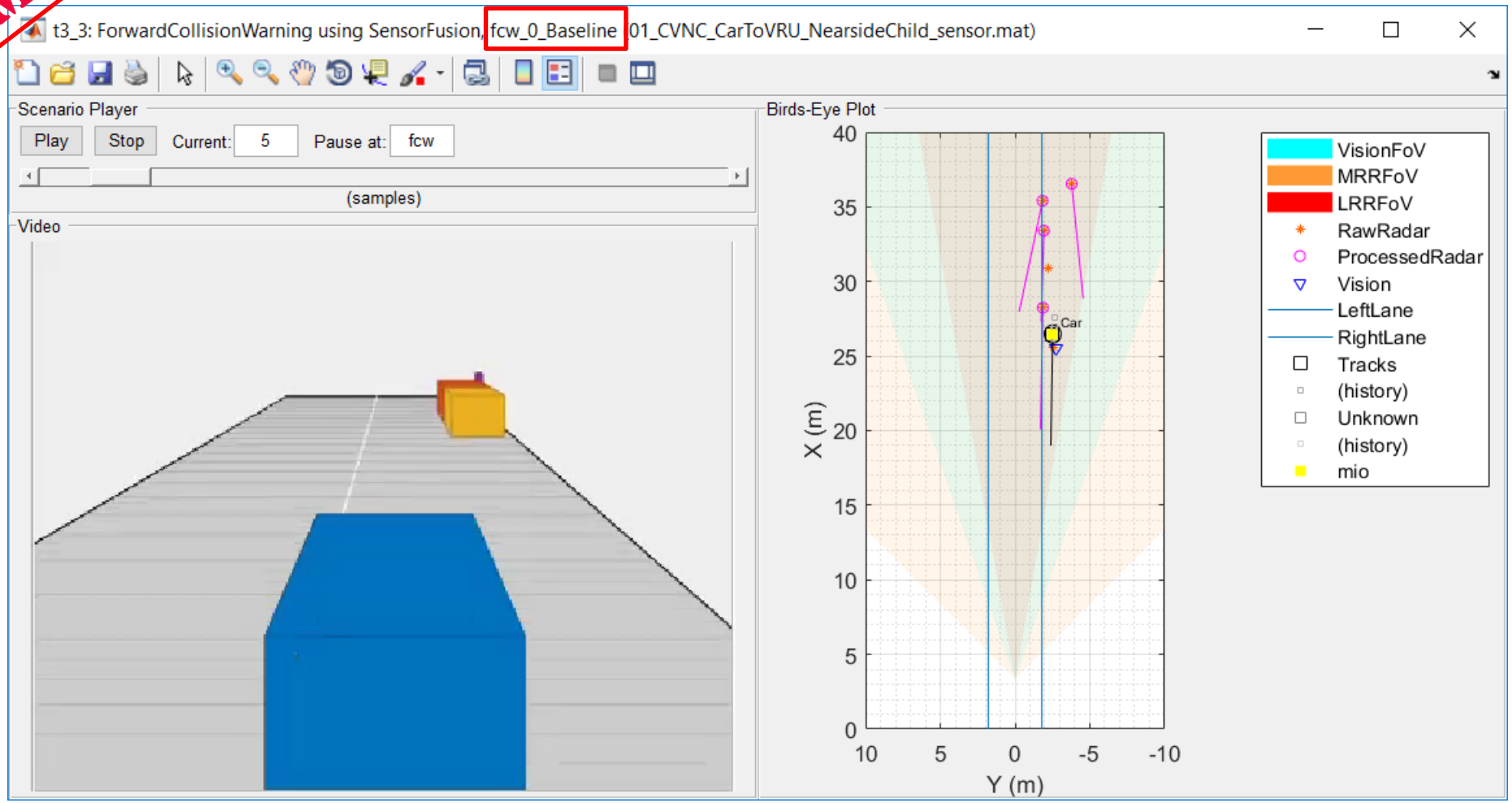
Pack the detections into a desired data format



Sensor Fusion with Synthetic Data (baseline algorithm)

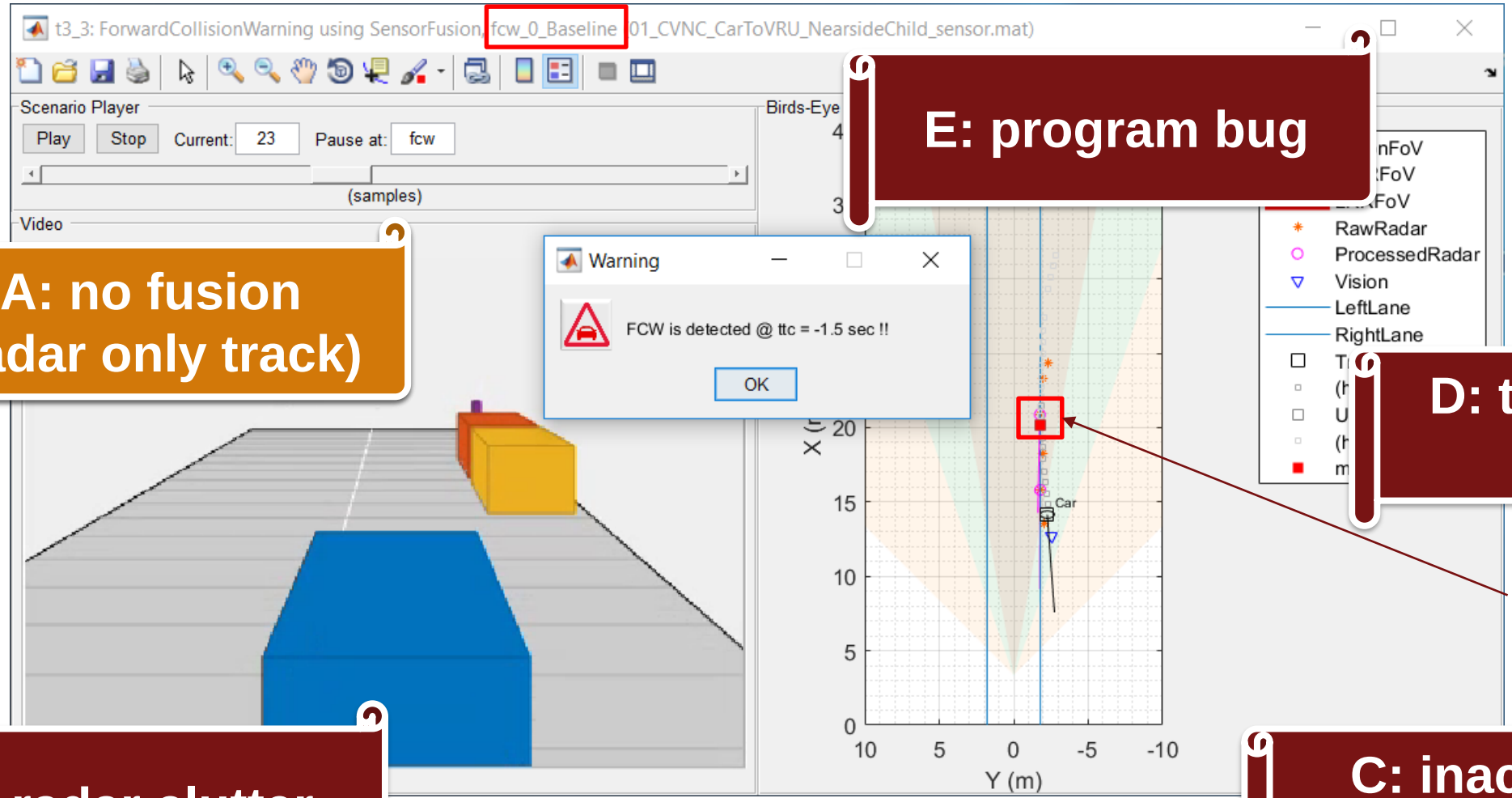
DEMO

t3_3_fcwSensorFusionWithSyntheticData.m



Pop Quiz

What causes the false positive?



A: no fusion (radar only track)

E: program bug

D: too sensitive warning

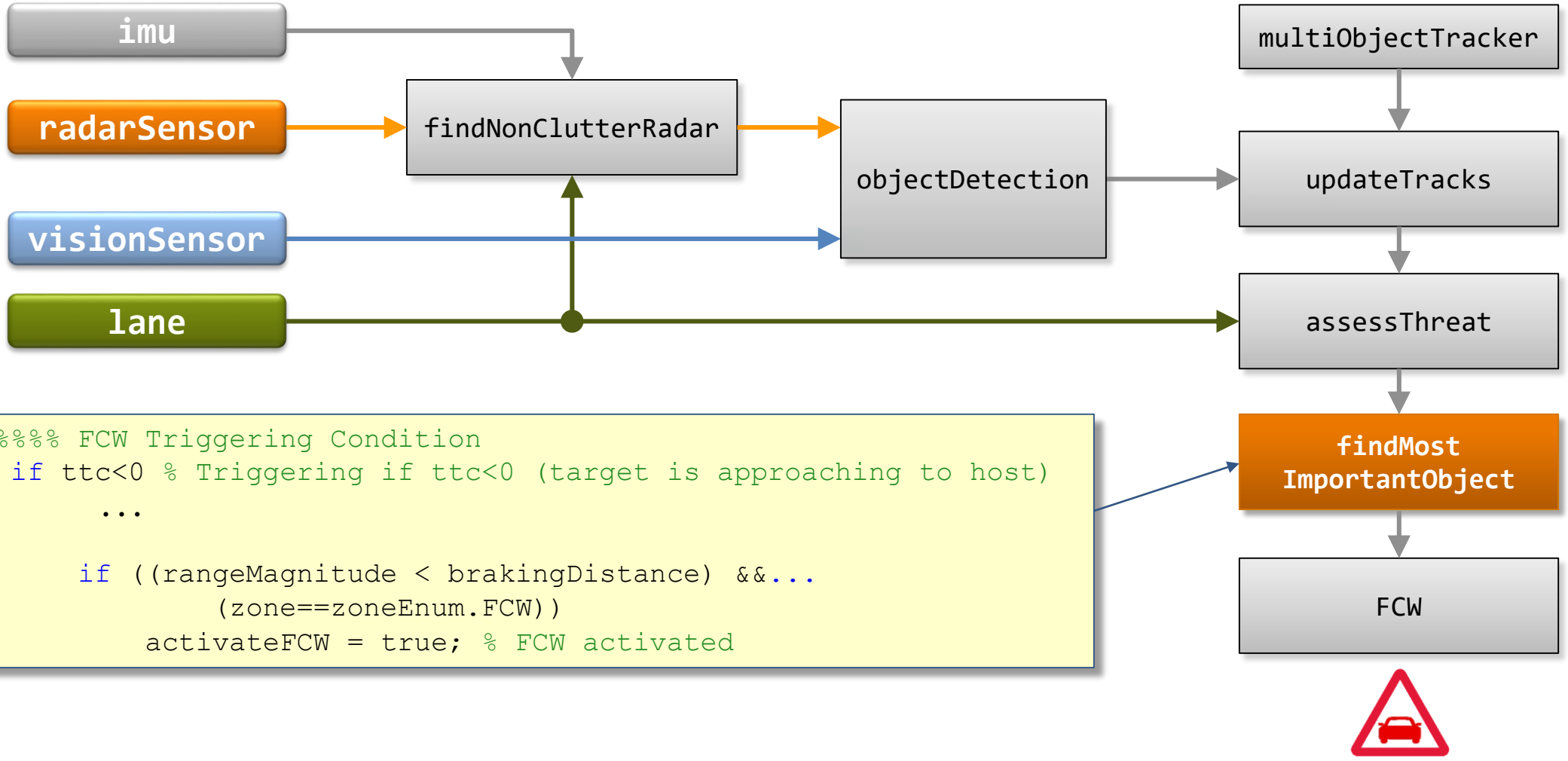
False Alarm by radar track

B: radar clutter

C: inaccurate tracking

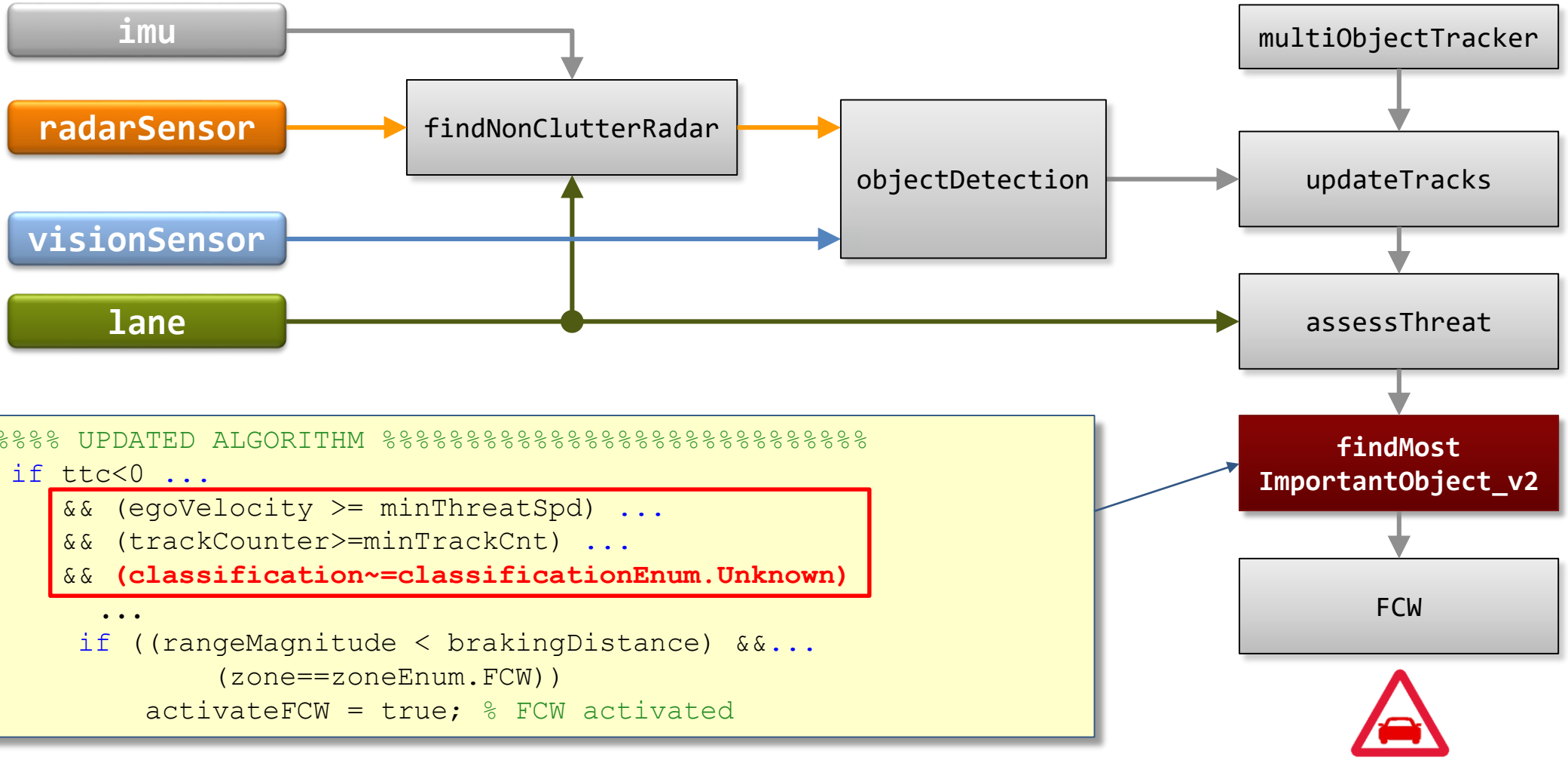
Baseline sensor fusion algorithm (fcw_0_Baseline)

A: no fusion (radar only track)



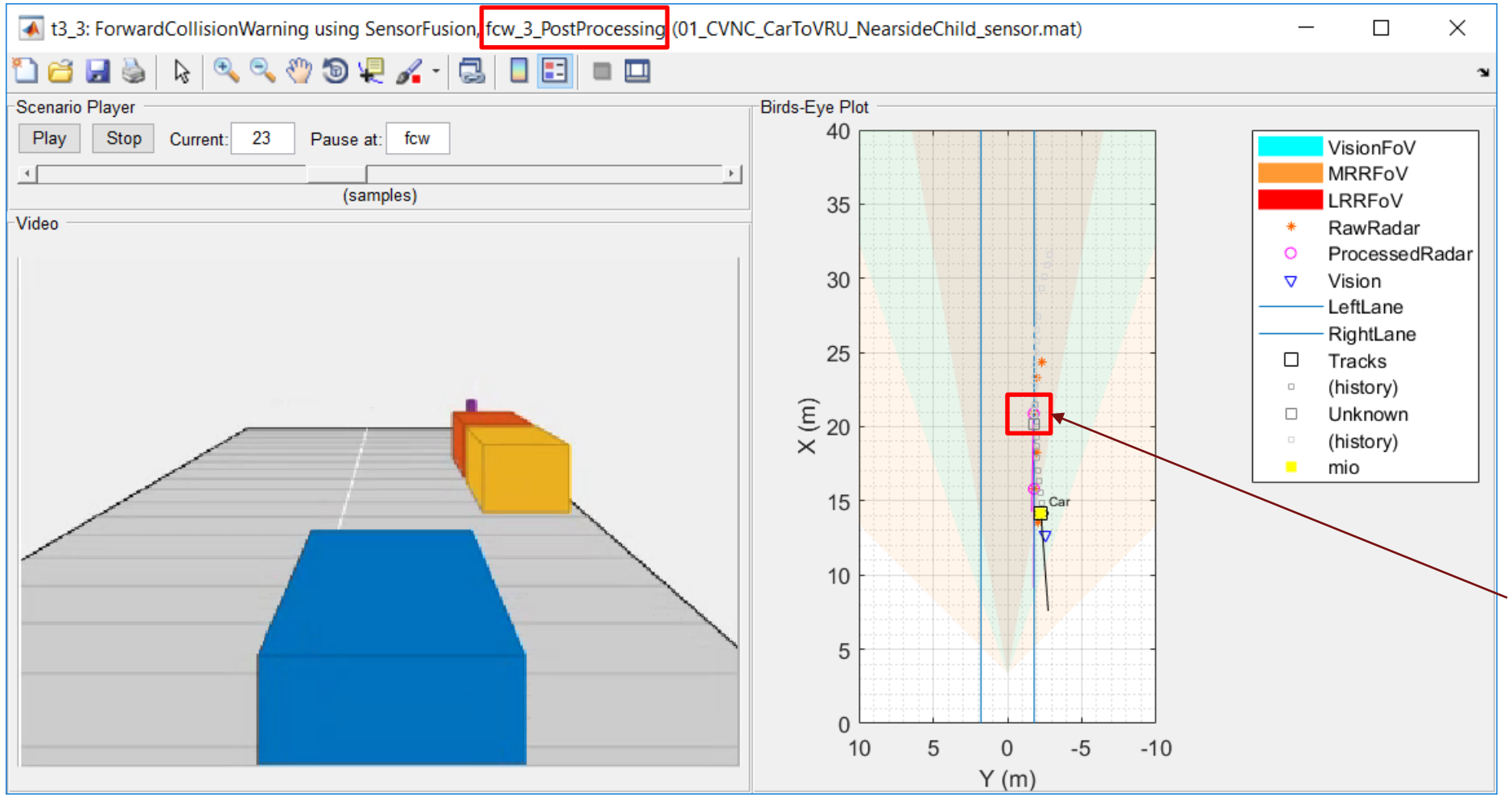

```
%%%%%% FCW Triggering Condition
   if ttc<0 % Triggering if ttc<0 (target is approaching to host)
        ...

       if ((rangeMagnitude < brakingDistance) &&...
             (zone==zoneEnum.FCW))
          activateFCW = true; % FCW activated
```

Modified algorithm (fcw_3_PostProcessing)



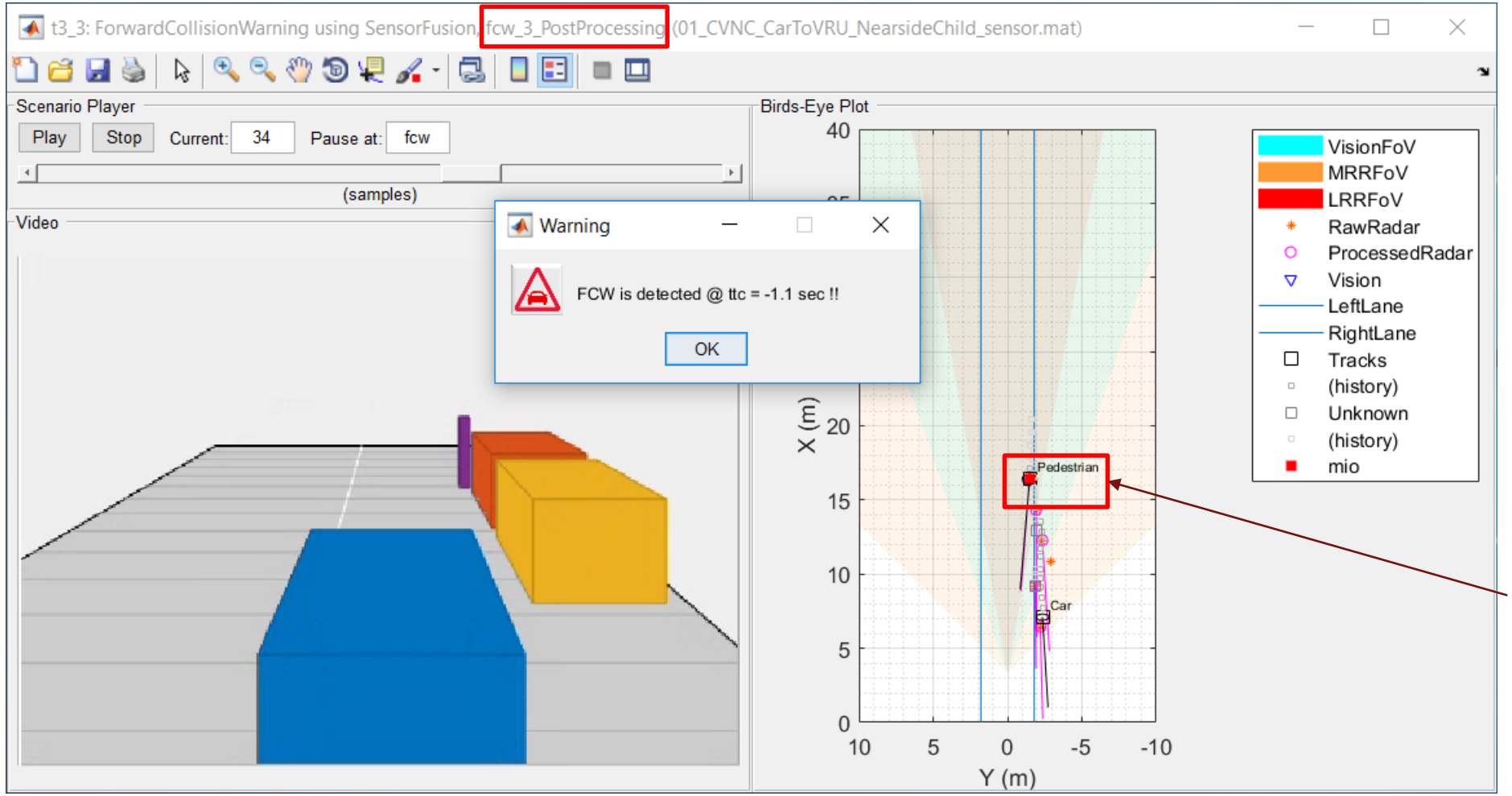

```
%%%%%% UPDATED ALGORITHM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   if ttc<0 ...
      && (egoVelocity >= minThreatSpd) ...
      && (trackCounter>=minTrackCnt) ...
      && (classification~=classificationEnum.Unknown)
         ...
        if ((rangeMagnitude < brakingDistance) &&...
                (zone==zoneEnum.FCW))
            activateFCW = true; % FCW activated
```

Sensor Fusion with Synthetic Data (*fcw_3_PostProcessing*)



track with unknown classification is disqualified for “mio”

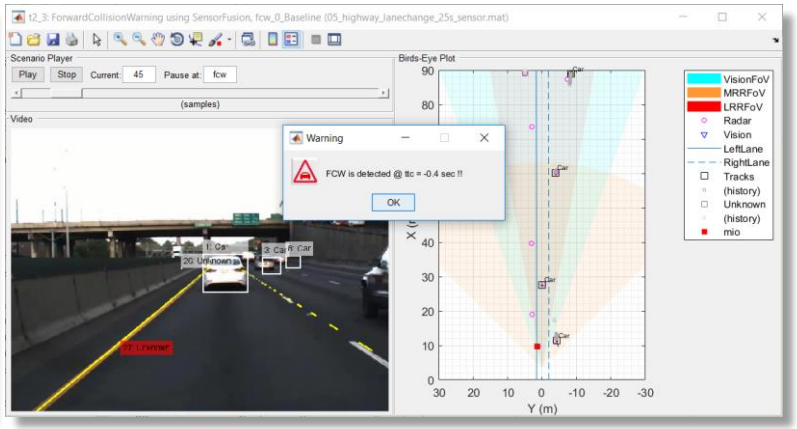
Sensor Fusion with Synthetic Data (*fcw_3_PostProcessing*)

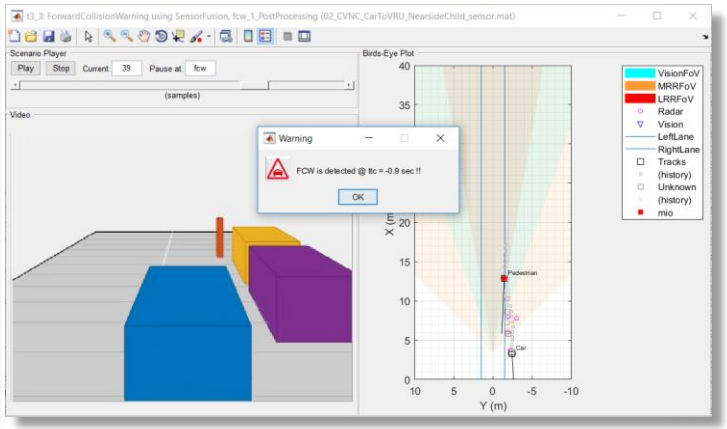


FCW detected by pedestrian detection

Automated Driving System Toolbox R2017a

Case study: Vision and radar-based sensor fusion





Explore a baseline sensor fusion algorithm

- Read logged vehicle data
- Visualize vehicle data
- Create multi-object tracker
- Implement forward collision warning

Test the algorithm with new data set

- Test the baseline algorithm with new data set
- Tune algorithm parameters
- Customize the algorithm

Synthesize data to further test the algorithm

- Create driving scenario
- Add sensor detections
- Generate synthetic data
- Test the algorithm