Automated Driving System Toolbox \textit{R2017a}

\textit{Case study:}

\textit{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

Video Display
- VideoReader
- readFrame
- imshow

Video Annotation
- parabolicLaneBoundary
- insertLaneBoundary
- vehicleToImage
- insertObjectAnnotation

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

multiObjectTracker
- objectDetection
- trackingEKF
- trackingUKF
- trackingKF
- updateTracks
Explore a baseline sensor fusion algorithm + ScenarioPlayer + PointCloud

**ScenarioPlayer**
- play/pause
- stop
- pause at
- slider

**Pause at FCW**

**Display LaneBoundaryType**

**Lidar Point Cloud**
- pcplayer
- pointCloud
- findPointsInROI
- findNeighborsInRadius
- select
- pcnoiise/pcdownsample
- pcfitplane

**t2_fcwSensorFusion.m**

Explore a baseline sensor fusion algorithm + ScenarioPlayer + PointCloud

**Display LaneBoundaryType**

**Lidar Point Cloud**
- pcplayer
- pointCloud
- findPointsInROI
- findNeighborsInRadius
- select
- pcnoiise/pcdownsample
- pcfitplane
Baseline sensor fusion algorithm

- imu
- radarSensor
- visionSensor
- lane

1. findNonClutterRadar
2. objectDetection
3. updateTracks
4. assessThreat
5. findMostImportantObject
6. FCW
**findNonClutterRadar**

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

**Enumeration**

- Safe (0)
- Warn (1)
- FCW (2)

**Diagram:**
- Ego vehicle speed
- Relative speed
- Position
- Object Zone
- Calculate Ground Speed
- Motion Type
- Moving or Stationary
- Find Non-Clutter Radar

**Diagram Elements:**
- X: Distance
- Y: Position
- Vision: Detection Area
- LRF: Range
- Radar: Signal Range
- Clutter: Obstacle Zones
- Safe/Warning/FCW Zones
- Ego Lane
- Non-clutter Area
Forward Collision Warning using Sensor Fusion

- imu
- radarSensor
- visionSensor
- lane

Process:
1. imu
2. radarSensor to findNonClutterRadar
3. visionSensor
4. lane
5. objectDetection
6. updateTracks
7. multiObjectTracker
8. assessThreat
9. findMostImportantObject
10. FCW

Flow:
- imu to findNonClutterRadar
- radarSensor to findNonClutterRadar
- visionSensor to objectDetection
- lane to objectDetection
- objectDetection to updateTracks
- multiObjectTracker to assessThreat
- assessThreat to findMostImportantObject
- findMostImportantObject to FCW

Warning Icon: 🔴
Multi-Object Tracker

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

- Update tracker with new detections
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) James Munkres's variant of the Hungarian assignment algorithm
Multi-Object Tracker

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

Track Manager

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

Kalman filter

New Detections

Updated Tracks
**Multi-Object Tracker**

- **Track Manager**
  - Assign detections to existing tracks
  - Initiate tracks
  - Update & confirm tracks
  - Predict & delete tracks

- **Kalman filter**
  - Initialize filter
  - Correct filter with new measurement
  - Update & confirm tracks
  - Predict & delete tracks

- **Function**

  ```
  function filter = initcaekf(detection)
  ...
  filter = trackingEKF(@constacc, ... % State-transition function (const-accel model)
  ...
  end
  ```

- **Tracker**

  ```
  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
  ```

- **Variables**

  - Time
  - Measurement
  - MeasurementNoise
  - SensorIndex
  - MeasurementParameters
  - ObjectClassID
  - ObjectAttributes

- **Updated Tracks**
**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.

![Graph 1](image1.png)

![Graph 2](image2.png)

* '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
What causes the false positive?

- **Pop Quiz**

- **A**: radar ghost detection
- **B**: radar clutter
- **C**: inaccurate tracking
- **D**: too sensitive warning
- **E**: program bug

**False Positive**
Baseline sensor fusion algorithm (fcw_0_Baseline)

**FCW Triggering Condition**

```plaintext
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)*

```plaintext
%%%%%% 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
Baseline sensor fusion algorithm *(fcw_0_Baseline)*

- **imu**
- **radarSensor**
- **visionSensor**
- **lane**

**if** Zone == zoneEnum.FCW || (Zone ~= zoneEnum.Safe && MotionType == objectMotionEnum.Moving)
Modified algorithm (**fcw_2_PreProcessing**)

- imu
- radarSensor
- visionSensor
- lane

```plaintext
if Zone ~= zoneEnum.Safe
```

- findNonClutterRadar_v1
- objectDetection
- multiObjectTracker
- updateTracks
- assessThreat
- findMostImportantObject
- FCW
fcw_0_Baseline vs. 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.
Create Driving Scenario

- chasePlot
- road
- vehicle
- actor
- path
- plot
- radarDetectionGenerator
- visionDetectionGenerator
- coverageAreaPlotter
- plotCoverageArea
- detectionPlotter
- plotDetection
- laneBoundaryPlotter
- plotLaneBoundary
- trackPlotter
- plotTrack

Egocentric projective perspective plot

Generate radar or vision detections for driving scenario
**Workflow for driving scenario**

1. **Create a driving scenario object**
   
   $s = \text{drivingScenario}$

2. **Add a road to driving scenario**
   
   $\text{road}(s, \text{RoadCenters, Width})$

3. **Create a vehicle or actor within driving scenario**
   
   $v = \text{vehicle}$
   $a = \text{actor}$

4. **Create actor or vehicle path in driving scenario**
   
   $\text{path}(v, \text{waypoints, speed})$
   $\text{path}(a, \text{waypoints, speed})$

5. **Create driving scenario plot and egocentric projective perspective plot**
   
   $\text{plot}(s)$
   $\text{chasePlot}(\text{egoCar})$

6. **Generate radar or vision detections for driving scenario**
   
   $\text{radarDetectionGenerator}$
   $\text{visionDetectionGenerator}$

7. **Advance driving scenario simulation by one time step**
   
   $\text{advance}(s)$
   Or, move($v, \text{time}$)

8. **Update scenario and chase plots**
   
   $\text{updatePlots}(s)$

9. **Generate sensor detections**
   
   $\text{generateDetections}$

10. **Update Birds-eye plot**
    
    $\text{updateBirdsEyePlot}$

11. **Advance or move actor to time**
    
    Or, move($v, \text{time}$)

12. **Generate radar or vision detections for driving scenario**
    
    $\text{radarDetectionGenerator}$
    $\text{visionDetectionGenerator}$
Sensor model for vision detection

```matlab
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', cameralntrinsics( ...  % 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

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

- **generateSyntheic**
  - radarClusterDetection
  - trackRadarDetections
  - dataPacking

- **drivingScenario**
  - road
  - vehicle
  - actor
  - actorProfiles
  - path
  - plot
  - chasePlot
  - radarDetectionGenerator
  - visionDetectionGenerator
Generate Synthetic Data from Driving Scenario

- **advance(s)**
  - Or, move(v, time)

- **updatePlots(s)**

- **generateDetections**

- **updateBirdsEyePlot**

- **targets = targetPoses(egoCar)**

- **rDets = radarSensor(targets, time)**
  - **visionSensor = visionDetectionGenerator**
  - **vDets = visionSensor(targets, time)**

- **radarClusterDetections(rDets)**

- **trackRadarDetections**

- **dataPacking**

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

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

- Radar cluster

- Radar tracks by multi-object tracker

- Pack the detections into a desired data format
Sensor Fusion with Synthetic Data (baseline algorithm)
What causes the false positive?

Pop Quiz

A: no fusion (radar only track)
B: radar clutter
C: inaccurate tracking
D: too sensitive warning
E: program bug

False Alarm by radar track
Baseline sensor fusion algorithm (fcw_0_Baseline)

```matlab
%%%%%% 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)

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

---

### Updated Algorithm

**Input Sensors:**
- **imu**
- **radarSensor**
- **visionSensor**
- **lane**

**Processing Flow:**
1. **findNonClutterRadar**
2. **objectDetection**
3. **updateTracks**
4. **assessThreat**
5. **findMostImportantObject_v2**
6. **FCW**
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