MATLAB EXPO 2017
KOREA
4월 27일, 서울
등록 하기 matlabexpo.co.kr
Automated Driving System Toolbox 소개

이제훈 차장
Common Questions from Automated Driving Engineers

How can I Visualize Sensor data?

How can I design and verify Perception algorithms?

How can I design and verify Sensor fusion?
Common Questions from Automated Driving Engineers

How can I visualize sensor data?

How can I design and verify perception algorithms?

How can I design and verify sensor fusion?
Automated Driving **Sensor data**

- **Camera**
- **Radar**
- **Lidar**
- **IMU**
- **Object Detection**
- **Sensor fusion & Tracking**

*IMU: Inertial Measurement Unit*
Automated Driving Sensor data

Camera (640 x 480 x 3)

<table>
<thead>
<tr>
<th>SensorID</th>
<th>Timestamp</th>
<th>NumDetections</th>
<th>Detections(1)</th>
<th>Detections(2)</th>
<th>Detections(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1461634696379742</td>
<td>6</td>
<td>TrackID: 0, Classification: 5, Position: [22.61 -0.43 2.24], Velocity: [-9.86 0 0], Size: [0 1.75 0]</td>
<td>TrackID: 1, Classification: 5, Position: [22.8 3.12 2.24], Velocity: [-9.37 0 0], Size: [0 1.8 0]</td>
<td>TrackID: 12, Classification: 5, Position: [57.69 3.13 0.34], Size: [0 1.75 0]</td>
</tr>
</tbody>
</table>

Vision Detector

<table>
<thead>
<tr>
<th>SensorID</th>
<th>Timestamp</th>
<th>NumDetections</th>
<th>Detection(1)</th>
<th>Detection(2)</th>
<th>Detection(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1461634696379742</td>
<td>6</td>
<td>TrackID: 0, Classification: 5, Position: [22.61 -0.43 2.24], Velocity: [-9.86 0 0], Size: [0 1.75 0]</td>
<td>TrackID: 1, Classification: 5, Position: [22.8 3.12 2.24], Velocity: [-9.37 0 0], Size: [0 1.8 0]</td>
<td>TrackID: 12, Classification: 5, Position: [57.69 3.13 0.34], Size: [0 1.75 0]</td>
</tr>
</tbody>
</table>

Radar Detector

<table>
<thead>
<tr>
<th>SensorID</th>
<th>Timestamp</th>
<th>NumDetections</th>
<th>Detections(1)</th>
<th>Detections(2)</th>
<th>Detections(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1461634696407521</td>
<td>23</td>
<td>TrackID: 0, TrackStatus: 6, Position: [56.07 17.73 0.34], Velocity: [-8.5 2.86 0], Amplitude: 3</td>
<td>TrackID: 1, TrackStatus: 6, Position: [35.35 19.59 0.34], Velocity: [-8.02 4.92 0], Amplitude: 3</td>
<td>TrackID: 12, TrackStatus: 5, Position: [57.69 3.13 0.34], Size: [0 1.75 0]</td>
</tr>
</tbody>
</table>

Lidar (47197 x 3)

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>-12.2911</td>
<td>1.4790</td>
<td>-0.5900</td>
</tr>
<tr>
<td>-14.8852</td>
<td>1.7755</td>
<td>-0.6478</td>
</tr>
<tr>
<td>-18.8020</td>
<td>2.2231</td>
<td>-0.7403</td>
</tr>
<tr>
<td>-25.7033</td>
<td>3.0119</td>
<td>-0.9246</td>
</tr>
<tr>
<td>-0.0632</td>
<td>0.0815</td>
<td>1.2501</td>
</tr>
<tr>
<td>-0.0978</td>
<td>0.0855</td>
<td>1.2561</td>
</tr>
<tr>
<td>-0.2814</td>
<td>0.1064</td>
<td>1.2575</td>
</tr>
<tr>
<td>-0.3375</td>
<td>0.1129</td>
<td>1.2650</td>
</tr>
</tbody>
</table>

Inertial Measurement Unit

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Velocity</th>
<th>YawRate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1461634696379742</td>
<td>9.2795</td>
<td>0.0040</td>
</tr>
</tbody>
</table>
Visualize sensor data
Visualize **Sensor data** in vehicle coordinates

- ISO 8855 vehicle axis coordinate system
  - Positive x is forward
  - Positive y is left

```matlab
%% Plot in vehicle coordinates
ax2 = axes(...
    'Position',[0.6 0.12 0.4 0.85]);
bep = birdsEyePlot(...
    'Parent',ax2,...
    'Xlimits',[0 45],...
    'Ylimits',[-10 10]);
legend('off');
```
Visualize **Sensor data** - expected coverage area

```matlab
%% Create coverage area plotter
covPlot = coverageAreaPlotter(bep,...
   'FaceColor','blue',...
   'EdgeColor','blue');

%% Update coverage area plotter
plotCoverageArea(covPlot,...
   [sensorParams(1).X ... % Position x
    sensorParams(1).Y],... % Position y
   sensorParams(1).Range,...
   sensorParams(1).YawAngle,...
   sensorParams(1).FoV(1)) % Field of view
```

Plot sensor coverage area with `coverageAreaPlotter`
Visualize **Sensor data** - detected objects (vehicle coordinates)

```matlab
%% Create detection plotter
detPlot = detectionPlotter(bep, ...
    'MarkerEdgeColor','blue',...
    'Marker','^');

%% Update detection plotter
n = round(currentTime/0.05);
numDets = vision(n).numObjects;
pos = zeros(numDets,3);
vel = zeros(numDets,3);
labels = repmat({''},numDets,1);
for k = 1:numDets
    pos(k,:) = vision(n).object(k).position;
    vel(k,:) = vision(n).object(k).velocity;
    labels{k} = num2str(...
        vision(n).object(k).classification);
end
plotDetection(detPlot,pos,vel,labels);
```

detectionPlotter can be used to visualize
vision detector, radar detector, and
lidar point cloud
Visualize **Sensor data** - detected objects (image coordinates)

```matlab
%% Bounding box positions in image coordinates
imBoxes = zeros(numDets,4);
for k = 1:numDets
    if vision(n).object(k).classification == 5
        vehPosLR = vision(n).object(k).position(1:2)';
        imPosLR = vehicleToImage(sensor, vehPosLR);
        boxHeight = 1.4 * 1333 / vehPosLR(1);
        boxWidth = 1.8 * 1333 / vehPosLR(1);
        imBoxes(k,:)= [imPosLR(1) - boxWidth/2, ...
                        imPosLR(2) - boxHeight, ...
                        boxWidth, boxHeight];
    end
end

%% Draw bounding boxes on image frame
frame = insertObjectAnnotation(frame, ...'
    'Rectangle', imBoxes, labels, ...'
    'Color','yellow', 'LineWidth',2);
im.CData = frame;
```
Learn more about visualizing vehicle data by exploring examples in the Automated Driving System Toolbox R2017a

- Plot object detectors in vehicle coordinates
  - Vision & radar detector
  - Lane detectors
  - Detector coverage areas

- Transform between vehicle and image coordinates

- Plot lidar point cloud
Common Questions from Automated Driving Engineers

How can I Visualize Sensor data?

How can I design and verify Perception algorithms?

How can I design and verify Sensor fusion?
Automated Driving Perception Algorithms

Object Detection:
Locate and classify object in image

Pedestrian Detection

Vehicle Detection
MATLAB Tools to Train Detectors

```
imageDS = imageDatastore(dir)
```

Easily manage large sets of images
- Single line of code to access images
- Operates on disk, database, big-data file system
MATLAB Tools to **Train** Detectors

Images → **Label Ground Truth** → Ground Truth → **Train detector** → **Object detector**

**Label ground truth**

Automate Labeling of Ground Truth
MATLAB Tools to Train Detectors

Design object detectors with the Computer Vision System Toolbox

<table>
<thead>
<tr>
<th>Machine Learning</th>
<th>Aggregate Channel Feature</th>
<th>trainACFObjectDetector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cascade</td>
<td>trainCascadeObjectDetector</td>
</tr>
<tr>
<td>Deep Learning</td>
<td>R-CNN (Regions with Convolutional Neural Networks)</td>
<td>trainRCNNObjectDetector</td>
</tr>
<tr>
<td></td>
<td>Fast R-CNN</td>
<td>trainFastRCNNObjectDetector</td>
</tr>
<tr>
<td></td>
<td>Faster R-CNN</td>
<td>trainFasterRCNNObjectDetector</td>
</tr>
</tbody>
</table>
Designing **Perception Algorithms**

*Computer Vision Algorithms for Automated Driving*

---

**Vehicle Detection**
Deep learning and ACF based (pre-trained)

**Pedestrian Detection**
ACF and HOG/SVM based (pre-trained)
Designing **Perception Algorithms**

*Additional Computer Vision Algorithms for Automated Driving*

- **Vehicle detection** with distance estimation using mono-camera
- **Lane Detection and Classification**
  - RANSAC-based lane boundary fitting
  - Lane boundary visualization
Designing **Perception Algorithms**

**LiDAR Processing Algorithms**
Example of Vision System Detection

How can I verify this detection is correct?
Ground truth labeling to **Train** Detectors

Images → Label Ground Truth → Ground Truth → Train detector → Object detector

Ground truth labeling to **Evaluate** Detectors

Images → Label Ground Truth → Ground Truth → Detections → Evaluate detections
Evaluate detections against ground truth
Learn more about verifying perception algorithms by exploring examples in the Automated Driving System Toolbox R2017a.

- **Train object detector** using deep learning and machine learning techniques
- **Label detections** with Ground Truth Labeler App
- **Extend connectivity** of Ground Truth Labeler App

---

**Train a Deep Learning Vehicle Detector**

**Define Ground Truth Data for Video or Image Sequences**

**Connect Lidar Display to Ground Truth Labeler**
Common Questions from Automated Driving Engineers

How can I visualize sensor data?

How can I design and verify perception algorithms?

How can I design and verify sensor fusion?
Automated Driving **Sensor fusion** with radar and vision

Can we fuse detections to better track the vehicle?
Design multi-object tracker
Sensor fusion framework

- 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

Object Detections

Sensor fusion Framework

Track Manager

Tracking Filter

Tracks

Time
Measurement
Measurement Noise

Time
State
State Covariance
Track ID
Age
Is Confirmed
Is Coasted
Sensor fusion - Data Association

Pairs of visions and associated radars

Assignments
- \( V_1 + R_2 \)
- \( V_2 + R_1 \)
- ...  
- \( V_n + R_m \)

Fusion
- \( f(V_1) + f(R_2) \)
- \( f(V_2) + f(R_1) \)
- ...  
- \( f(V_n) + f(R_m) \)

Fused Object List

[assignments, unassignedVisions, unassignedRadars] = ...
assignDetectionsToTracks(costMatrix, param.costOfNonAssignment);
Sensor fusion - Kalman Filter

Initial state & covariance

Previous state & covariance

Time Update ("Predict")

(1) Predict state based on physical model and previous state

\[
\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k + w_k
\]

(2) Predict error covariance matrix

\[
P_k^- = AP_{k-1}A^T + Q
\]

Measurement Update ("Correct")

(1) Compute Kalman gain

\[
K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}
\]

(2) Update estimate state with measurement

\[
\hat{\hat{x}}_k = \hat{x}_k^- + K_k (z_k - H\hat{x}_k^-)
\]

(3) Update the error covariance matrix

\[
P_k = (I - K_k H) P_k^-
\]

Output of updated state

\(k \rightarrow k-1\)

Current becomes previous

\(A\) : State matrix relates the state at the previous, \(k-1\) to the state at the current, \(k\)

\(u\) : Control variable matrix

\(w\) : Process (state) noise

\(P_k^- = E[e_k^- e_k^-^T]\) : Process (state) covariance matrix

\(e_k^- = x_k - \hat{x}_k^-\) : Process noise covariance matrix

\(Q = E[ww^T]\) : Process noise covariance matrix

\(K\) : Kalman gain

\(R\) : Sensor noise covariance matrix (measurement error)

\(u\) : Control variable matrix

\(w\) : Process (state) noise

\(v\) : Measurement noise

\(V\) : Measurement noise

\(H\) : Output matrix relates the state to the measurement

\(z_k = Hx_k + v_k\)

\(z_k\) : Measurement

\(z_k\) : Measurement

\(\hat{\hat{x}}_k\) : Output of updated state

\(\hat{x}_0\) : Initial state

\(P_0\) : Initial covariance

\(P_{k-1}\) : Previous covariance

\(P_k\) : Current covariance

\(\hat{x}_k\) : Output of updated state

\(\hat{x}_0\) : Initial state
Sensor fusion - Kalman Filter

**Time Update ("Predict")**

\[
[z_{\text{pred}}, x_{\text{pred}}, P_{\text{pred}}] = \text{predict}(\text{obj})
\]

- \(z_{\text{pred}}\): prediction of measurement
- \(x_{\text{pred}}\): prediction of state
- \(P_{\text{pred}}\): state estimation error covariance at the next time step

**Measurement Update ("Correct")**

\[
[z_{\text{corr}}, x_{\text{corr}}, P_{\text{corr}}] = \text{correct}(\text{obj}, z)
\]

- \(z_{\text{corr}}\): correction of measurement
- \(x_{\text{corr}}\): correction of state
- \(P_{\text{corr}}\): state estimation error covariance

**Output of updated state**

- \(x_{\text{corr}}\)
- \(P_{\text{corr}}\)

**Linear KF (trackingKF)**
- initcvkf

**Extended KF (trackingEKF)**
- initcvekf
- initcvukf

**Unscented KF (trackingUKF)**
- initcaekf
- initcaukf

**Constant velocity**
- initcvkf
- initcvekf
- initcvukf

**Constant acceleration**
- initcakf
- initcaekf
- initcaukf

**Constant turn**
- Not applicable
- initctekf
- initctukf
Synthesize Driving Scenario for Sensor fusion

Simulated data for worst-case scenarios

- OEM specific test scenarios
- Scenarios identified from real world test drive data
- Fail Operation test scenarios
%% Create a new scenario
s = drivingScenario('SampleTime', 0.05);

%% Create road
road(s, [ 0  0; ... % Centers [x,y] (m)
     45  0],... 
     5); % Width (m)
road(s, [35  20; ...
     35 -10],...
     5);

%% Plot scenario
p1 = uipanel('Position',[0.5 0 0.5 1]);
a1 = axes('Parent',p1);
plot(s,'Parent',a1,...
     'Centerline','on','Waypoints','on')
a1.XLim = [0 45];
a1.YLim = [-6 20];
Specify ego vehicle path using waypoints and speeds

%% Add ego vehicle
egoCar = vehicle(s);
waypoints = [ 2  -1.25; ... % [x y] (m)
              28  -1.25; ... 
              30  -1.25; ... 
              36.25  4; ... 
              36.25  6; ... 
              36.25  14];
speed = 13.89; % (m/s) = 50 km/hr
path(egoCar, waypoints, speed);

%% Play scenario
while advance(s)
    pause(s.SampleTime);
end
%% Add child pedestrian actor
child = actor(s,'Length',0.24,...
       'Width',0.45,...
       'Height',1.7,...
       'Position',[40 -5 0],...
       'Yaw',180);

path(child,...
     [30 15; 40 15],... % Waypoints (m)
     1.39); % Speed (m/s) = 5 km/hr

%% Add Target vehicle
targetVehicle = vehicle(s);
path(targetVehicle,...
     [44 1; -4 1],... % Waypoints (m)
     [5 ; 14]); % Speeds (m/s)
Synthesize Driving Scenario for Sensor fusion

radarSensor =

radarDetectionGenerator with properties:

SensorIndex: 1
UpdateInterval: 0.1000

SensorLocation: [3.4000 0]
Height: 0.2000
Yaw: 0
Pitch: 0
Roll: 0

FieldOfView: [20 5]
MaxRange: 150
RangeRateLimits: [-100 100]

DetectionProbability: 0.9000
FalseAlarmRate: 1.0000e-06

Show all properties

visionSensor =

visionDetectionGenerator with properties:

SensorIndex: 1
UpdateInterval: 0.1000

SensorLocation: [1.9000 0]
Height: 1.1000
Yaw: 0
Pitch: 1
Roll: 0

Intrinsics: [1×1 cameraIntrinsics]

FieldOfView: [43.6028 33.3985]
MaxRange: 150
MaxSpeed: 50
MaxAllowedOcclusion: 0.5000
MinObjectImageSize: [15 15]

DetectionProbability: 0.9000
FalsePositivesPerImage: 0.1000

Show all properties
Euro NCAP TEST PROTOCOL – AEB VRU systems

Car-to-VRU Nearside Child (CVNC)

- Vehicle
  - Travels towards a VRU
  - VRU
  - A child pedestrian crossing the road
  - Scenario
    - Pedestrian’s path running from behind and obstruction
    - Vehicle strikes the pedestrian at 50% of the vehicle’s width when no braking action is applied.

```matlab
%% Create a new scenario
s = drivingScenario;
s.SampleTime = 0.05;

%% Create road
RoadCenters = [0 0; 50 0];
road(s, RoadCenters, 10);

%% Add actors
egoCar = vehicle(s,'Position',[0.4 -1 0],'Yaw',180);
Waypoints = [0.4 -1; 36 -1]; % in meters
Speed = 13.89; % egoCar speed = 13.89 m/s = 50 km/hr
path(egoCar, Waypoints, Speed); % create egoCar path

%% --- two stationary cars
vehicle(s,'Position',[35.3 -3.8 0]);
vehicle(s,'Position',[29.6 -3.8 0]);

%% --- child pedestrian crossing it’s path running from behind of stationary cars
child = actor(s,'Length',0.24,'Width',0.45,'Height',1.7,...
'Position',[40 -5 0],'Yaw',190);
Waypoints = [40 -5; 40 190]; % in meters
Speed = 1.39; % child speed = 1.39 m/s = 5 km/hr
path(child, Waypoints, Speed); % create child path
```

C scenario, Running Child from Nearside from Obstruction vehicles (see Annex B)
Learn more about sensor fusion by exploring examples in the Automated Driving System Toolbox R2017a

- Design multi-object tracker based on logged vehicle data
- Generate C/C++ code from algorithm which includes a multi-object tracker
- Synthesize driving scenario to test multi-object tracker
Common Questions from Automated Driving Engineers

How can I visualize sensor data?

How can I design and verify perception algorithms?

How can I design and verify sensor fusion?
Thank you