What’s new in MATLAB® and Simulink® for Automated Driving

Mark Corless
Automated Driving Segment Manager
Industry Marketing
2018-05-02
How can you use MATLAB and Simulink to develop automated driving algorithms?

- Perception
- Control
- Planning
Examples of how you can use MATLAB and Simulink to develop automated driving algorithms

- Perception
- Control
- Planning

Deep learning

Sensor models & model predictive control

Sensor fusion with live data

Path planning
How can you use MATLAB and Simulink to develop perception algorithms?

- Deep learning
- Sensor fusion with live data
- Perception

Control

Sensor models & model predictive control

Planning

Path planning
Automated Driving System Toolbox introduced:
Ground Truth Labeling App to label video data
Automate labeling lanes with Ground Truth Labeler
Specify sublabels and attributes in Ground Truth Labeler App
Automate labeling pixels with Ground Truth Labeler
Learn how to train a deep learning network using this example

- Train free space detection network using deep learning
  
  *Computer Vision System Toolbox*™
Load and overlay pixel labels

% Load pixel labels
classes = ['Sky'; 'Building'; ...
    'Pole'; 'Road'; 'Pavement'; 'Tree'; ...
    'SignSymbol'; 'Fence'; 'Car'; ...
    'Pedestrian'; 'Bicyclist'];
pxds = pixelLabelDatastore(...
    labelDir,classes,labelIDs);

% Display labeled image
C = readimage(pxds, 1);
cmap = camvidColorMap;
B = labeloverlay(I,C,'ColorMap',cmap);
imshow(B)

pixelLabelDatastore manages large collections of pixel labels
Visualize distribution of labeled pixels

% Visualize label count by class
tbl = countEachLabel(pxls)

frequency = tbl.PixelCount / ... 
            sum(tbl.PixelCount);

bar(1:numel(classes),frequency)
xticks(1:numel(classes))
xticklabels(tbl.Name)
xtickangle(45)
ylabel('Frequency')

Labeled pixels in this set are imbalanced
Add weighted layer to compensate for imbalanced data set

% Create weighted layer
pxLayer = pixelClassificationLayer(...
    'Name','weightedLabels', 'ClassNames',tbl.Name,...
    'ClassWeights',classWeights)

% Replace layer
lgraph = removeLayers(lgraph, 'pixelLabels');
lgraph = addLayers(lgraph, pxLayer);
lgraph = connectLayers(lgraph,...
    'softmax', 'weightedLabels');

% Display network structure
plot(lgraph); ylim([0 9.5])
title('Replaced Layers Graph')
Train network and view progress

\[
[\text{net, info}] = \text{trainNetwork}(\text{datasource, lgraph, options});
\]
Assess similarity using intersection-over-union (IoU) metric

\[
iou = \text{jaccard}(\text{actual}, \ldots, \text{expected});
\]

table(classes, iou)

ans =

11×2 table

<table>
<thead>
<tr>
<th>classes</th>
<th>iou</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Sky&quot;</td>
<td>0.92659</td>
</tr>
<tr>
<td>&quot;Building&quot;</td>
<td>0.7987</td>
</tr>
<tr>
<td>&quot;Pole&quot;</td>
<td>0.16978</td>
</tr>
<tr>
<td>&quot;Road&quot;</td>
<td>0.95177</td>
</tr>
<tr>
<td>&quot;Pavement&quot;</td>
<td>0.41877</td>
</tr>
<tr>
<td>&quot;Tree&quot;</td>
<td>0.43401</td>
</tr>
<tr>
<td>&quot;SignSymbol&quot;</td>
<td>0.32509</td>
</tr>
<tr>
<td>&quot;Fence&quot;</td>
<td>0.492</td>
</tr>
<tr>
<td>&quot;Car&quot;</td>
<td>0.068756</td>
</tr>
<tr>
<td>&quot;Pedestrian&quot;</td>
<td>0</td>
</tr>
<tr>
<td>&quot;Bicyclist&quot;</td>
<td>0</td>
</tr>
</tbody>
</table>

Difference

![Image of actual and expected images with difference highlighted]
Distribution of labels in data affects intersection-over-union (IoU)

Distribution of labels in original data set

Evaluation metrics of network

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>IoU</th>
<th>MeanBFScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sky</td>
<td>0.93544</td>
<td>0.89279</td>
<td>0.88239</td>
</tr>
<tr>
<td>Building</td>
<td>0.79978</td>
<td>0.75543</td>
<td>0.59861</td>
</tr>
<tr>
<td>Pole</td>
<td>0.73166</td>
<td>0.18361</td>
<td>0.51426</td>
</tr>
<tr>
<td>Road</td>
<td>0.93644</td>
<td>0.90663</td>
<td>0.7086</td>
</tr>
<tr>
<td>Pavement</td>
<td>0.90624</td>
<td>0.72932</td>
<td>0.70585</td>
</tr>
<tr>
<td>Tree</td>
<td>0.86587</td>
<td>0.73694</td>
<td>0.67097</td>
</tr>
<tr>
<td>SignSymbol</td>
<td>0.76118</td>
<td>0.35339</td>
<td>0.44175</td>
</tr>
<tr>
<td>Fence</td>
<td>0.83258</td>
<td>0.49648</td>
<td>0.50265</td>
</tr>
<tr>
<td>Car</td>
<td>0.90961</td>
<td>0.75263</td>
<td>0.64837</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>0.83751</td>
<td>0.35409</td>
<td>0.46796</td>
</tr>
<tr>
<td>Bicyclist</td>
<td>0.84156</td>
<td>0.5472</td>
<td>0.46933</td>
</tr>
</tbody>
</table>

Underrepresented classes such as Pedestrian and Bicyclist are not segmented as well as classes such as Sky and Road.
Detection drivable space using semantic segmentation
Learn more about developing deep learning perception algorithms with these examples

- **Add semantic segmentation automation algorithm to Ground Truth Labeler App Automated Driving System Toolbox™**
- **Train free space detection network using deep learning**
  *Computer Vision System Toolbox™*
- **Generate CUDA® code to execute directed acyclic graph network on an NVIDIA GPU**
  *GPU Coder™*
How can you use MATLAB and Simulink to develop perception algorithms?

Perception

Deep learning

Sensor fusion with live data

Control

Sensor models & model predictive control

Planning

Path planning
Automated Driving System Toolbox introduced: Multi-object tracker to develop sensor fusion algorithms

Multi-Object Tracker

- 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
How can I test my sensor fusion algorithm with live data?
Test forward collision warning algorithm with live data from vehicle

FCW application
- Read sensor data stream and video stream
  - Vision Object
  - Radar Object
  - Lane
  - Vehicle Speed
  - Yaw Rate
  - Video frame

FCW algorithm

Visualization
Test forward collision warning algorithm with live data from “surrogate” vehicle
Send live CAN FD and TCP/IP data
Receive live CAN FD and TCP/IP data

Video received over TCP/IP

Detections received over CAN FD
Generate C/C++ code for algorithm
Stream live CAN FD and TCP/IP data into compiled algorithm code

Algorithm uses 1’s of msec in software-in-the-loop mode
Learn more about developing sensor fusion algorithms

- **Design** algorithm with multi-object tracker and recorded vehicle data
  Automated Driving System Toolbox™

- **Generate C/C++ code from algorithm** which includes a multi-object tracker
  MATLAB Coder™

- **Stream CAN FD data to prototype algorithm on your laptop**
  Vehicle Network Toolbox™
How can you use MATLAB and Simulink to develop control algorithms?

- Deep learning
- Sensor fusion with live data
- Perception
- Control
- Planning

Sensor models & model predictive control

Path planning
Automated Driving System Toolbox introduced:
Synthesizing scenarios to test sensor fusion algorithms
Simulate closed loop system with radar/vision detections, sensor fusion, and model-predictive control
Synthesize detections to test sensor fusion and model-predictive controller
Synthesize lane detection with Vision Detection Generator

Vision Detection Generator

Sensor simulation block used to generate vision detections from simulated actor poses. Detections are generated at intervals of the sensor's update interval. A statistical model generates measurement noise, true detections, and false positives. The random numbers used by the statistical model are controlled by the random number generator settings on the Measurements tab.

Parameters

- **Unique Identifier of sensor:**
  - Unique Identifier of sensor: 1

- **Types of detections generated by sensor:**
  - Types of detections: Lanes and objects

- **Required interval between sensor updates (s):**
  - Required interval between sensor updates: 1.0

- **Required interval between lane detection updates (s):**
  - Required interval between lane detection updates: Lanes with occlusion

Sensor Extrinsics

- **Sensor's (x,y) position (m):**
  - Sensor's position: [1.9, 0]

- **Sensor's height (m):**
  - Sensor's height: 1.1

- **Yaw angle of sensor mounted on ego vehicle (deg):**
  - Yaw angle: 0

- **Pitch angle of sensor mounted on ego vehicle (deg):**
  - Pitch angle: 1

- **Roll angle of sensor mounted on ego vehicle (deg):**
  - Roll angle: 0
Create highway double curve with drivingScenario

- Driver waypoints simulate distraction at curvature changes
Simulate distracted driver

Distracted driver departs lane
Simulate lane keep assist at distraction events
Compare distracted and assisted results

- Detect lane departure and maintain lane during distraction
Simulate lane following by increasing minimum safe distance
Graphically edit scenarios with Driving Scenario Designer
Export MATLAB code to generate scenarios
Explore what is required to follow high curvature paths

Follows curvy lanes at slower speed with wider field of view
Learn about synthesizing sensor detections to develop control algorithms with these examples

- **Simulate and generate C++ for model-predictive control and sensor fusion algorithms**

- **Simulate and generate C++ for model-predictive control with lane detections**

- **Edit roads, cuboid actors, and sensors with Driving Scenario Designer App drivingScenarioDesigner**

![Adaptive Cruise Control with Sensor Fusion](image1.png)

![Lane Keeping Assist with Lane Detection](image2.png)

![Generate Synthetic Detections from an Interactive Driving Scenario](image3.png)
Learn about modeling vehicle dynamics to develop control algorithms with these examples

- **Simulate vehicle dynamics** for closed loop design
  Vehicle Dynamics Blockset™

- **Co-simulate with Unreal Engine** and to set actor positions get camera image
  Vehicle Dynamics Blockset™
How can you use MATLAB and Simulink to develop planning algorithms?

- Deep learning
- Sensor models & model predictive control
- Perception
- Control
- Sensor fusion with live data
- Path planning
- Planning
Robotics System Toolbox introduced: Connectivity with the ROS ecosystem

- Communicate via ROS to integrate with externally authored ROS components
- Communication with Gazebo to visualize and simulated system
- Follow path for differential drive robot with ROS based simulator
We are investing in design and simulation of path planning for automobiles

Motion planning: Plan path to next waypoint (RRT*)

Rapidly-exploring Random Tree (RRT*)
Learn about developing path planning algorithms with these examples

- **Plan path** for automobile given pre-defined map
  Automated Driving System Toolbox™

- **Plot map tiles** using World Street Map (Esri)
  Automated Driving System Toolbox™

- **Simulate V2X communication** to assess channel throughput
  LTE System Toolbox™
Examples of how you can use MATLAB and Simulink to develop automated driving algorithms

Perception

Control

Planning

Deep learning

Sensor models & model predictive control

Sensor fusion with live data

Path planning
MathWorks can help you customize MATLAB and Simulink for your automated driving application

- **Web based ground truth labeling**
  - Consulting project with Caterpillar
  - [2017 MathWorks Automotive Conference](#)

- **Lidar ground truth labeling**
  - Joint presentation with Autoliv
  - SAE Paper 2018-01-0043
  - 2018 MathWorks Automotive Conference

- **Lidar sensor model for Unreal Engine**
  - Joint paper with Ford
  - SAE Paper 2017-01-0107
How can we help you can use MATLAB and Simulink to develop automated driving algorithms?