

MATLAB EXPO

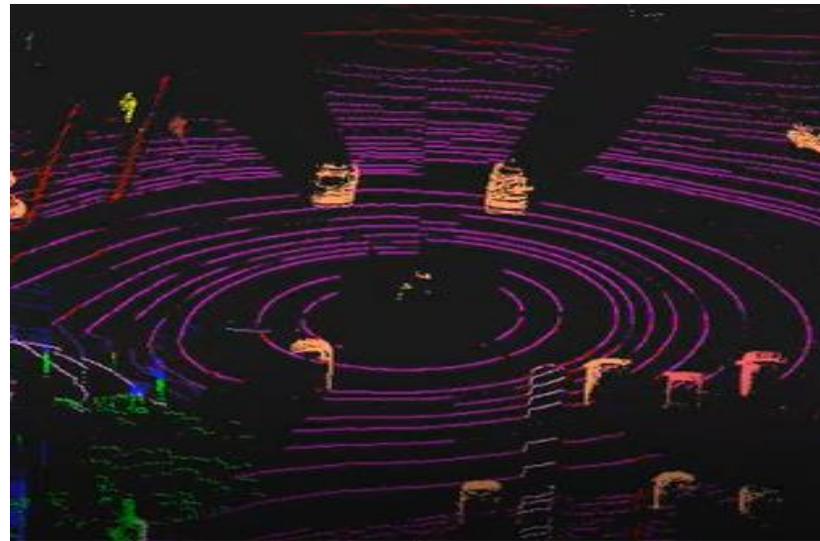
2021

레이더 및 라이다 데이터 처리를 위한 인공지능

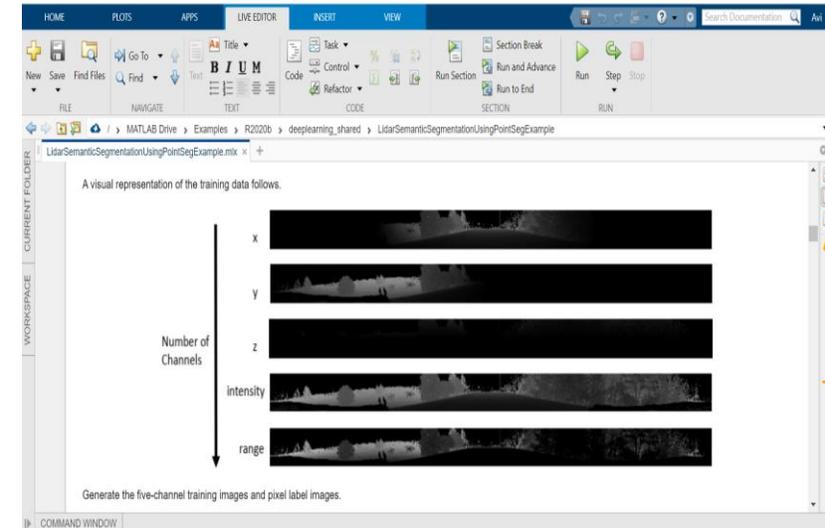
정승혁 차장



3 Things We'll Cover Today



- Data Synthesis
- Labeling
- Pre-processing
- Model selection and training
- Full system deployment



Insight

AI Applications for Radar and Lidar

Challenges

Common issues engineers face in practice

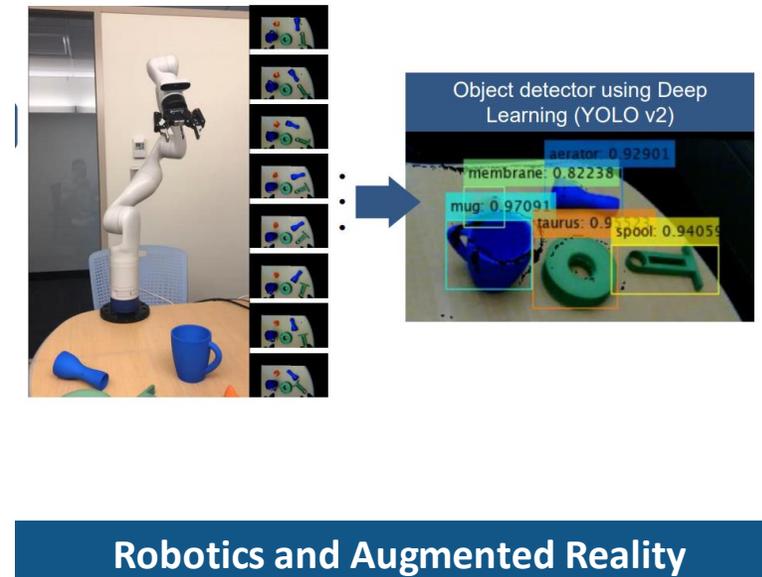
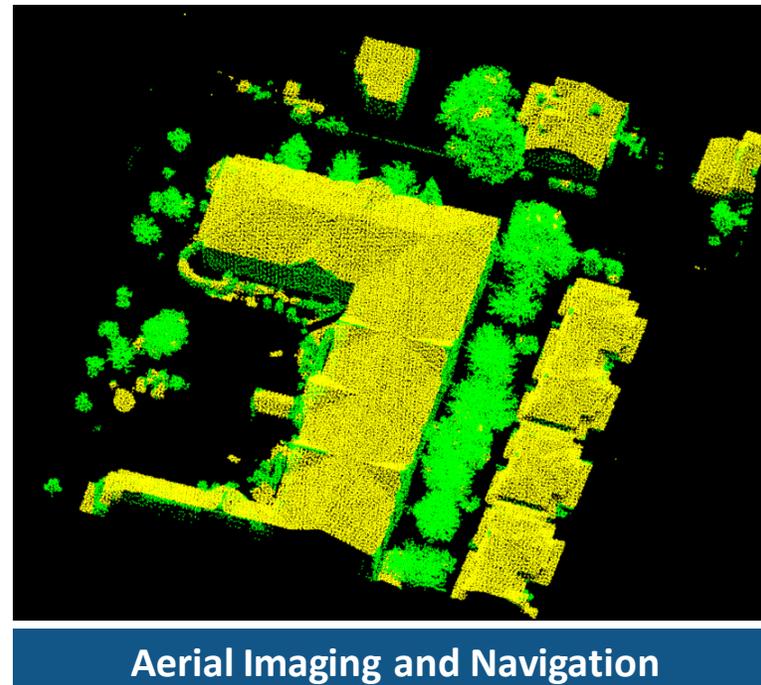
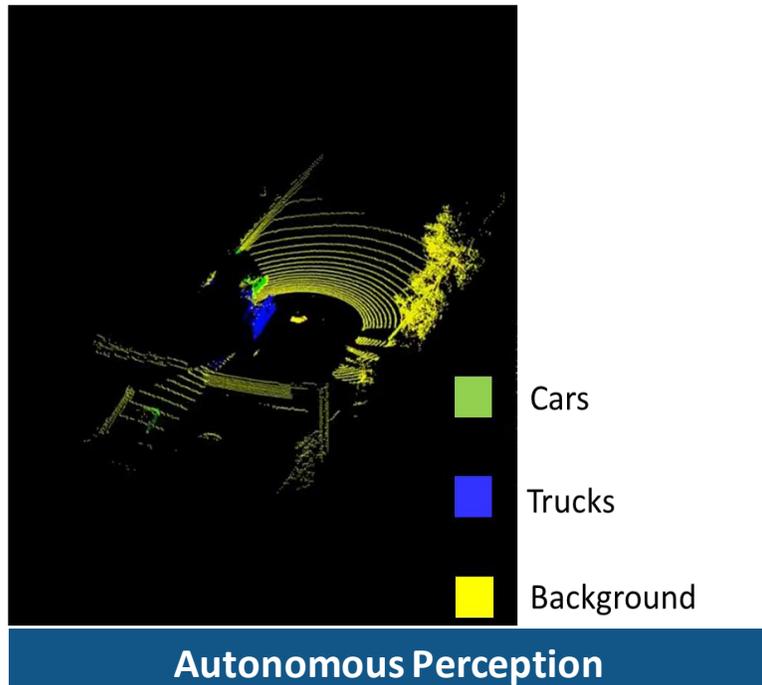
Interaction

AI models for radar and lidar data

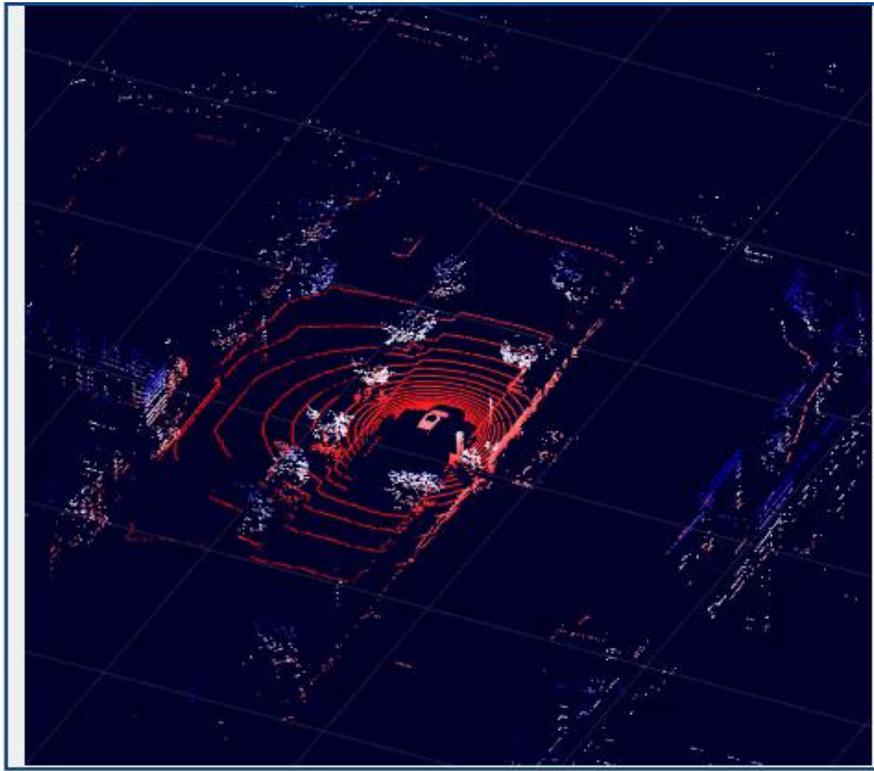
What is a lidar sensor and where is AI used ?

Lidar: **L**ight **d**etection **a**nd **r**anging

- Creates 2D or 3D point clouds representing depth using pulsed-light
- Also known as 3D laser scanner, laser scanner



What are the advantages and disadvantages of lidar sensors ?



Accurate
Depth



Dense
Data



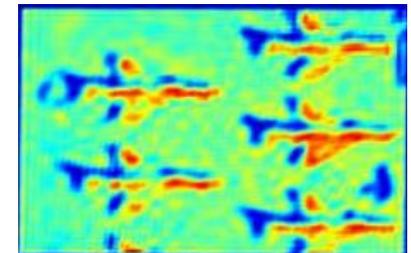
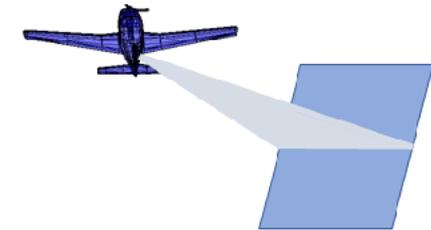
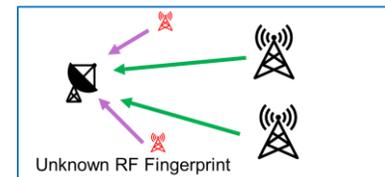
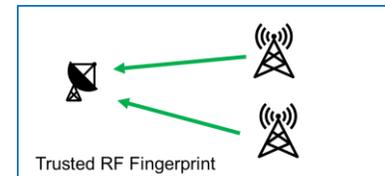
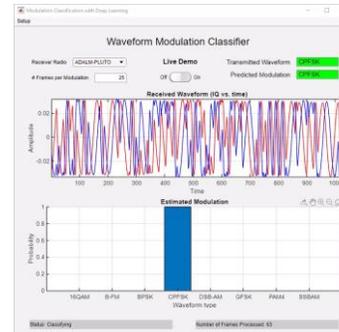
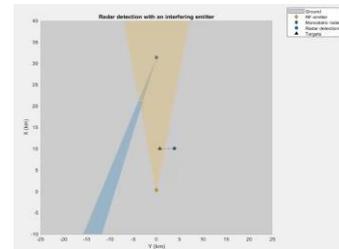
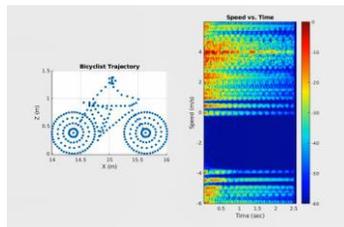
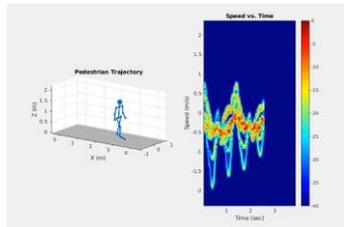
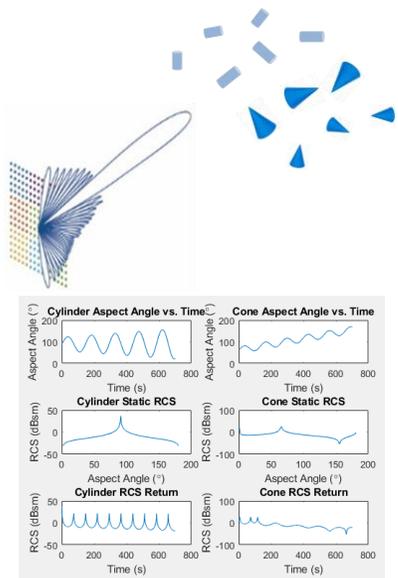
Disadvantages of lidar sensors

- Sensitive to rain, snow and weather effects
- Measurement effected by platform movement/vibration
- Accuracy drops as range increases

What is a radar sensor and where is AI used ?

Radar: Radio detection and ranging

- Use radio frequency echos to detect objects at a distance
- Estimate position, Doppler, and micro-Doppler.
- Generate images with 4D radar

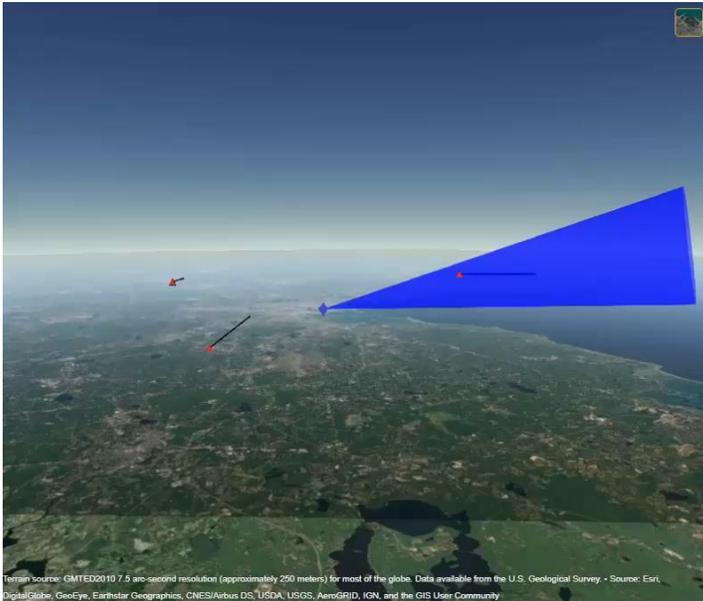


Target classification

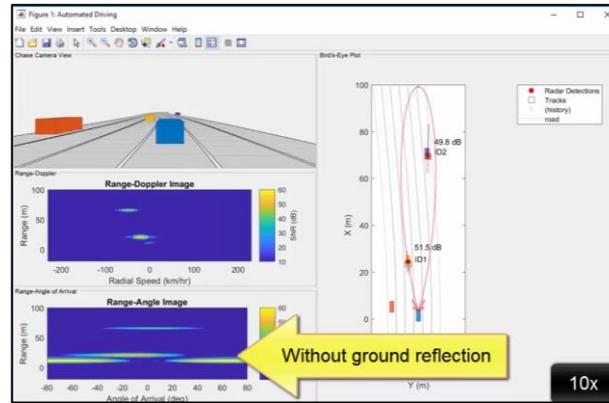
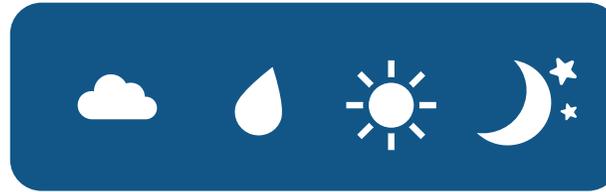
Signal identification

SAR imaging

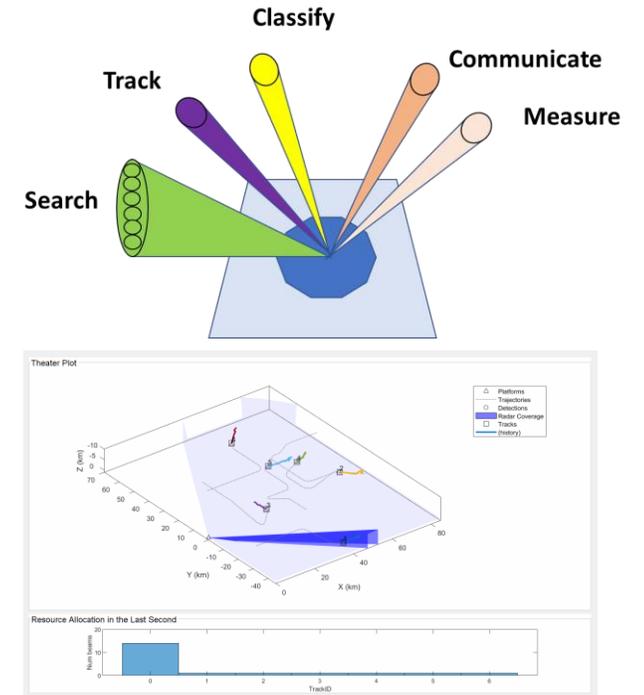
What are the advantages and disadvantages of radar sensors?



Long range operations



All weather, night and day



Flexibility

Disadvantages of radar sensors

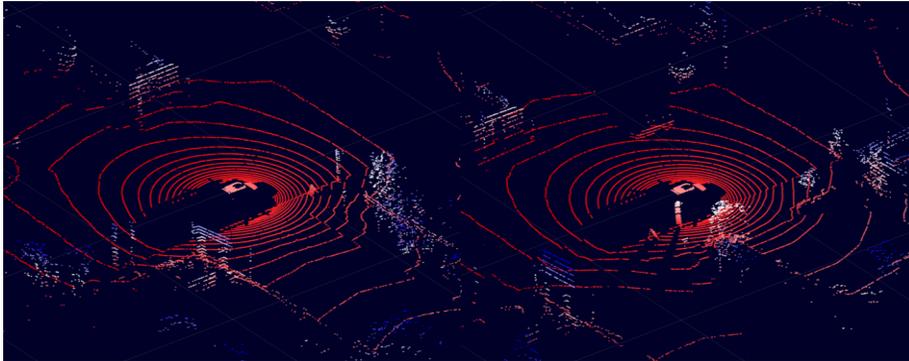
- Lower resolution than lidar
- Lower azimuthal resolution at longer ranges
- Multipath and clutter cause ghost detections and false detections

What are the common challenges engineers face using AI with radar and lidar ?

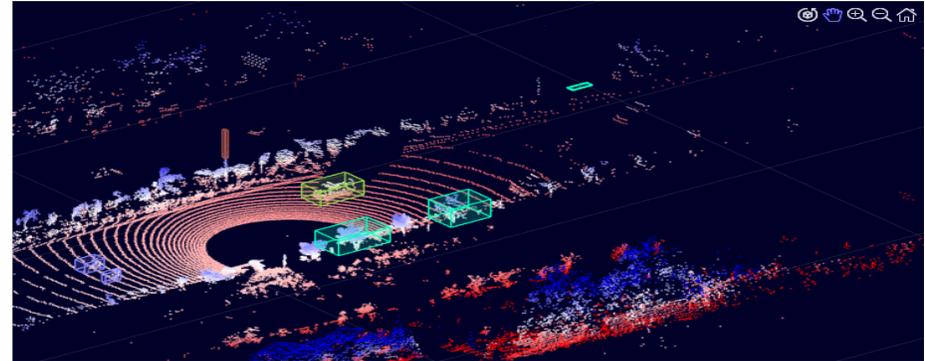
1. **Labeling recorded data** for AI training is manual and time consuming
2. Little-**no recorded data** to train models for safety-critical applications
3. **Unfamiliarity with AI models** for radar and lidar
4. Unclear how to **pre-process sensor signals** for best results
5. Real-world systems require **deployment of more than AI model**

Challenge

Labeling data is repetitive, manual and time consuming



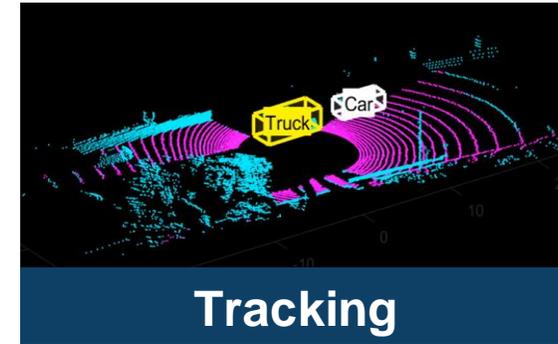
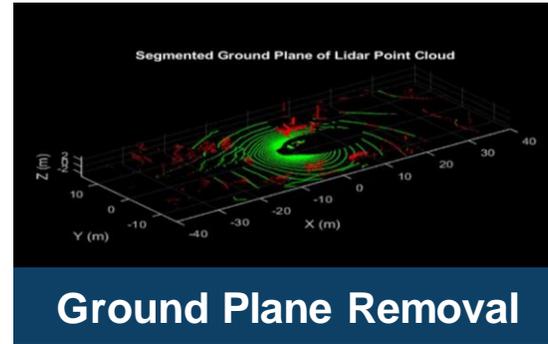
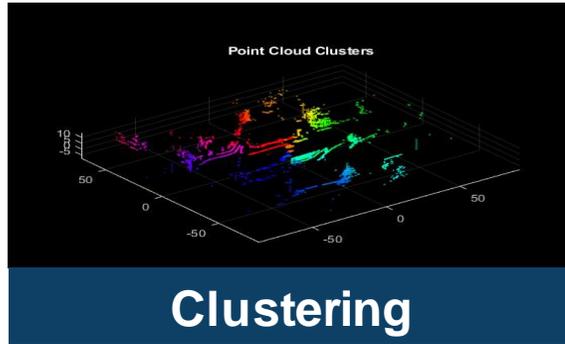
Repetitive and manual
Very little variation frame-frame



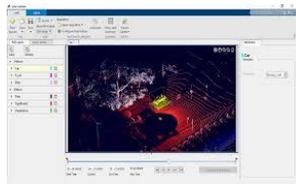
Noise
Majority of points not required to train AI model

Two steps to improving accuracy and efficiency of labeling process

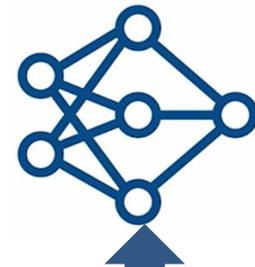
1. Automation using non-AI techniques



2. Iterative training and labeling

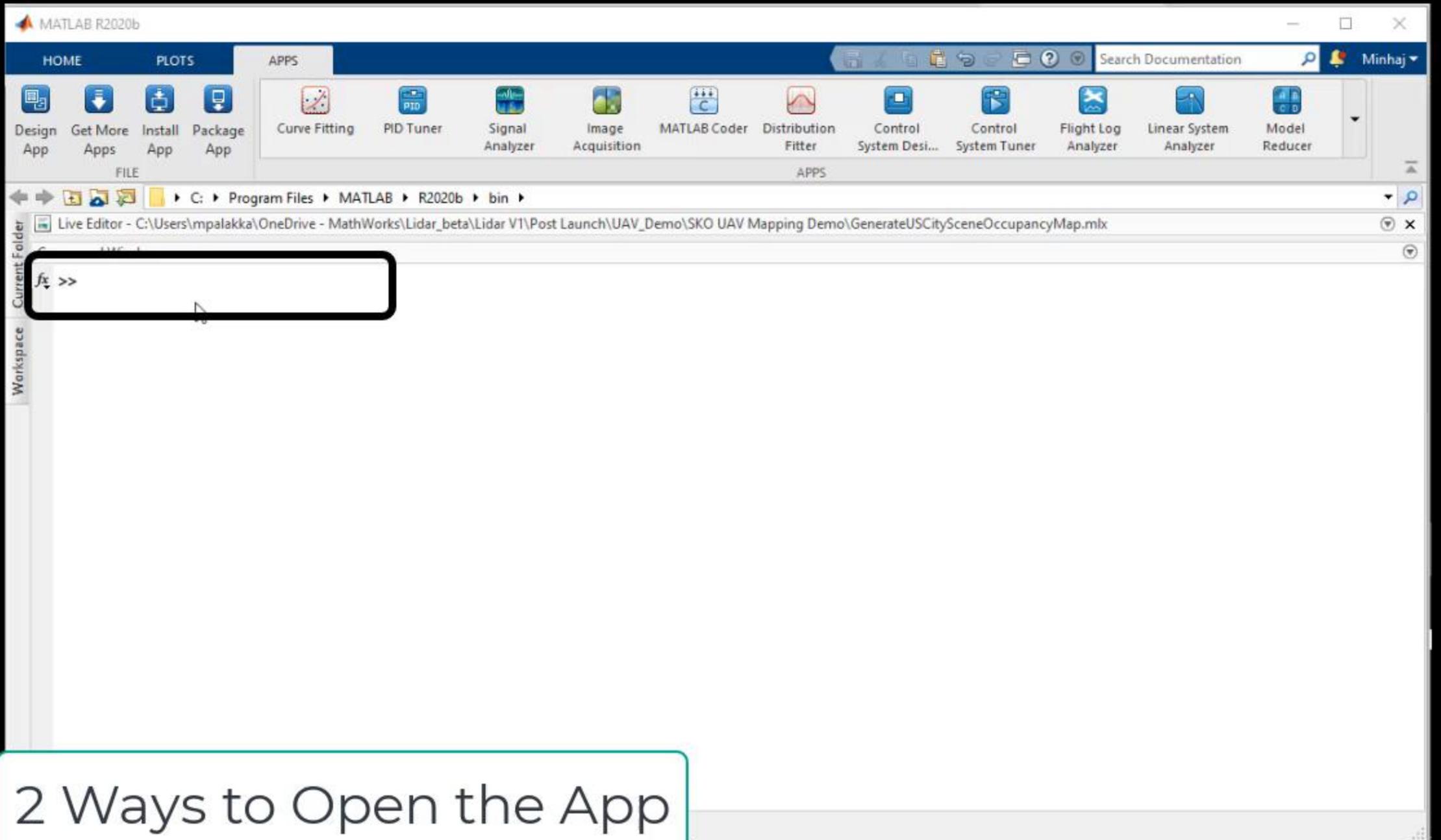


Train Model



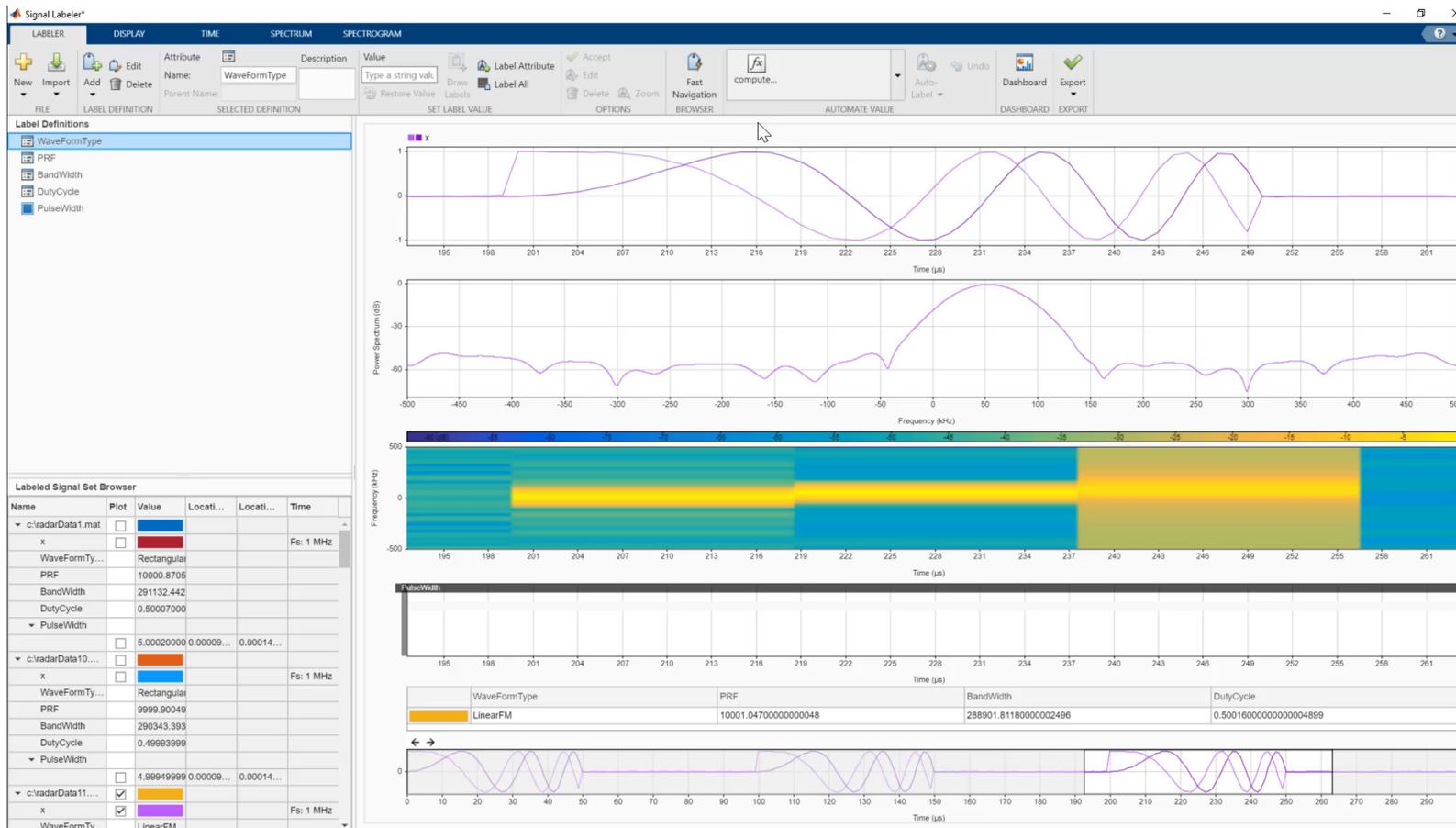
Iteration and Refinement





2 Ways to Open the App

Labelling radar signals can also be done automatically

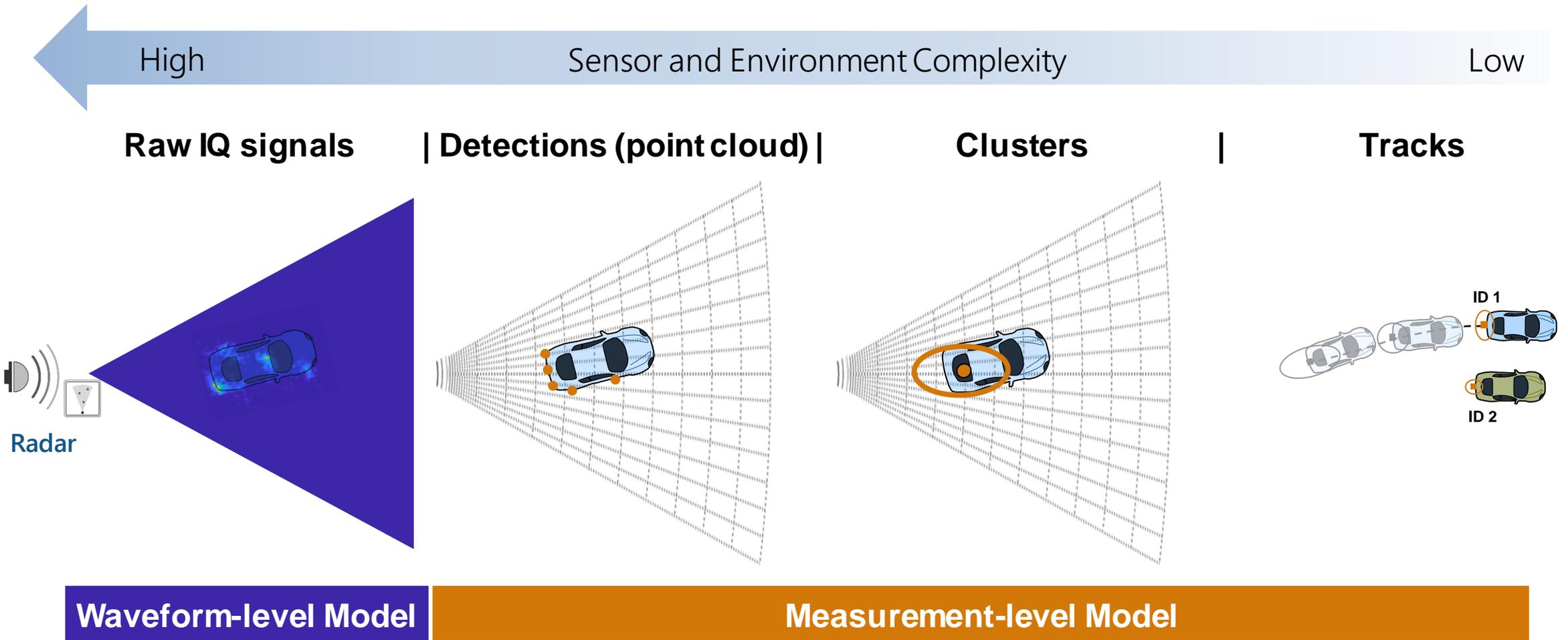


Automatically label signals with custom functions

Explore and label signals with time, frequency, and time-frequency views

Track labelling statistics with integrated dashboards

Simulating radar data in MATLAB and Simulink



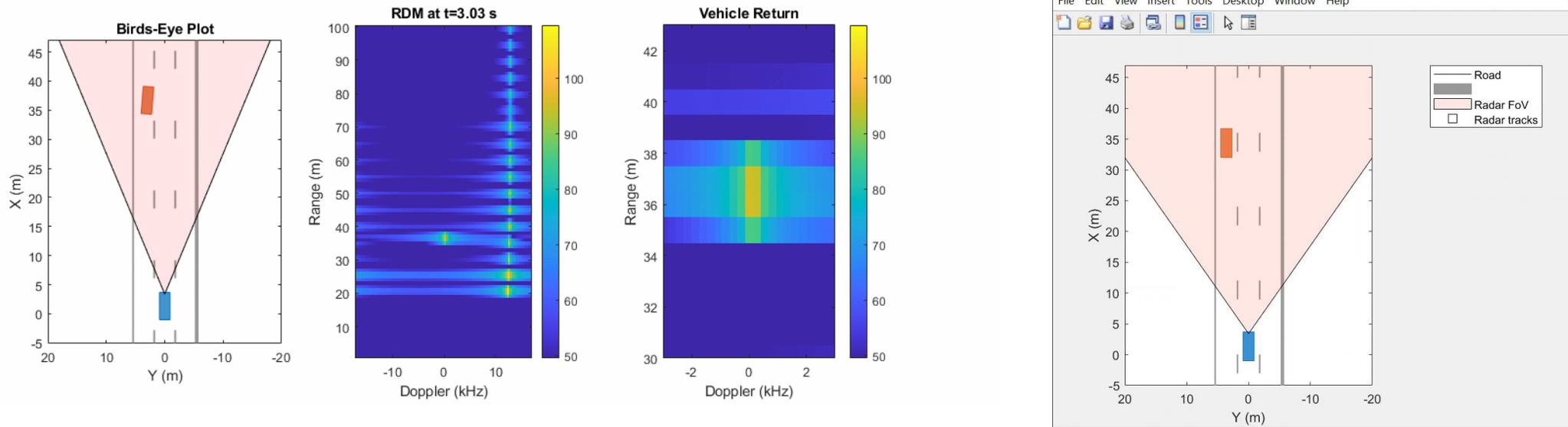
Simulating radar data in MATLAB and Simulink



Raw IQ signals

| Detections (point cloud) |

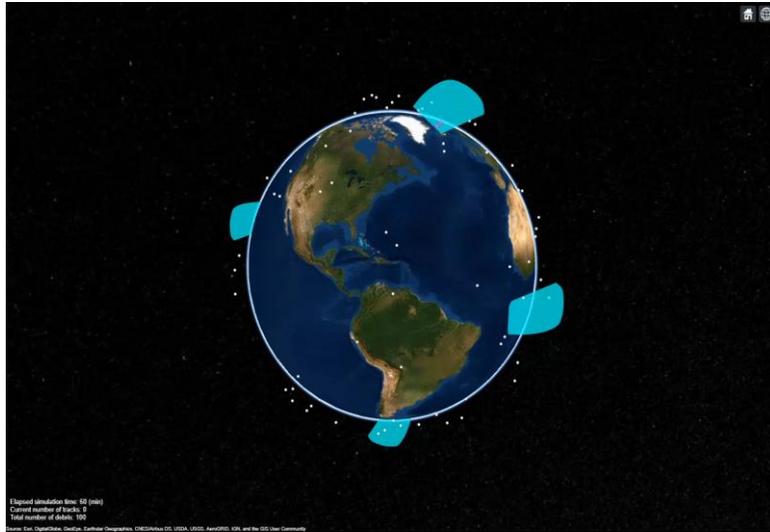
Tracks



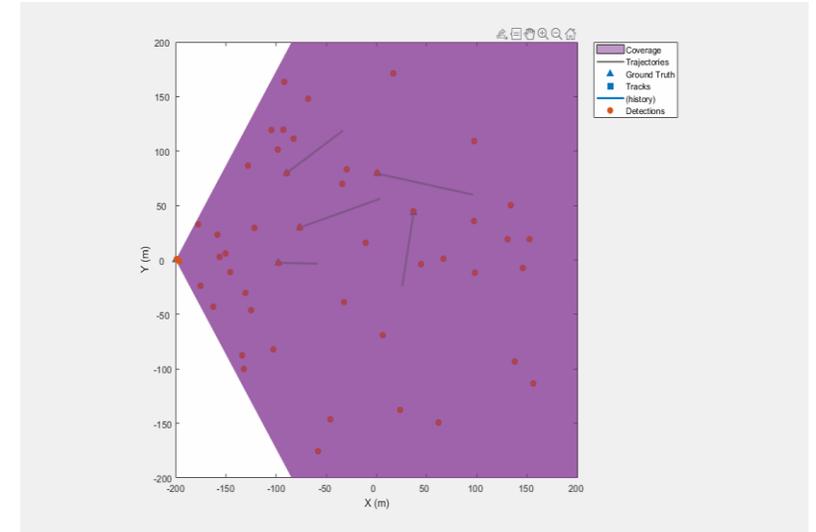
Waveform-level Model

Measurement-level Model

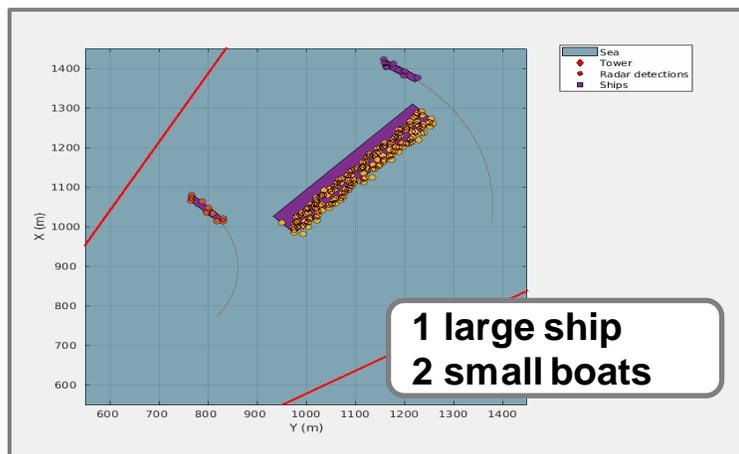
Wide range of data synthesis options for radar systems



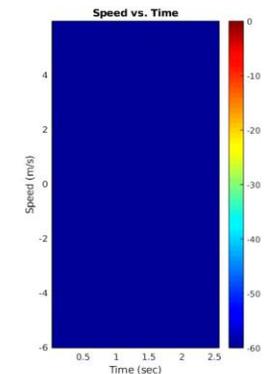
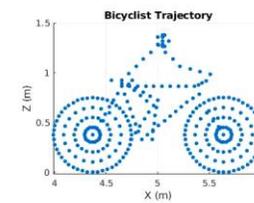
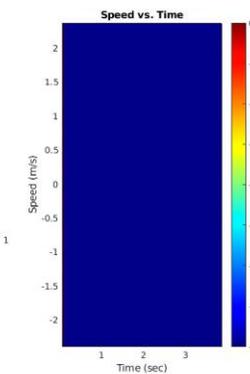
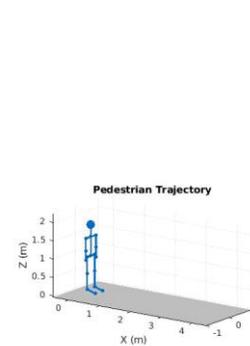
Long distance, multi-object operations



High clutter environments



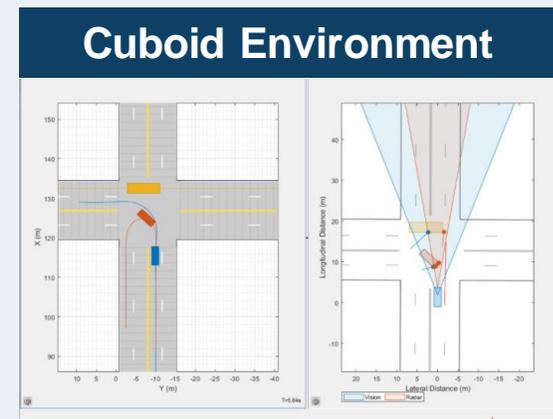
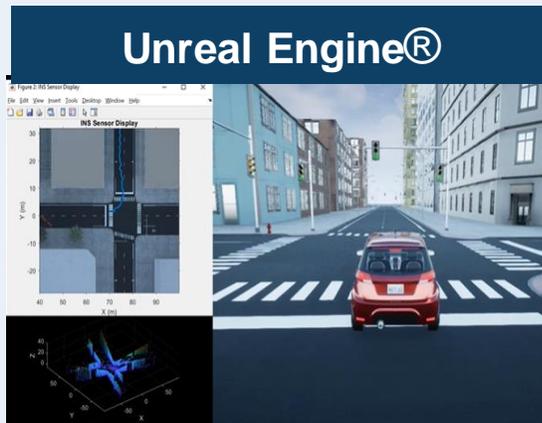
Extended objects



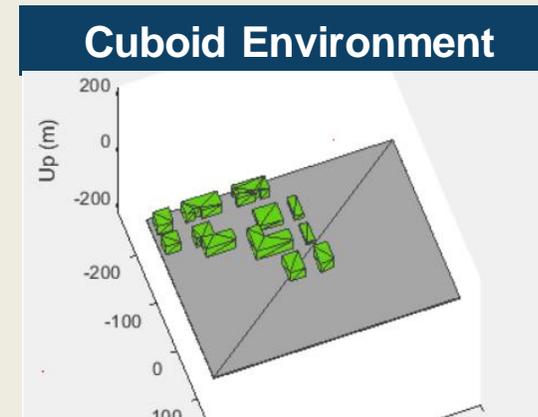
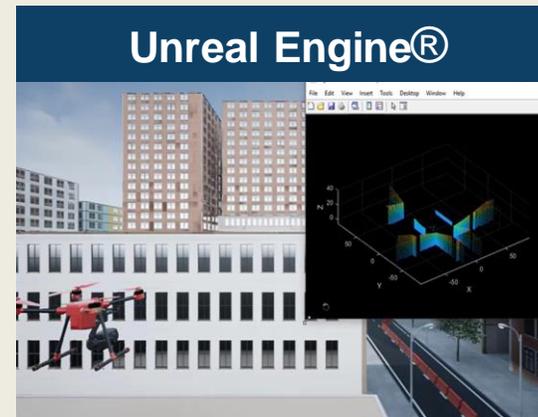
Micro-Doppler signatures

Simulating lidar sensor data in MATLAB and Simulink

Automated Driving Toolbox



UAV Toolbox



3D Scene Creation



What are the common challenges engineers face using AI with radar and lidar ?

1. Labeling recorded data for AI training is manual and time consuming
2. Little-no recorded data to train models for safety-critical applications
3. **Unfamiliarity with AI models** for radar and lidar
4. Unclear how to **pre-process sensor signals** for best results
5. Real-world systems require **deployment of more than AI model**

Challenge

Lack of knowledge on combination of model-type and data format best results

arXiv:1710.07368v1 [cs.CV] 19 Oct 2017

SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LIDAR Point Cloud

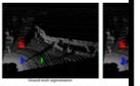
Bichen Wu, Alvin Wan, Xiangyu Yue and Kurt Keutzer
UC Berkeley
{bichen, alvinwan, yyue, keutzer}@berkeley.edu

Abstract—In this paper, we address automatic segmentation of road-objects from 3D LIDAR point clouds. In particular, we wish to detect and categorize instances of interest, such as cars, pedestrians and cyclists. We formulate this problem as a point-wise classification problem, and propose an end-to-end pipeline called SqueezeSeg based on convolutional neural networks (CNN). The CNN takes a transformed LIDAR point cloud as input and directly outputs a point-wise label map, which is then refined by a conditional random field (CRF) implemented as a recurrent layer. Instance-level labels are then obtained by conventional clustering algorithms. Our CNN model is trained on LIDAR point clouds from the KITTI [3] dataset, and our point-wise segmentation labels are derived from 3D bounding boxes from KITTI. To obtain extra training data, we built a LIDAR simulator using Grand Theft Auto V (GTA V) as a popular video game, to synthesize large amounts of realistic training data. Our experiments show that SqueezeSeg achieves high accuracy with substantially fast and stable runtime (0.7-1.0 ms per frame), highly desirable for autonomous driving applications. Furthermore, additionally training on synthesized data boosts validation accuracy on real-world data. Our source code and synthesized data will be open-sourced.

1. INTRODUCTION

Autonomous driving systems rely on accurate, real-time and robust perception of the environment. An autonomous vehicle needs to accurately categorize and locate “road-objects”, which we define to be driving-related objects such as cars, pedestrians, cyclists, and other obstacles. Different autonomous driving solutions may have different combinations of sensors, but the 3D LIDAR sensor is one of the most prevalent components. LIDAR sensors directly produce distance measurements of the environment, which are then used by vehicle controllers and planners. Moreover, LIDAR sensors are robust under almost all lighting conditions, whether it be day or night, with or without glare and shadows. As a result, LIDAR based perception tasks have attracted significant research attention.

In this work, we focus on road-object segmentation using (Veloxye style) 3D LIDAR point clouds. Given point cloud output from a LIDAR scanner, the task aims to isolate objects of interest and predict their categories, as shown in Fig. 1. Previous approaches comprise or use parts of the following stages: Remove the ground, cluster the remaining points into instances, extract (hand-crafted) features from each cluster, and classify each cluster based on its features. This workflow, despite its popularity [2], [3], [4], [5] has several disadvantages: a) Ground segmentation in the above



pipeline usually relies on hand-crafted rules – some approaches rely on a set of other more complicated features (normals [7] or invariant descriptors [8]) or invariant descriptors [8] fail to generalize and the latter of course preprocessing. Multi-stage pipeline effects of compounded errors, and classification algorithms in the pipeline above in context, most importantly the immense object. c) Many approaches rely on heuristic algorithms such as RANSAC (Random Sample Consensus) [9], GP-INLAC (Gaussian Process Inference) [10], or alignment of normals and accuracy of these algorithms depend on the quality of random initializations can be unstable. This instability is not embedded applications such as autonomous driving. An alternative approach use deep learning to develop a single-stage pipeline, and algorithms.

In this paper, we propose an end-to-end convolutional neural networks (CNN) with recurrent conditional random field (CRF) applied to segmentation tasks on 3D LIDAR point clouds. We apply CNNs to 3D LIDAR point clouds that accept transformed LIDAR point clouds as input and directly outputs a point-wise label map of labels, which is then refined by a CRF model. Instance-level labels are then obtained by conventional clustering (DBSCAN) on points within a category.

arXiv:1812.05784v2 [cs.LG] 7 May 2019

PointPillars: Fast Encoders for Object Detection from Point Clouds

Alex H. Lang, Sourabh Vora, Holger Caesar, Labing Zhou, Jong Yang, Oscar Beijbom
Waymo, an APTV company
{alex, sourabh, holger, labing, jinyang, oskar}@waymo.com

Abstract

Object detection in point clouds is an important aspect of many robotics applications such as autonomous driving. In this paper we consider the problem of encoding a point cloud into a format appropriate for a downstream detection pipeline. Recent literature suggests two types of encoders: hand-crafted encoders tend to be fast but sacrifice accuracy, while encoders that are learned from data are more accurate but slower. In this work we propose PointPillars, a novel encoder which utilizes PointNet to learn a representation of point clouds organized in vertical columns (pillars). While the encoded features can be used with any standard 2D convolutional detection architecture, we further propose a lean downstream network. Extensive experimentation shows that PointPillars outperforms previous encoders with respect to both speed and accuracy by a large margin. Despite only using lidar, our full detection pipeline significantly outperforms the state of the art, even among fusion methods, with respect to both the 3D and bird's eye view KITTI benchmarks. This detection performance is achieved while running at 62 Hz: a 2-4 fold runtime improvement. A faster version of our method matches the state of the art at 105 Hz. These benchmarks suggest that PointPillars is an appropriate encoder for object detection in point clouds.

1. Introduction

Deploying autonomous vehicles (AV) in urban environments poses a difficult technological challenge. Among other tasks, AVs need to detect and track moving objects such as vehicles, pedestrians, and cyclists in real-time. To achieve this, autonomous vehicles rely on several sensors out of which the lidar is arguably the most important. A lidar uses a laser scanner to measure the distance to the environment, thus generating a sparse point cloud representation. Traditionally, a lidar robotics pipeline interprets such point clouds as object detections through a bottom-up pipeline involving background subtraction, followed by spatiotemporal clustering and classification [1], [2].

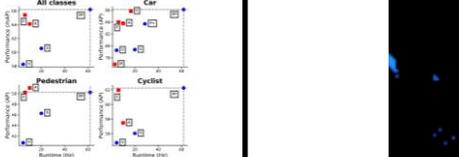
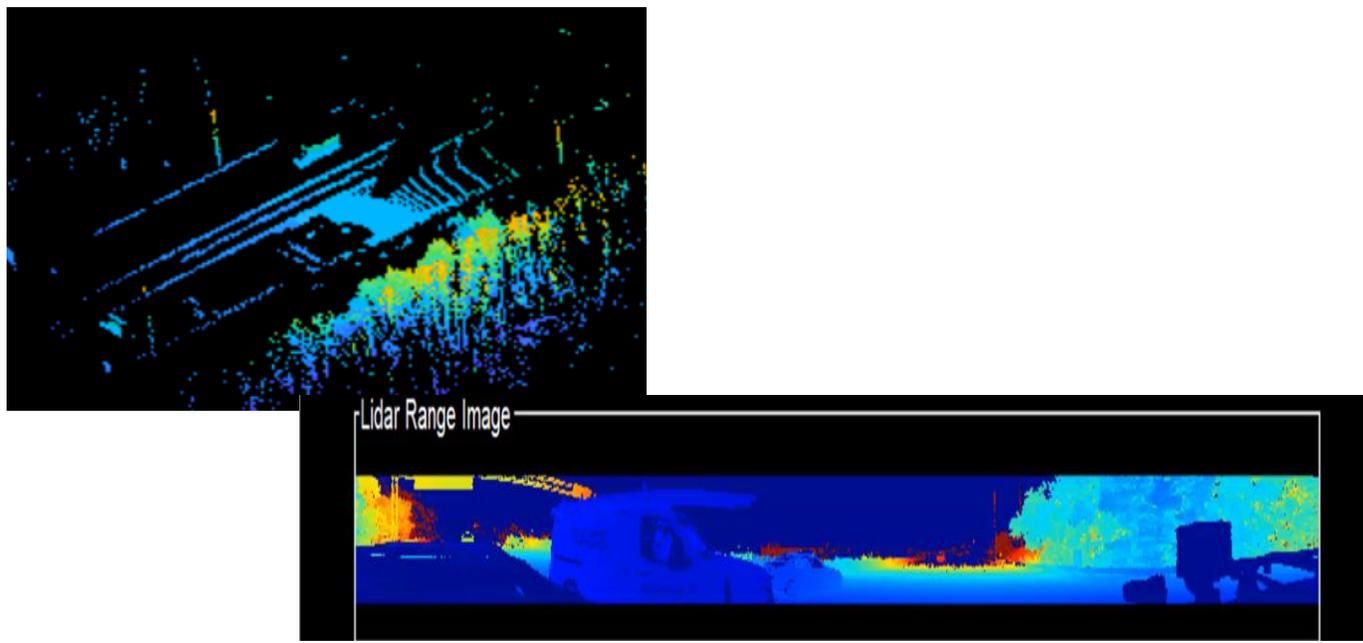


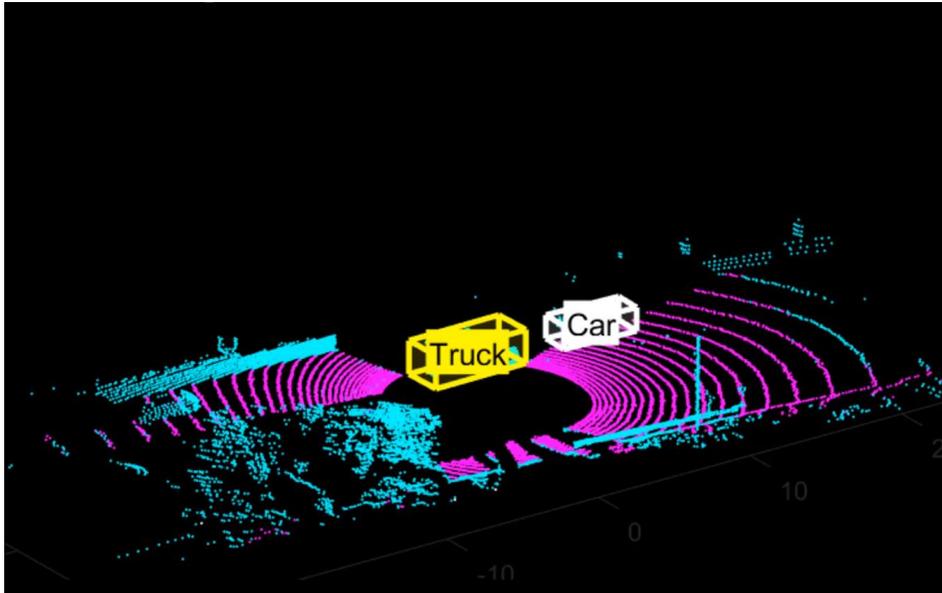
Figure 1. Bird's eye view performance vs speed for our proposed PointPillars. (PP) method on the KITTI [1] test set. Lidar-only methods shown as blue circles, lidar & vision methods shown as red squares. Also shown are top methods from the KITTI leader board: (M) MYO [3], (AV) AVO [4], (C) CarNet [15], (S) SqueezeSeg [11], (P) PointNet++ [12], (S) SECOND [13], (R) RFDNet [14]. PointPillars outperforms all other lidar-only methods in terms of both speed and accuracy by a large margin. It also outperforms all fusion-based methods except on pedestrians. Similar performance is achieved on the 3D metric (Table 2).



What model do I use ?
There are so many research papers.

How do I train a model ?
Raw sensor data or transformed.

MATLAB provides a curated library of models with different inputs and styles



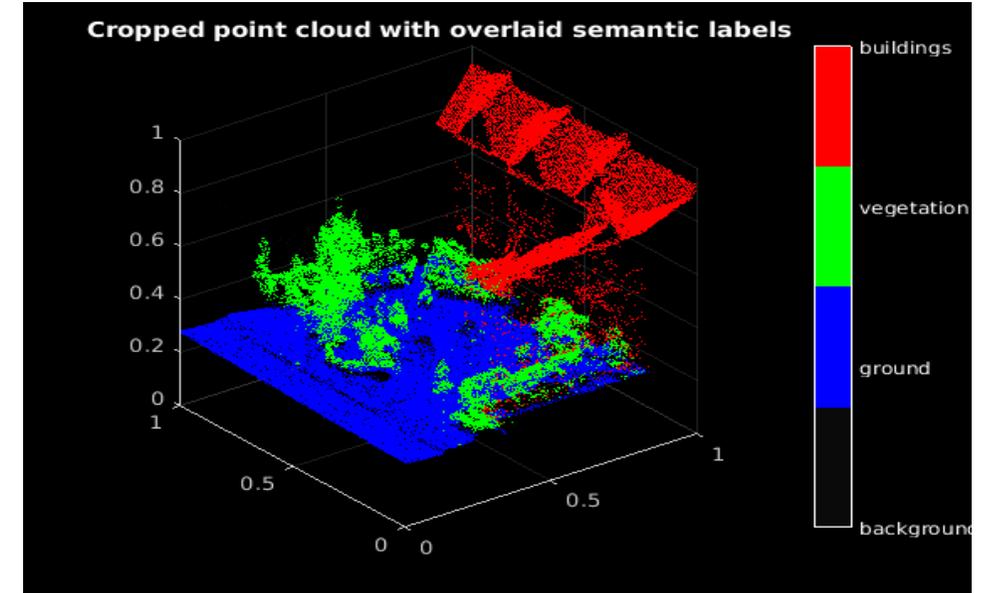
Object Detection

3D bounding box detection and classification

Curated Models

1. PointPillars

Raw
Data



Semantic Segmentation

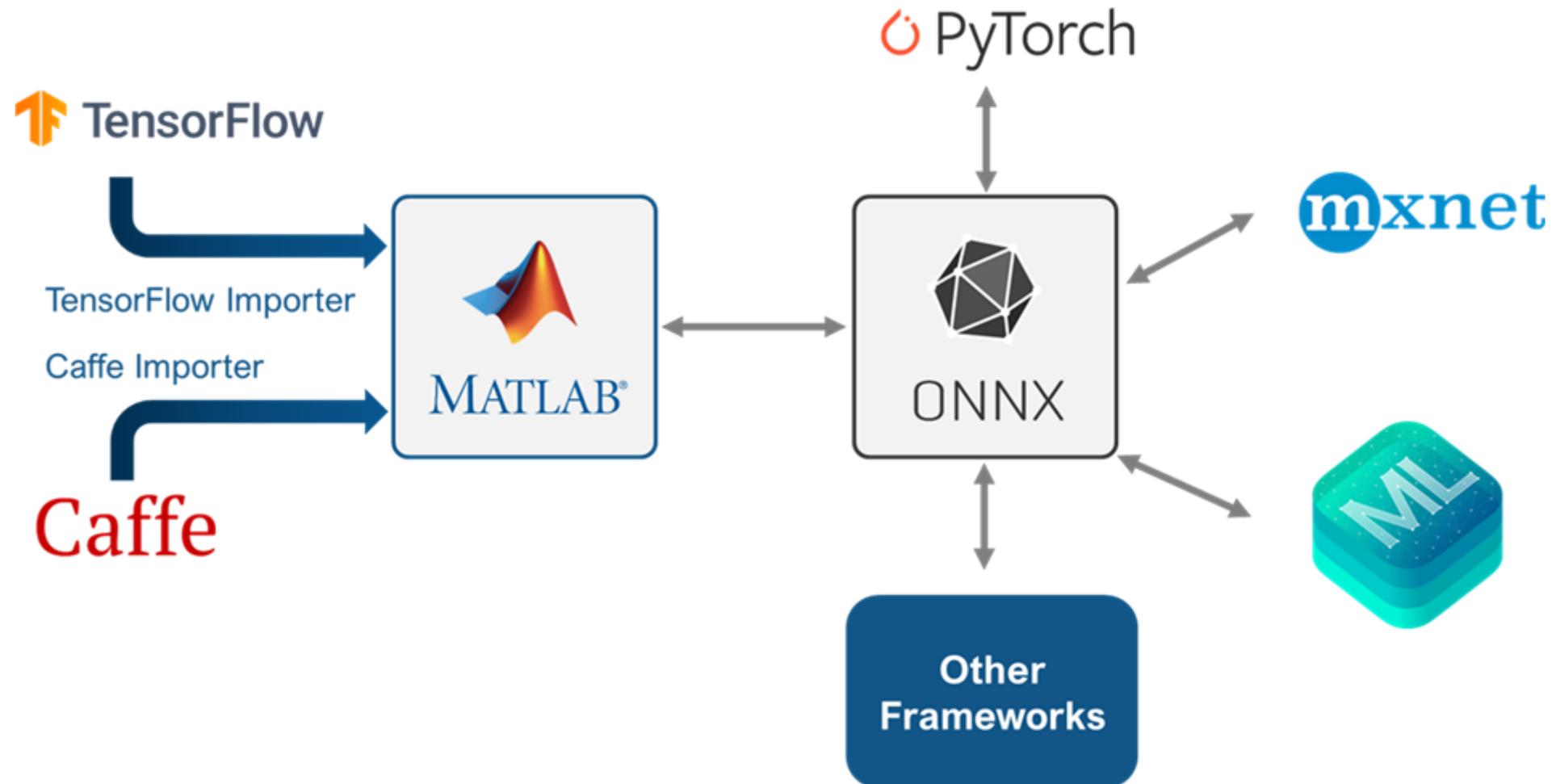
Classify each data point with label

Curated Models

1. SqueezeSeg v2
2. PointSeg
3. SalsaNext
4. PointNet
5. PointNet++

Image
Data

Import models from open source AI frameworks



Lidar 3-D Object Detection Using PointPillars Deep Learning

Load Data

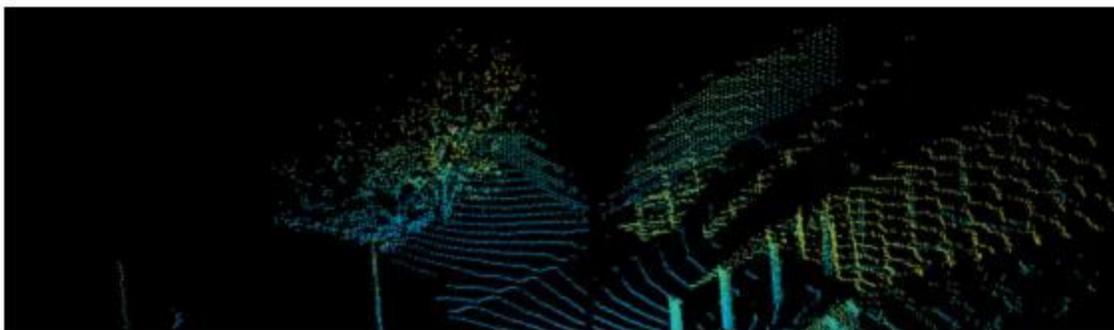
```
1 lidarURL = 'https://www.mathworks.com/supportfiles/lidar/data/WPI_LidarData.tar.gz';  
2 lidarData = downloadWPIData(outputFolder, lidarURL);
```

Load the 3-D bounding box labels.

```
3 load('WPI_LidarGroundTruth.mat', 'bboxGroundTruth');  
4 Labels = timetable2table(bboxGroundTruth);  
5 Labels = Labels(:,2:end);
```

Display the full-view point cloud.

```
6 figure  
7 ax = pcshow(lidarData{1,1}.Location);  
8 set(ax, 'XLim', [-50 50], 'YLim', [-40 40]);  
9 zoom(ax, 2.5);  
10 axis off;
```



Tune hyperparameters and reproduce training experiments

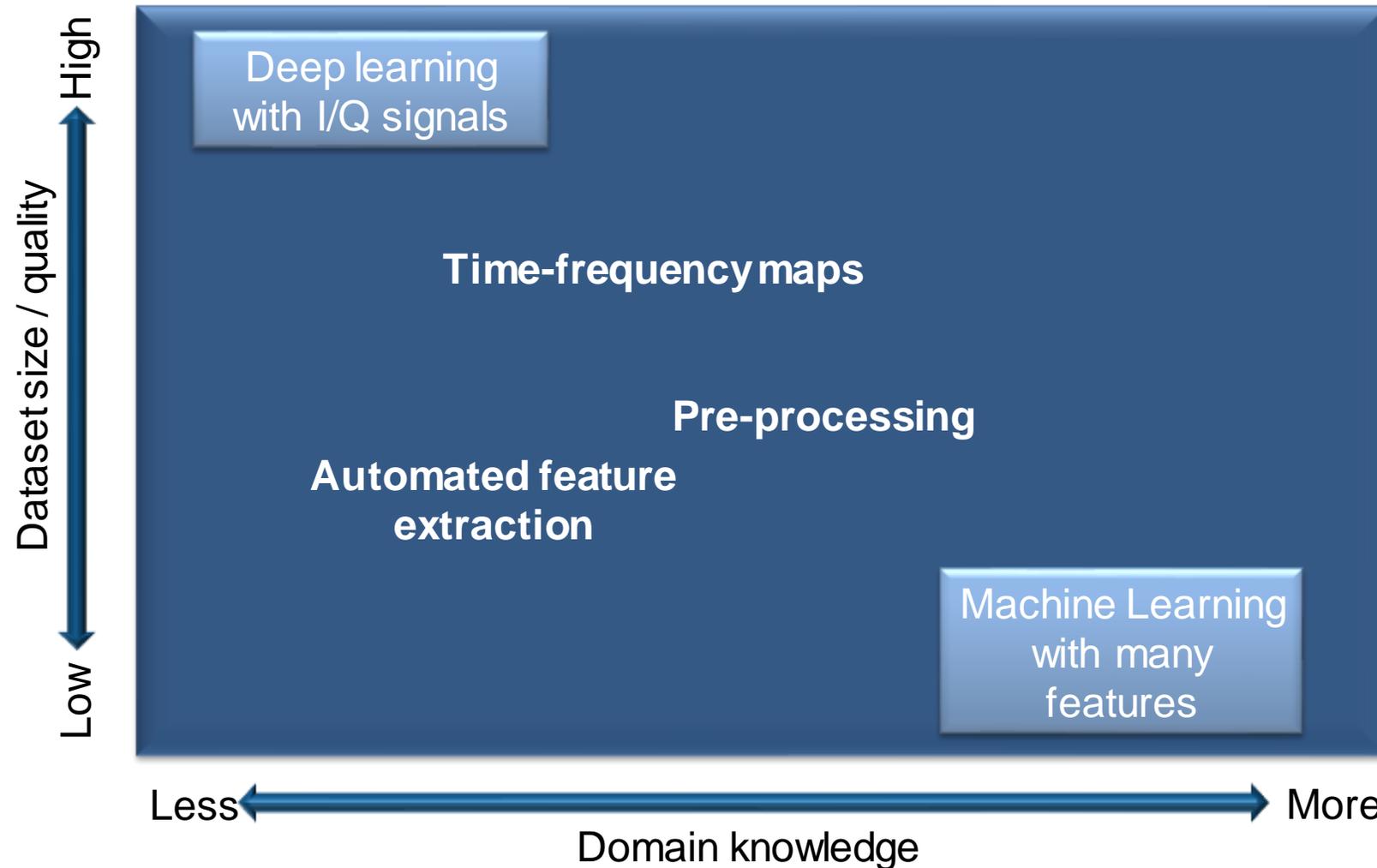
The screenshot shows the MATLAB Experiment Manager interface. The top toolbar includes buttons for New, Open, Save, Duplicate, Layout, Run, Stop, Training Plot, Confusion Matrix, Filter, and Export. The left sidebar shows the 'EXPERIMENT BROWSER' with a tree view for 'DigitsClassifier' containing 'Baseline Establishment' and 'Baseline Tuning'. The 'Baseline Tuning' section is expanded to show 'Result1 (Running)'. The main area displays 'Result Details' for 'Baseline Tuning' on 2/7/2020 at 12:53:36 PM, showing 7/16 trials completed. A summary table indicates 7 Complete, 1 Running, 8 Queued, 0 Stopped, 0 Error, and 0 Canceled trials. Below this is a detailed table of 16 trials with columns for Trial, Status, Progress, Elapsed Time, and various hyperparameters and metrics.

Trial	Status	Progress	Elapsed Time	myInitialLearn...	convFilterSize	Training Accu...	Training Loss	Validation Ac..
1	Complete	100.0%	0 hr 0 min 16 sec	1.0000e-6	3.0000	12.5000	2.6441	10.
2	Complete	100.0%	0 hr 0 min 15 sec	1.0000e-5	3.0000	25.7813	2.1228	20.
3	Complete	100.0%	0 hr 0 min 14 sec	0.0001	3.0000	64.8438	1.0878	42.
4	Complete	100.0%	0 hr 0 min 16 sec	0.0005	3.0000	90.6250	0.4648	49.
5	Complete	100.0%	0 hr 0 min 15 sec	1.0000e-6	4.0000	11.7188	2.4967	6.
6	Complete	100.0%	0 hr 0 min 15 sec	1.0000e-5	4.0000	23.4375	2.1213	14.
7	Complete	100.0%	0 hr 0 min 17 sec	0.0001	4.0000	72.6563	1.0283	39.
8	Running	30.7%	0 hr 0 min 4 sec	0.0005	4.0000			
9	Queued	0.0%		1.0000e-6	5.0000			
10	Queued	0.0%		1.0000e-5	5.0000			
11	Queued	0.0%		0.0001	5.0000			
12	Queued	0.0%		0.0005	5.0000			
13	Queued	0.0%		1.0000e-6	6.0000			
14	Queued	0.0%		1.0000e-5	6.0000			
15	Queued	0.0%		0.0001	6.0000			
16	Queued	0.0%		0.0005	6.0000			

What are the common challenges engineers face using AI with radar and lidar ?

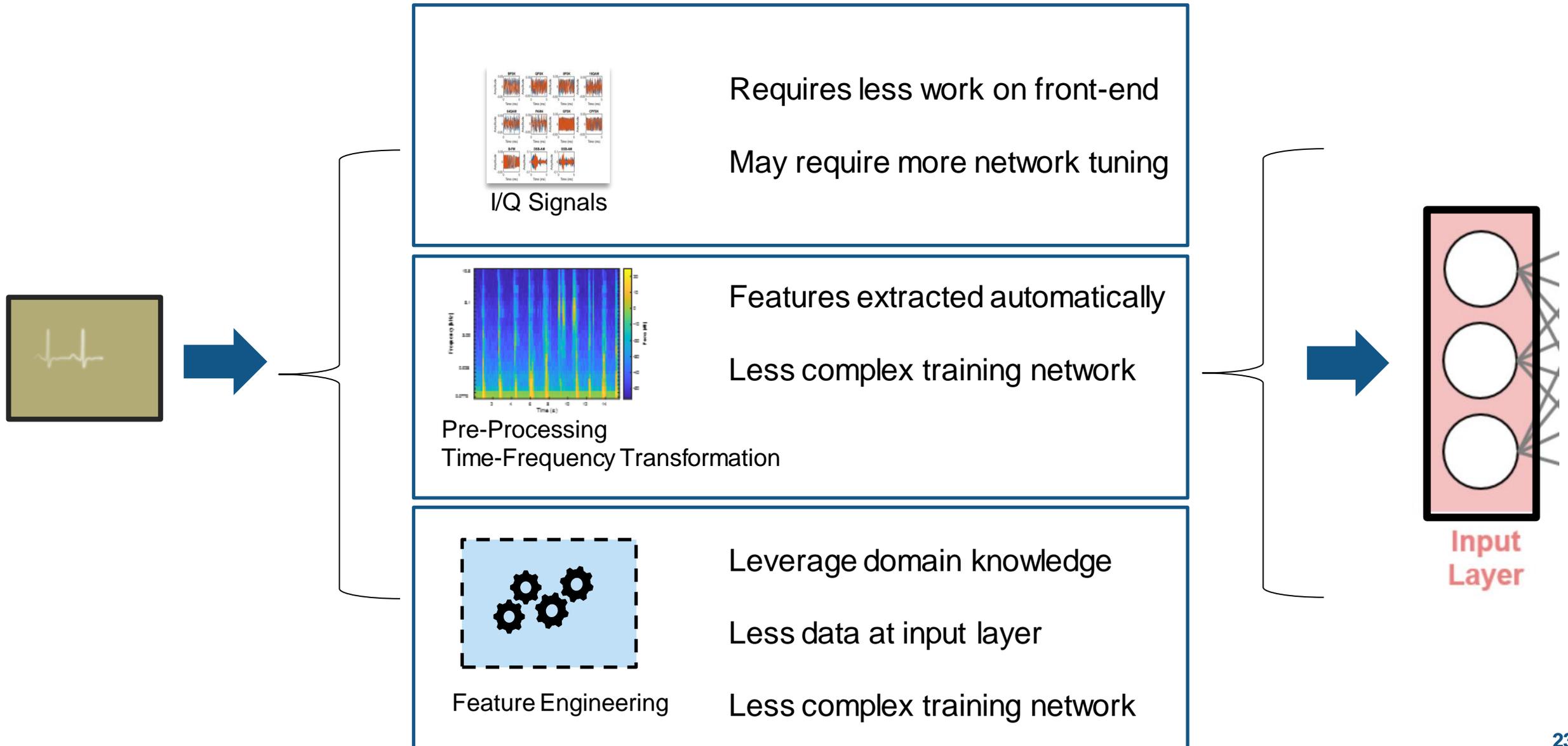
1. Labeling recorded data for AI training is manual and time consuming
2. Little-no recorded data to train models for safety-critical applications
3. Unfamiliarity with AI models for radar and lidar
4. Unclear how to pre-process sensor signals for best results
5. Real-world systems require deployment of more than AI model

Pre-processing radar data can improve performance of network

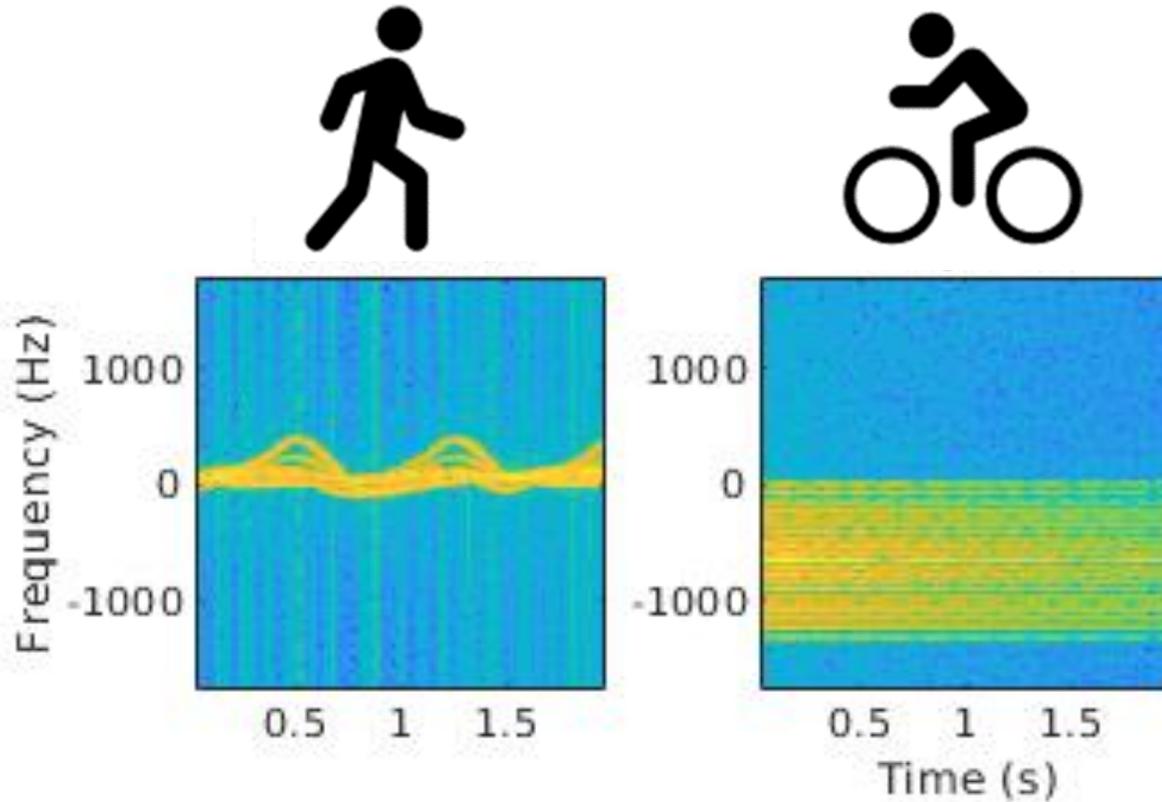


Dataset size vs. domain knowledge vs. compute resources

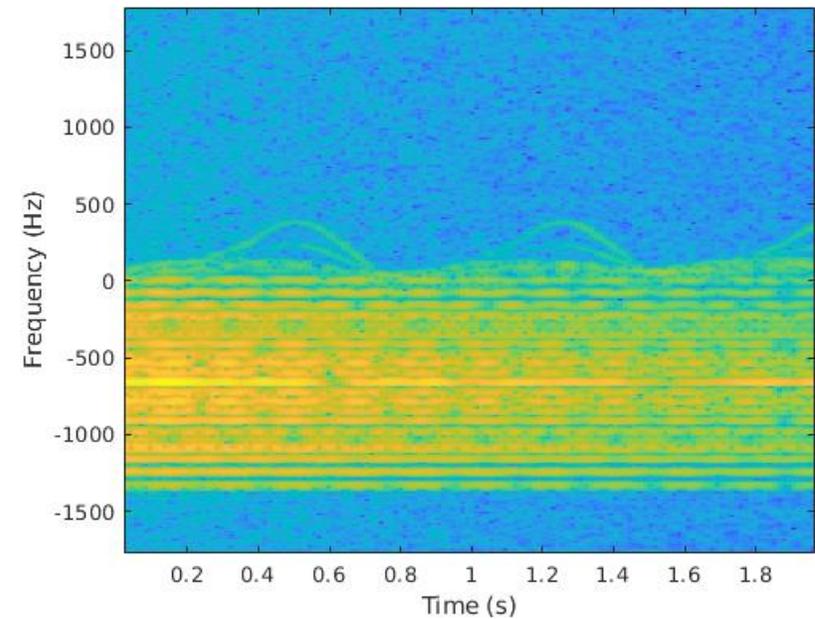
You can make the trade-off between pre-processing approaches



Time to test your ability to classify micro-Doppler returns ...



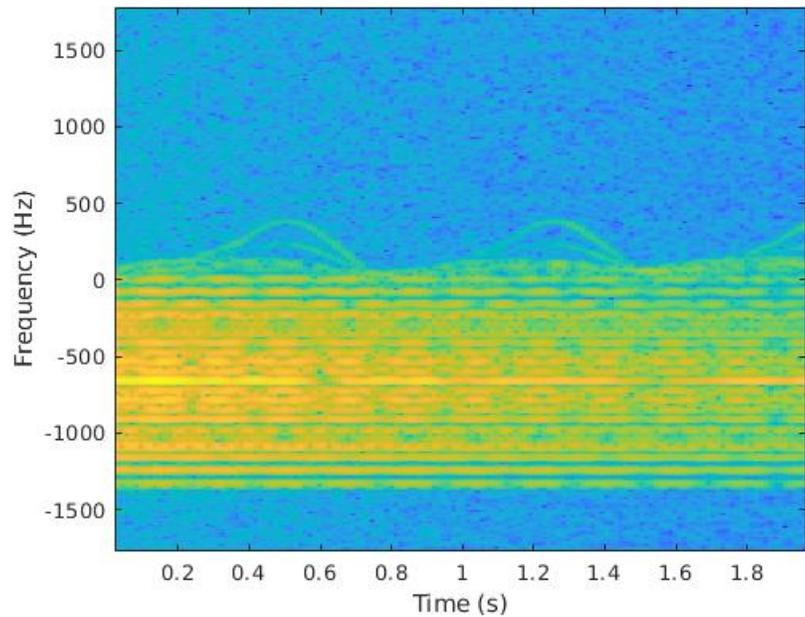
Is this a pedestrian or a bicyclist?



Ground truth – synthesized micro-Doppler

Poll

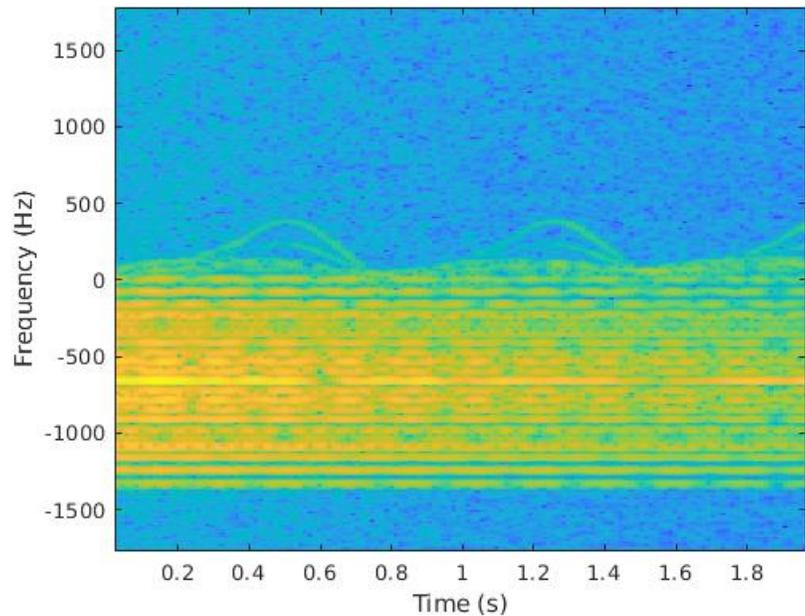
Is this a pedestrian or a bicyclist?



- A. One Pedestrian
- B. One Bicyclist
- C. One of each
- D. Not sure

And the answer is

Is this a pedestrian or a bicyclist?

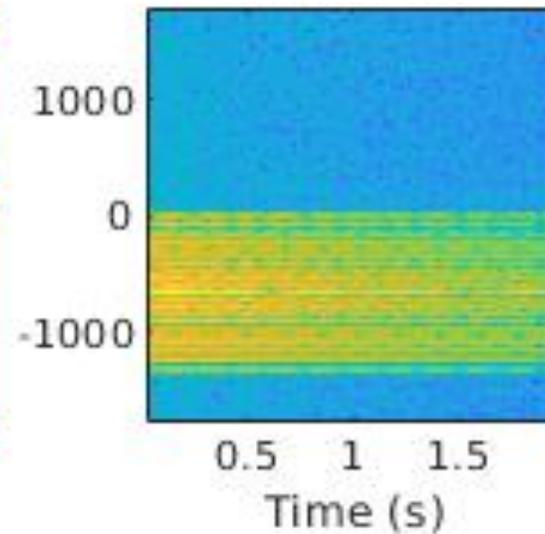
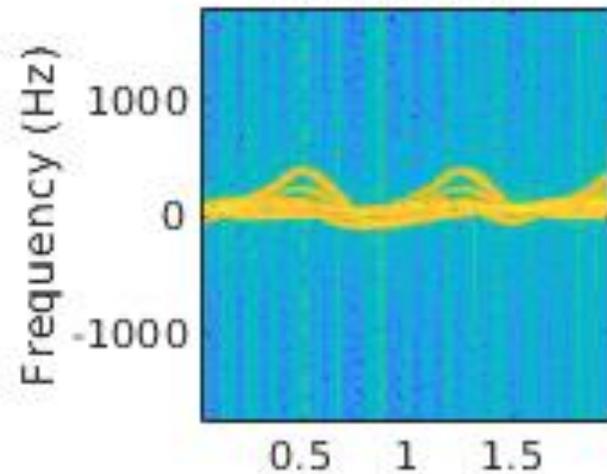


- A. Pedestrian
- B. Bicyclist
- C. One of each
- D. Not sure

This is a pedestrian and a bicyclist

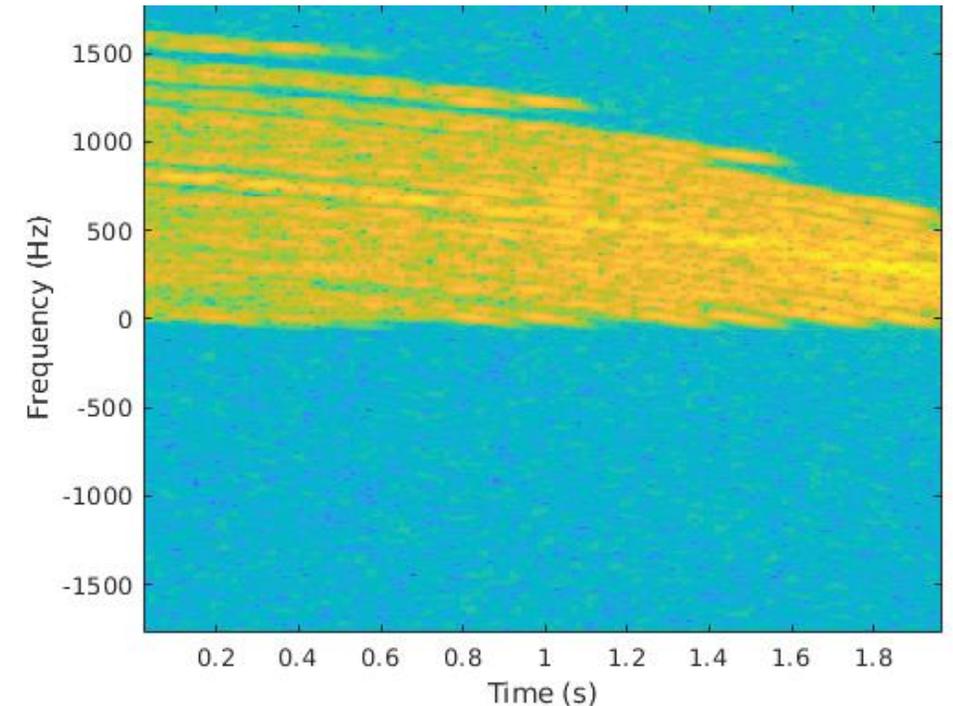
This one is a bit trickier. The network gets the correct answer

[Example Link](#)



Ground truth – synthesized micro-Doppler

Is this a pedestrian or a bicyclist?



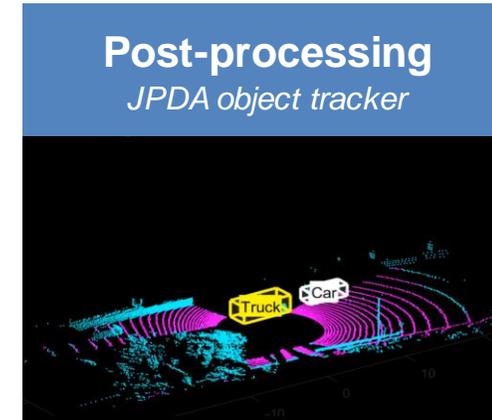
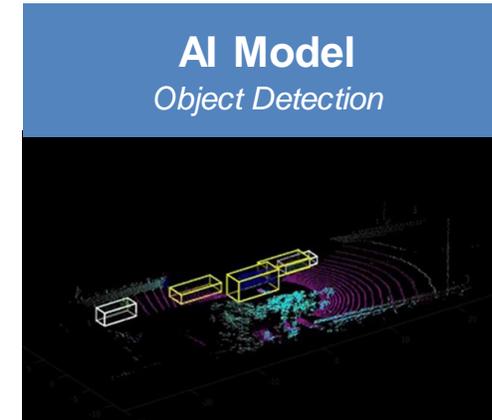
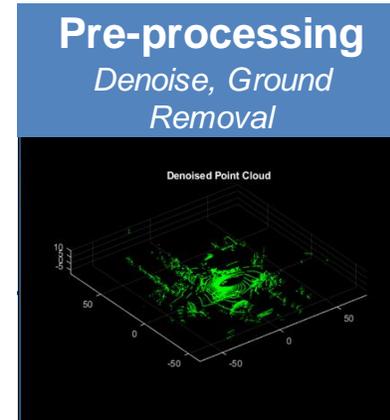
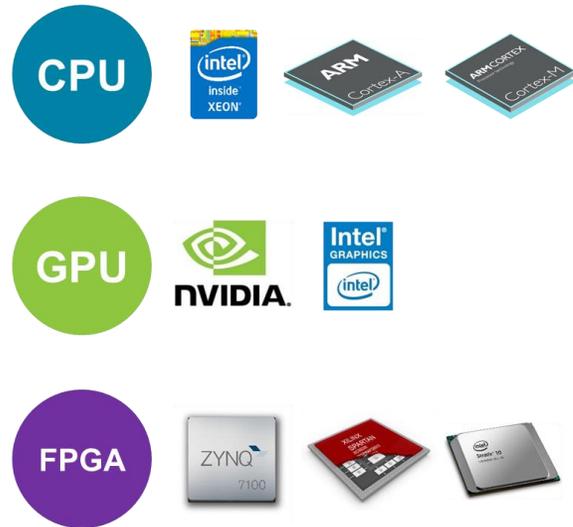
This is two bicyclists

What are the common challenges engineers face using AI with radar and lidar ?

1. Labeling recorded data for AI training is manual and time consuming
2. Little-no recorded data to train models for safety-critical applications
3. Unfamiliarity with AI models for radar and lidar
4. Unclear how to pre-process sensor signals for best results
5. Real-world systems require **deployment of more than AI model**

Challenge

Deploying AI model and application code prototype to a larger system

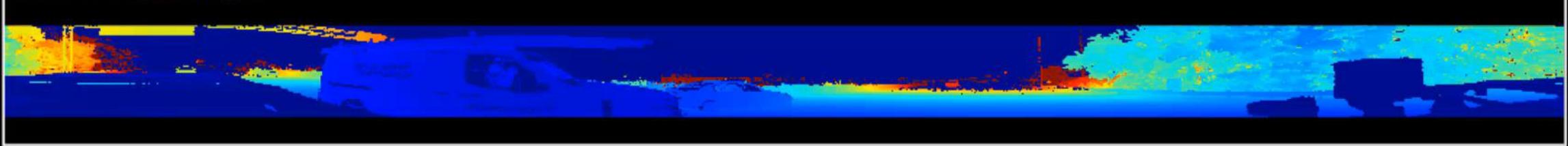


Multiple options for deployment platform
CPU/GPU/FPGA

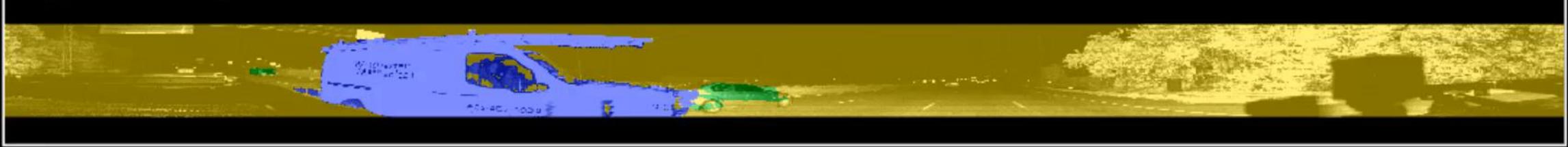
System requires AI model + pre and post processing



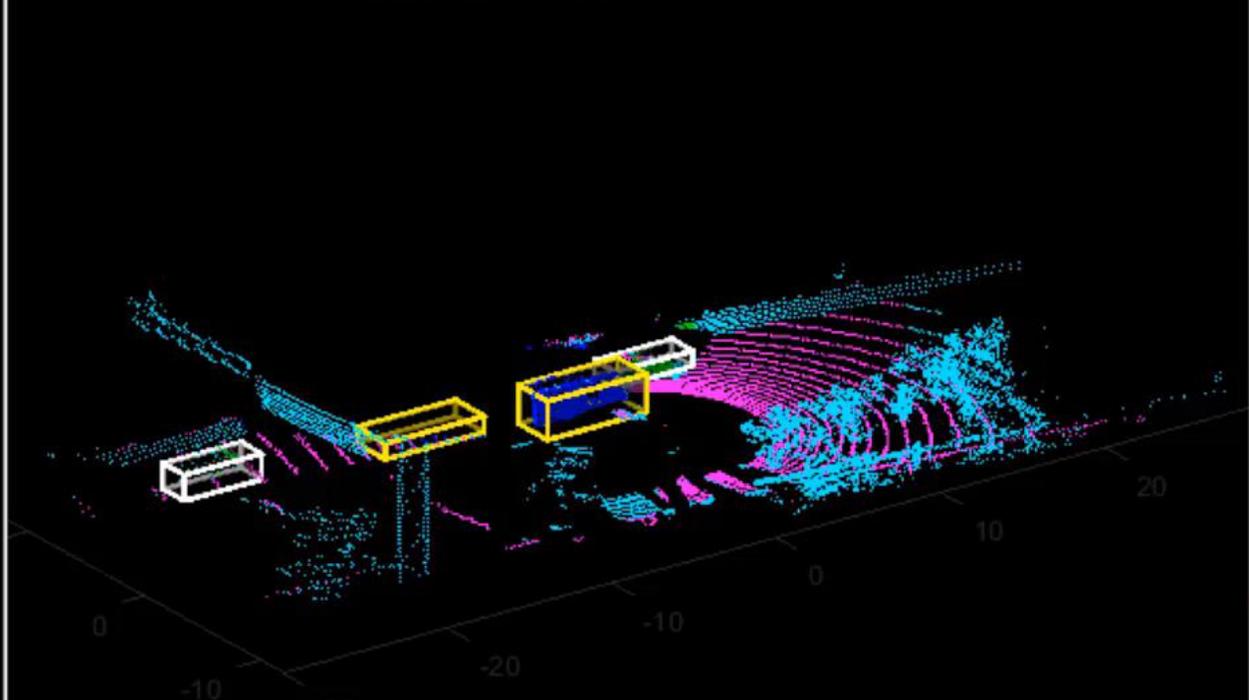
Lidar Range Image



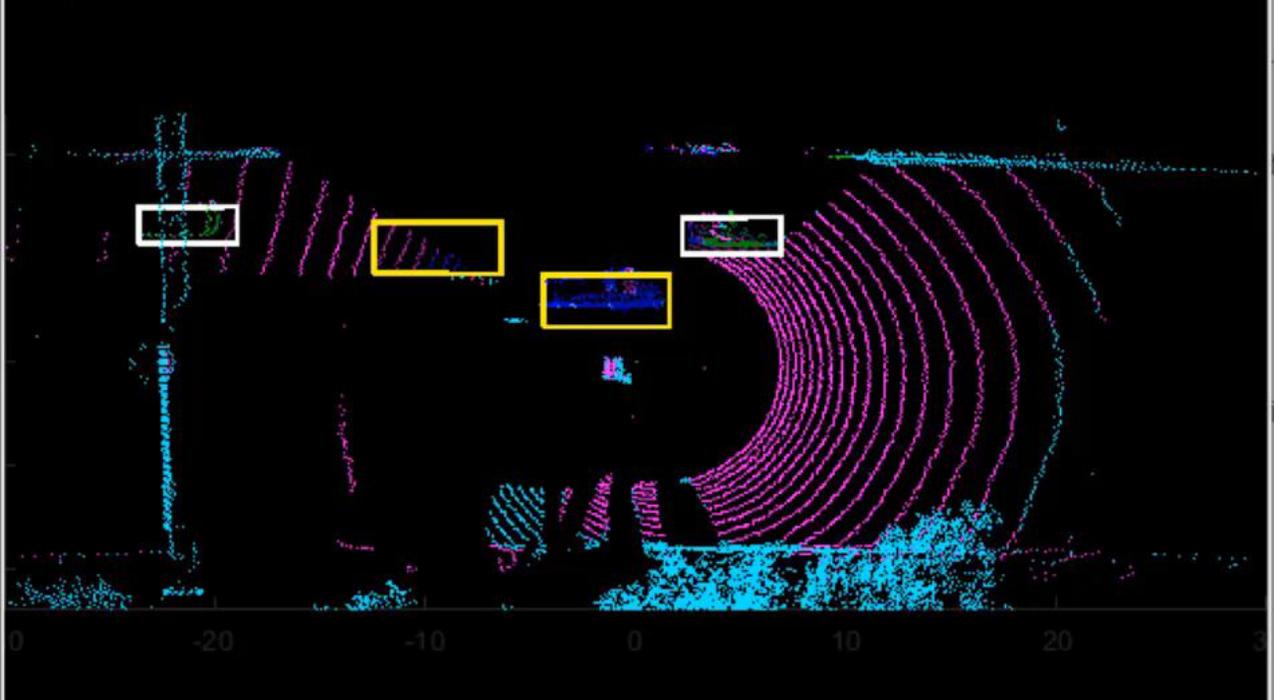
Segmented Image

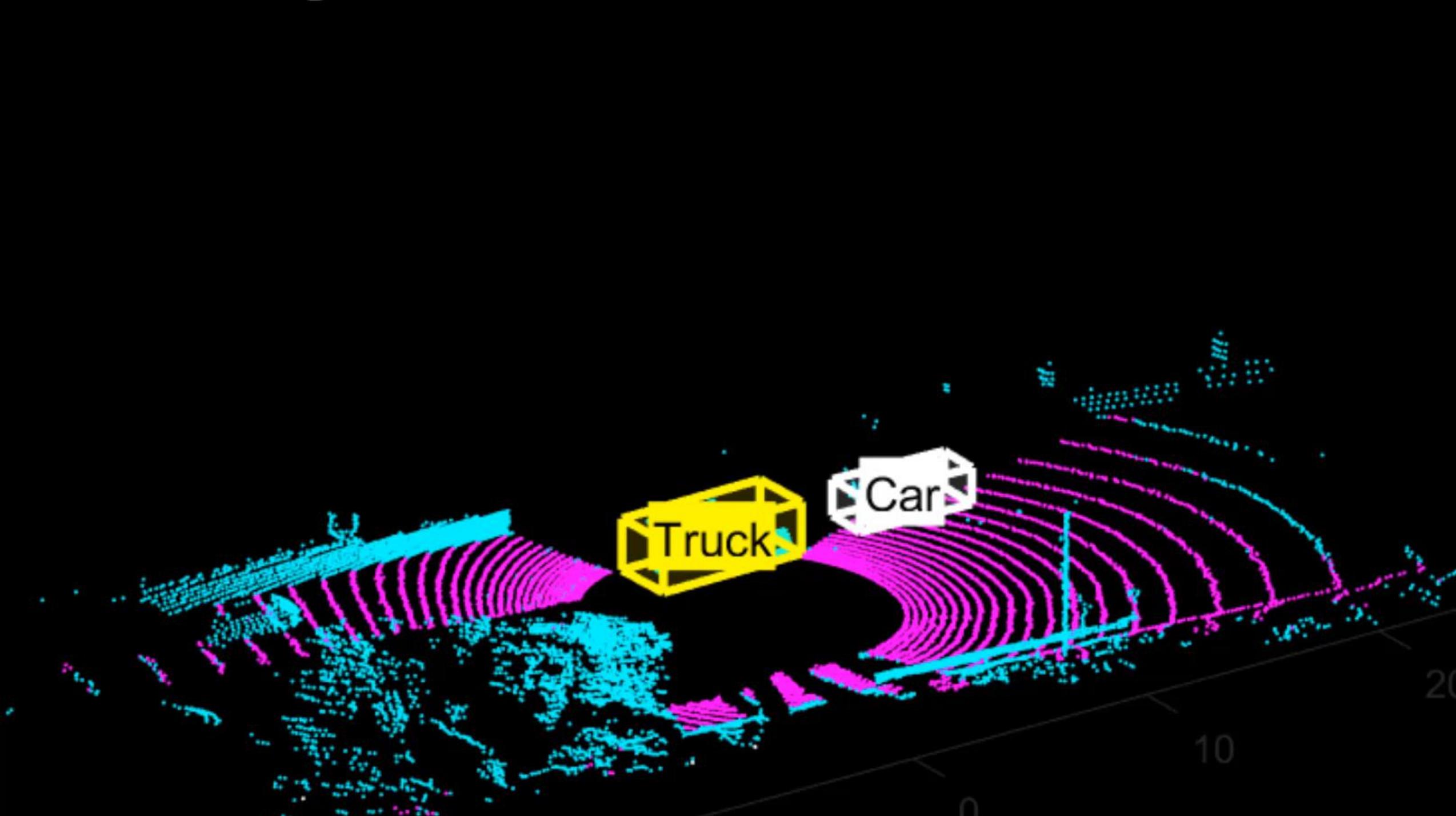


Oriented Bounding Box Detection



Top View





Truck

Car

HOME PLOTS APPS LIVE EDITOR INSERT VIEW

Design App Get More Apps Install App Package App

Curve Fitting PID Tuner Signal Analyzer Image Acquisition MATLAB Coder Distribution Fitter Control System Desi... Control System Tuner Flight Log Analyzer Linear System Analyzer Model Reducer

FILE APPS

C:\Users\mpalakka\OneDrive - MathWorks\Documents\MATLAB\Examples\R2020b\shared_driving_fusion_lidar\TrackVehiclesUsingLidarExample

Live Editor - C:\Users\mpalakka\OneDrive - MathWorks\Documents\Demos\DetectClassifyAndTrackOrientedBoundingBoxInLidarExample\DetectClassifyAndTrackOrientedBoundingBoxInLidarExample.mlx

DetectClassifyAndTrackOrientedBoundingBoxInLidarExample.mlx TrackVehiclesUsingLidarExample.m

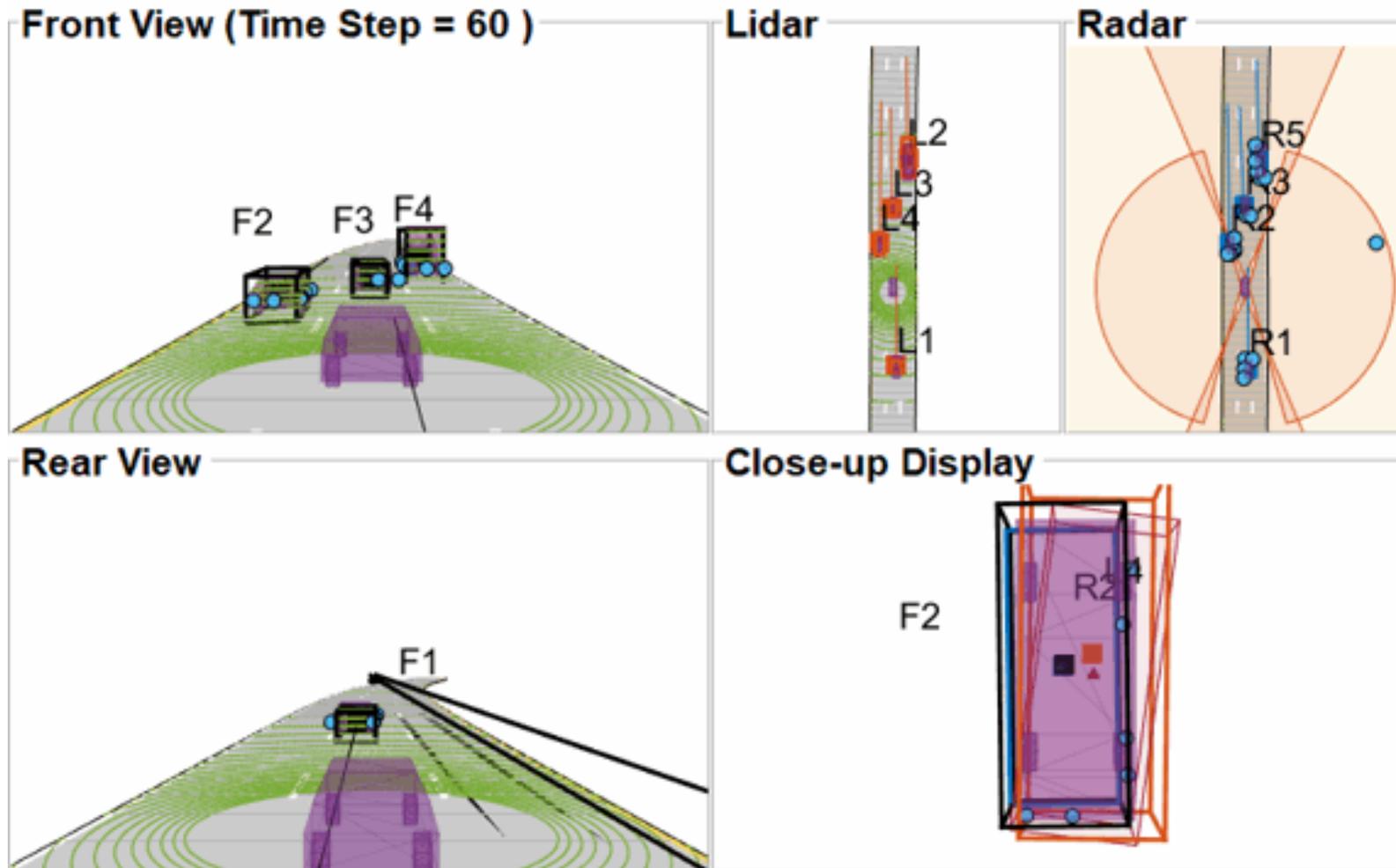
```
150 filterInitFcn = @helperMultiClassInitIMMfilter;
151
152 % A joint probabilistic data association tracker with IMM filter
153 tracker = trackerJPDA('FilterInitializationFcn',filterInitFcn,...
154     'TrackLogic','History',...
155     'AssignmentThreshold',assignmentGate,...
156     'ClutterDensity',Kc,...
157     'ConfirmationThreshold',confThreshold,...
158     'DeletionThreshold',delThreshold,'InitializationThreshold',0);
159
160 allTracks = struct([]);
161 time = 0;
162 dt = 0.1;
163
164 % Define Measurement Noise
165 measNoise = blkdiag(0.25*eye(3),25,eye(3));
166
167 numTracks = zeros(numFrames, 2);
```

The detected objects are assembled as a cell array of `objectDetection` objects using the `helperAssembleDetections` function.

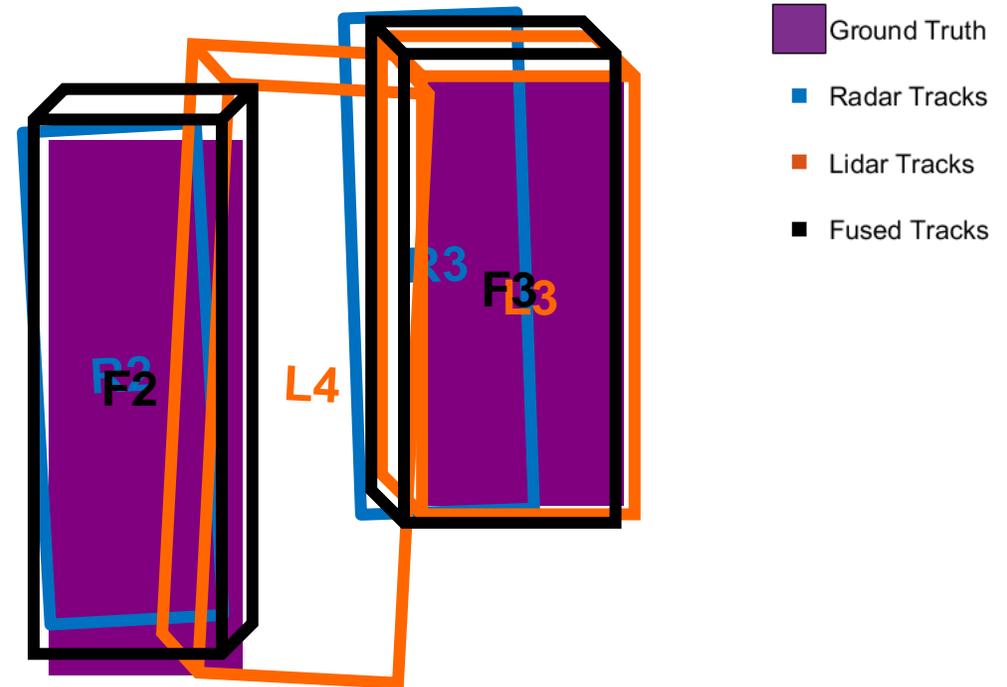
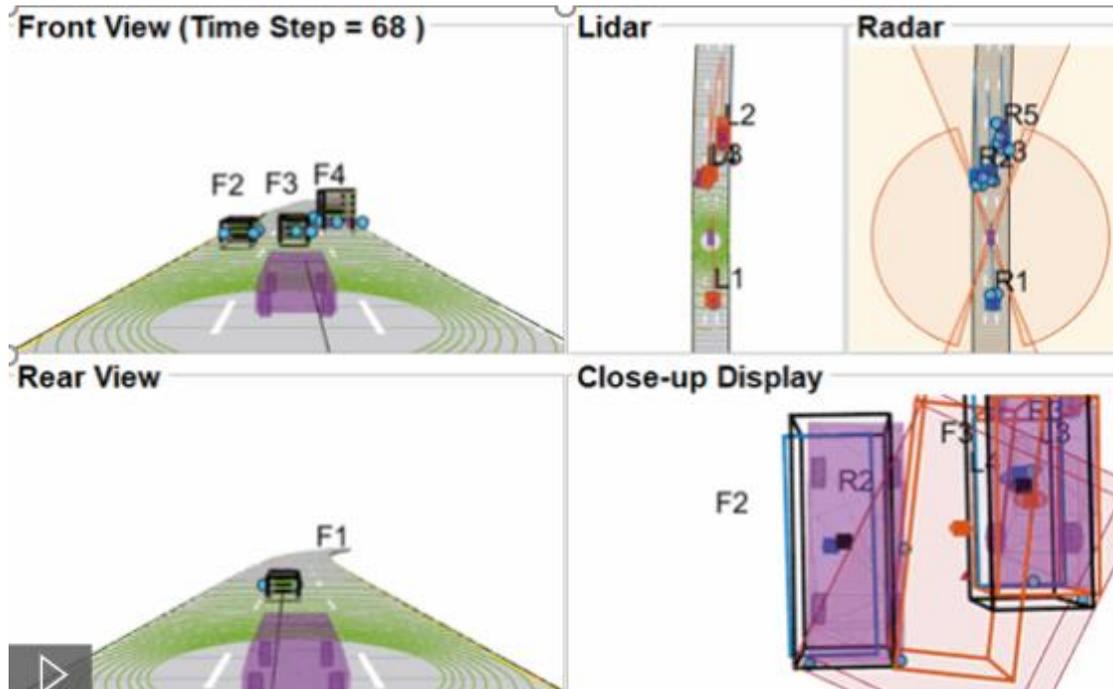
```
168 display = helperLidarObjectDetectionDisplay;
169 initializeDisplay(display);
170
171 for count = 1:numFrames
172     time = time + dt;
173     % Get current data
```

I

We can improve our results when we fuse the two sensors

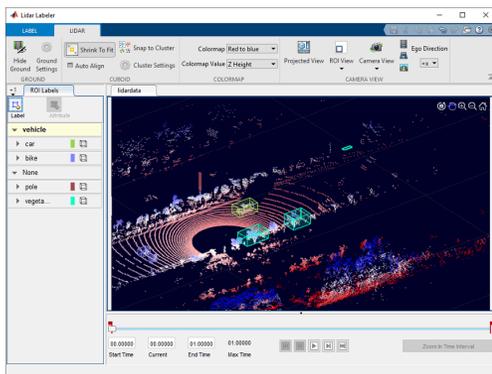


Let's take a closer look ...

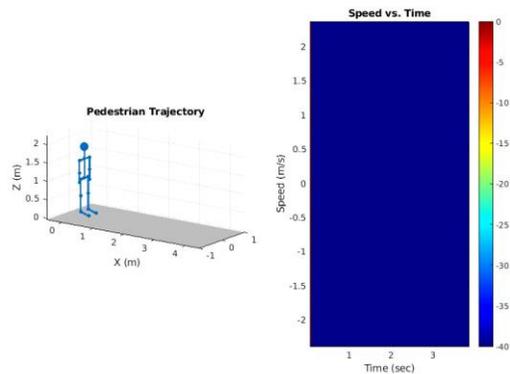


Fused tracks more accurate than individual sensor tracks

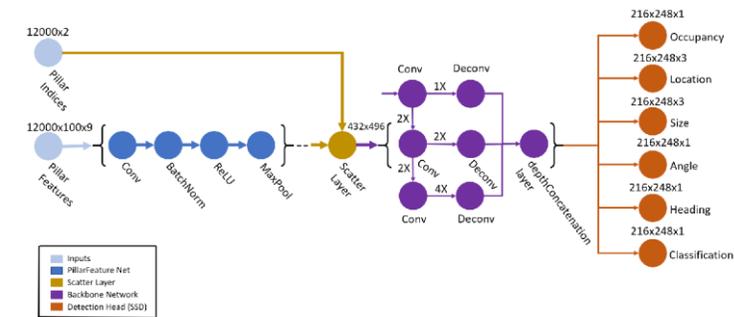
How MATLAB and Simulink help create AI-driven radar and lidar processing systems



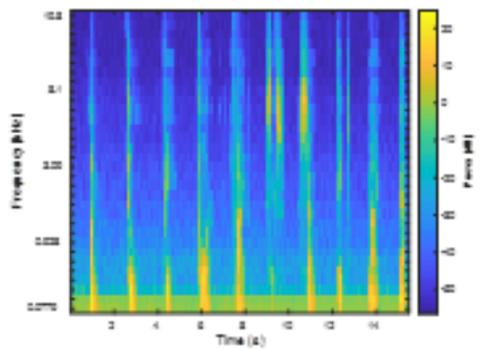
Labeling Automation



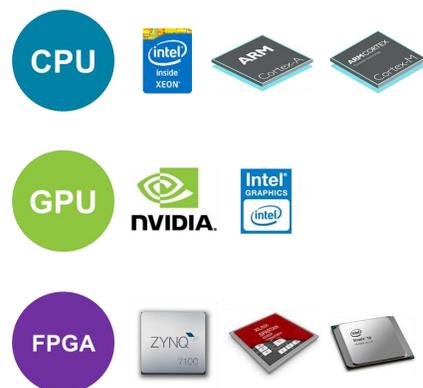
Data Synthesis



AI Workflow
Pre-trained models, training, evaluation, validation



Pre-processing



Full Application Deployment

MATLAB EXPO

2021

감사합니다

