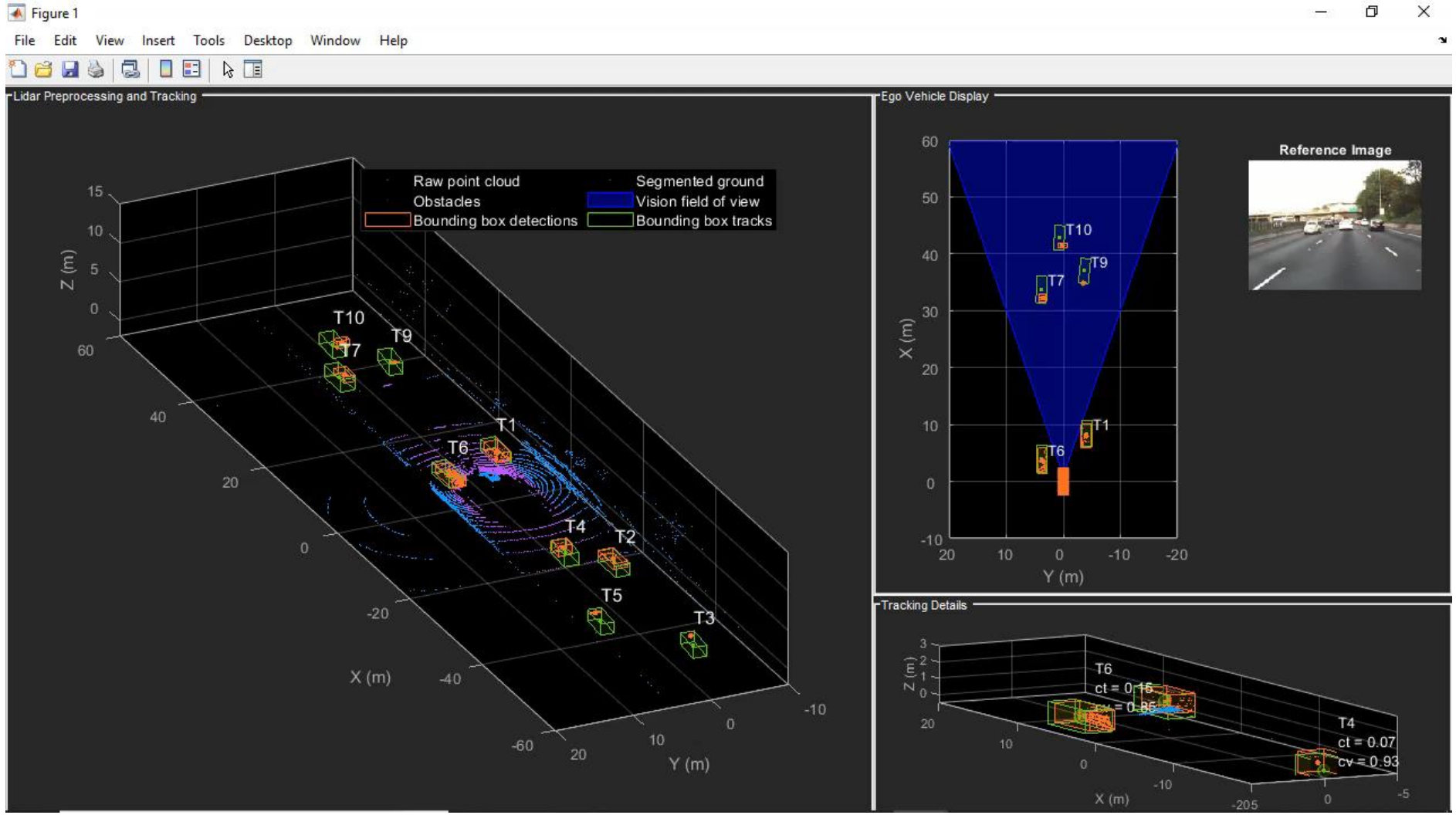
MATLAB EXPO 2019

LiDAR Processing for Automated Driving

Avinash Nehemiah
Principal Product Manager
Deep Learning, Computer Vision, Automated Driving

Complete Workflow for Lidar Processing in Automated Driving



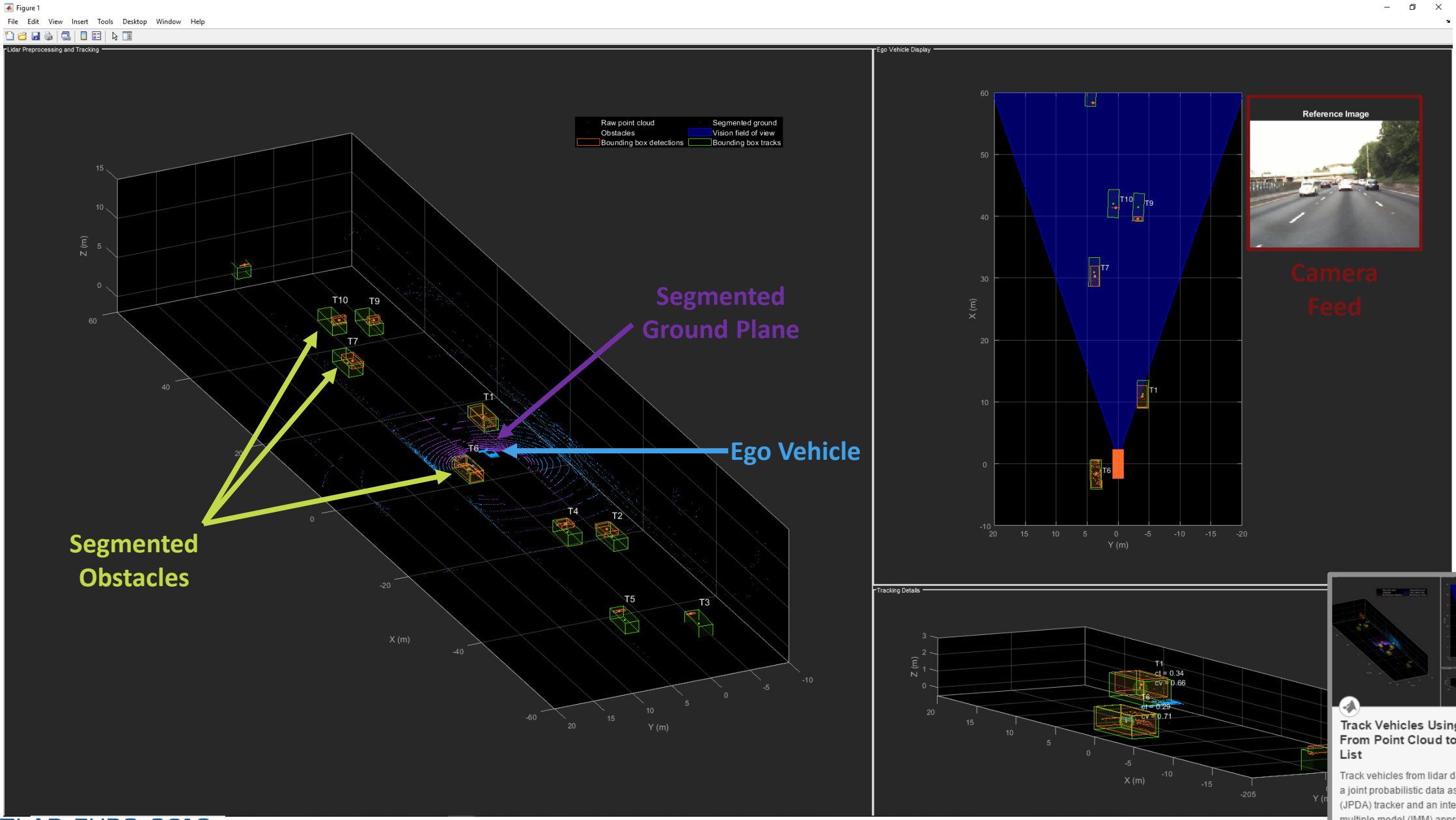


Figure 1
File Edit View Insert Tools Desktop Window Help
Lidar Preprocessing and Tracking
Raw point cloud
Obstacles
Bounding box detections
Segmented ground
Vision field of view
Bounding box tracks
Segmented Ground Plane
Ego Vehicle
Segmented Obstacles
Ego Vehicle Display
Reference Image
Camera Feed
Tracking Details
Track Vehicles Using Lidar: From Point Cloud to Track List
Track vehicles from lidar data using a joint probabilistic data association (JPDA) tracker and an interacting multiple model (IMM) approach.
Open Script

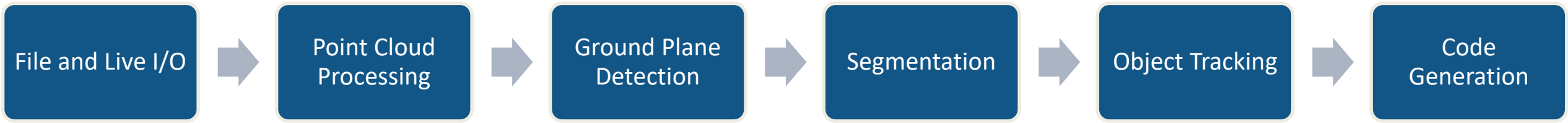
Why is lidar an essential sensor for automated driving ?

1. Accurate depth measurement (currently a function of radar)
2. 360-degrees of visibility (require multiple calibrated sensors to achieve)
3. Dense data to detect and classify objects (currently function of camera)

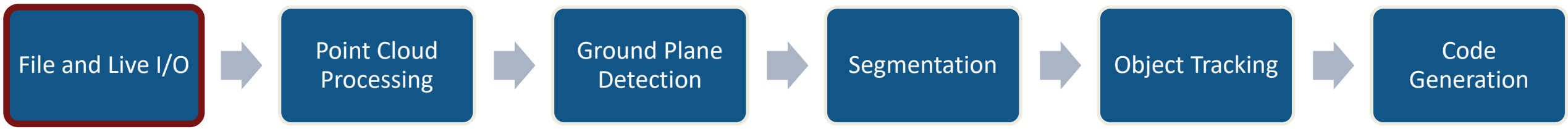
**Automated driving use camera, radar and lidar as complementary sensors*

** Robotics application use lidar alone*

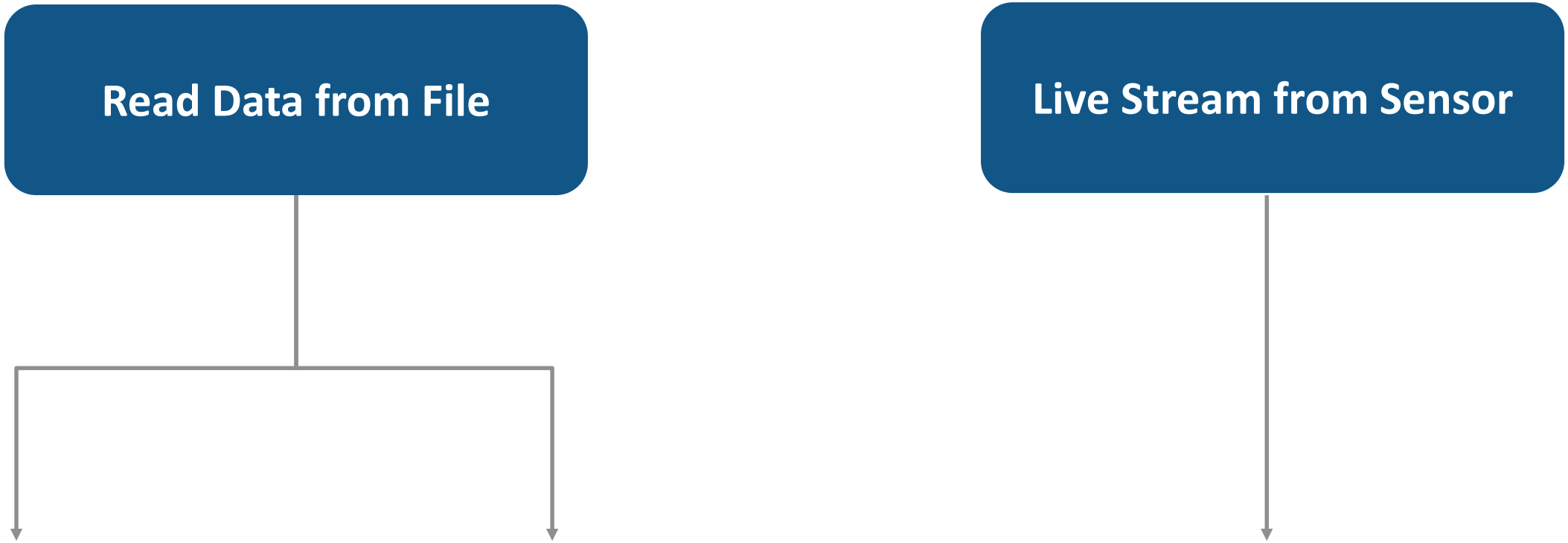
Talk Outline



Talk Outline

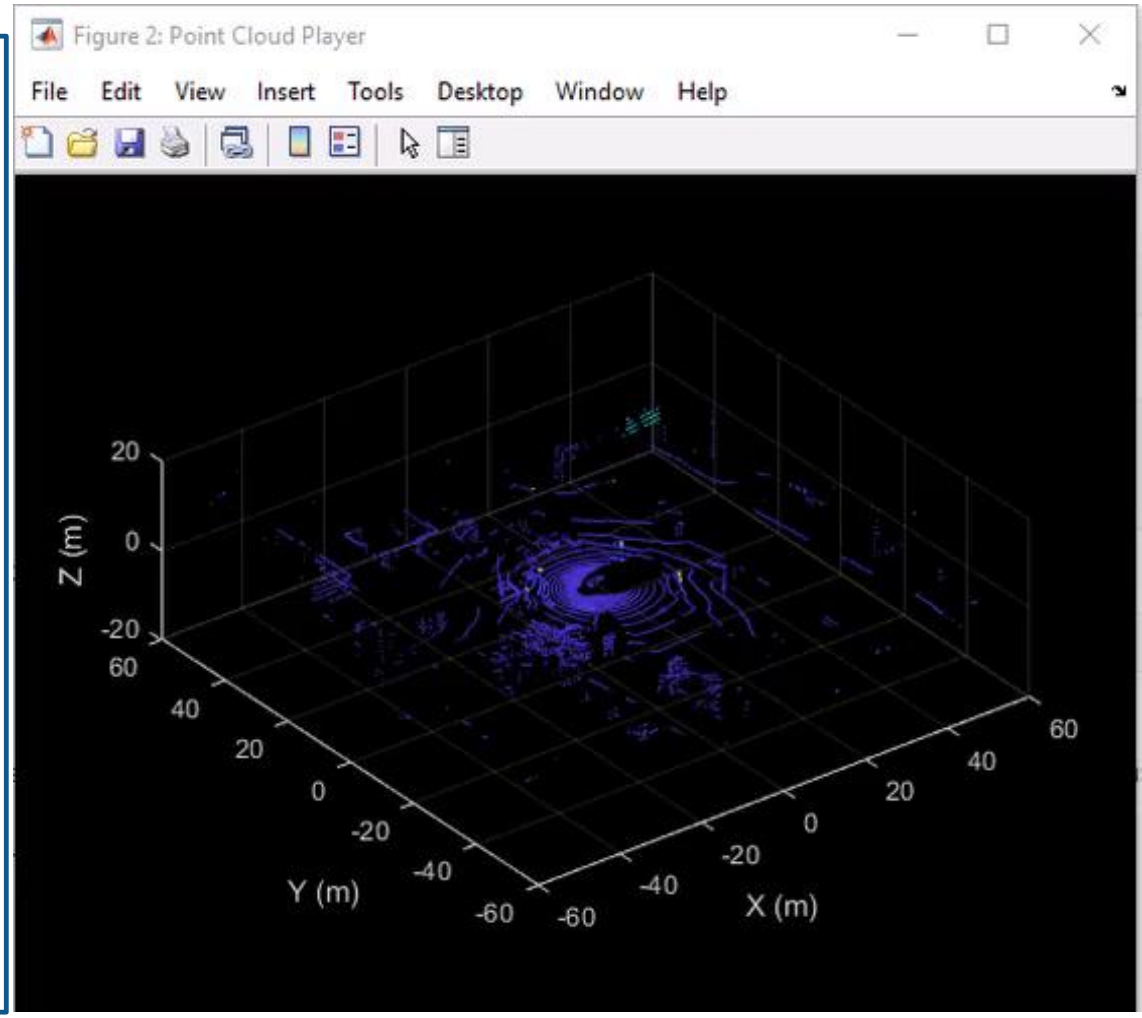


Reading Lidar Data from File or Sensor



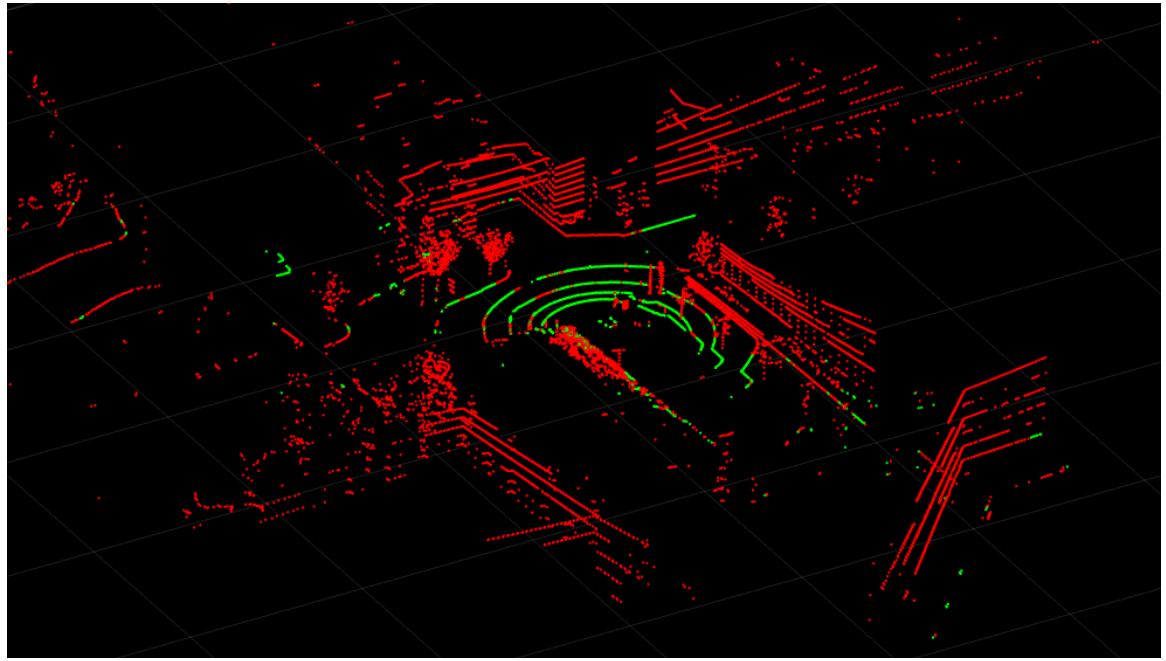
Read Input Sequence from PCAP File

```
% Read PCAP file recorded from HDL32E model sensor
veloReader =
velodyneFileReader('lidarData.pcap','HDL32E');


% Read from file and display
player = pcplayer(xlimits,ylimits,zlimits);
while(hasFrame(veloReader))
    ptCloudObj = readFrame(veloReader);

view(player,ptCloudObj.Location,ptCloudObj.Intensity
);
end
```



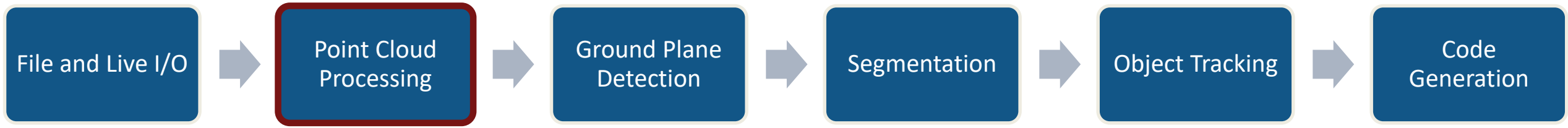
Live Acquisition from Velodyne LiDAR® Sensors

- Supported models:
  - Ultra Puck (VLP-32C)
  - Puck (VLP-16)
    - Puck Hi-Res
    - Puck LITE
    - Puck
  - HDL-32E



- Stream point clouds to data buffer and read into MATLAB on demand

Talk Outline



`pointCloud` Object: Lidar Data Container

`pointCloud` Object: Primary container for Lidar data

```
ptCloudObj =

  pointCloud with properties:

     Location: [32×1084×3 single]
        Color: []
       Normal: []
    Intensity: [32×1084 single]
        Count: 34688
      XLimits: [-80.0677 93.4626]
      YLimits: [-82.5912 84.3181]
      ZLimits: [-8.7008 10.6333]
```

[x,y,x] coordinates

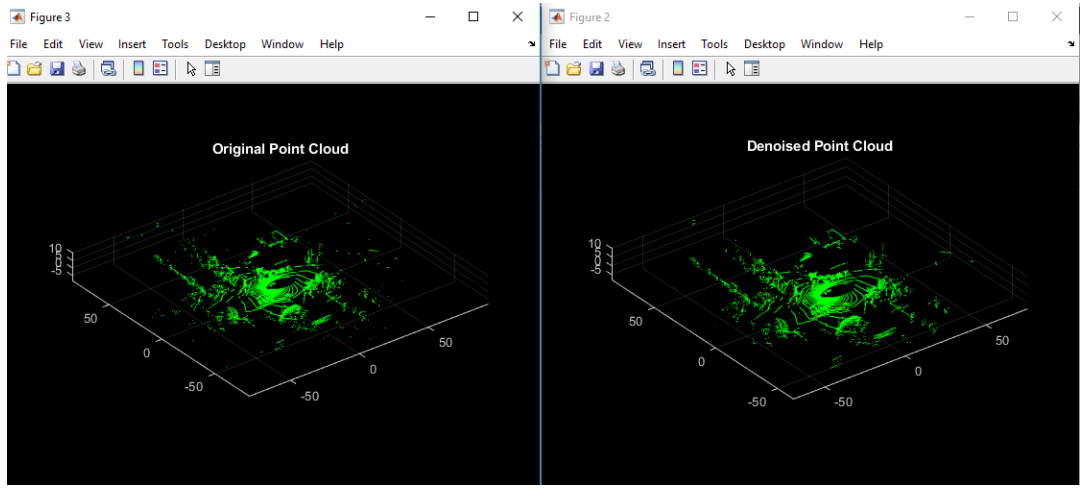
Intensity of return

Methods of **`pointCloud`** object

findNearestNeighbors	Find nearest neighbors of a point in point cloud
findNeighborsInRadius	Find neighbors within a radius of a point in the point cloud
findPointsInROI	Find points within a region of interest in the point cloud
removeInvalidPoints	Remove invalid points from point cloud
select	Select points in point cloud

Point Cloud Preprocessing

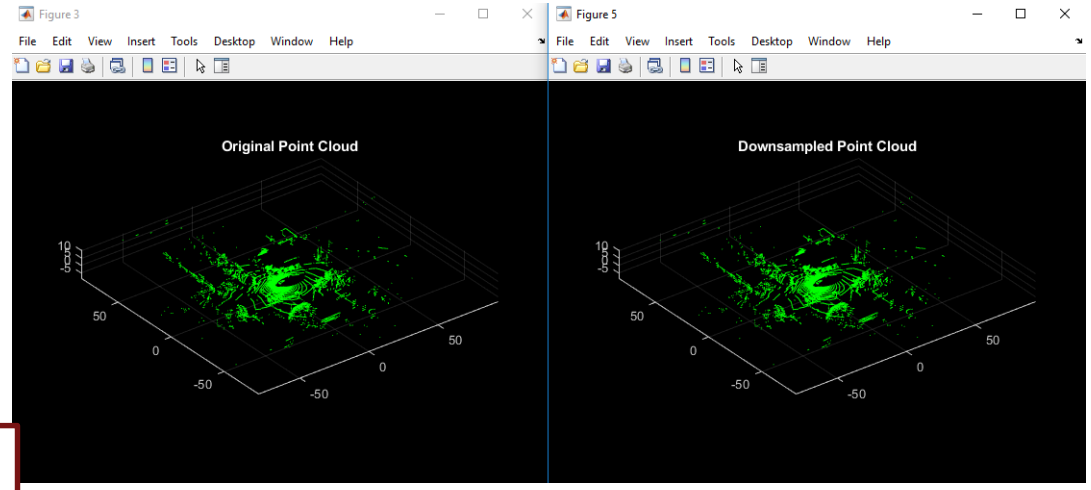
`pcdenoise`
Denoise point cloud



Original: ~ 34,000 points

Denoised: ~ 21,000 points

`pcdownsample`
Down sample point cloud



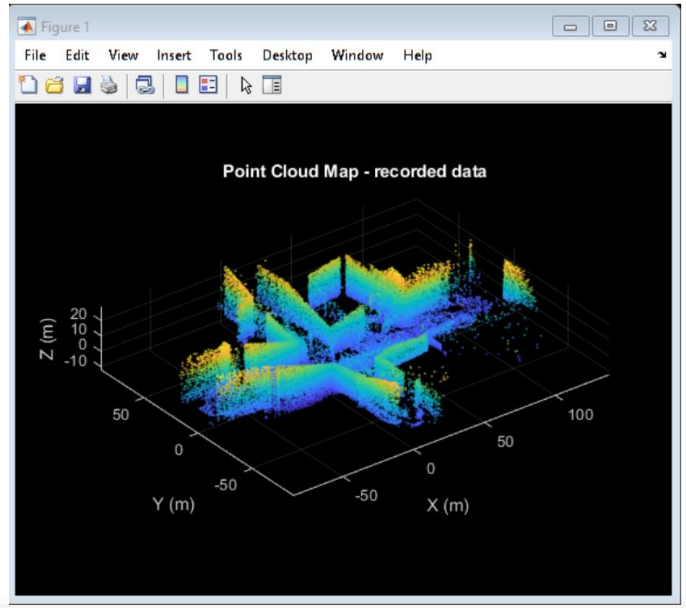
Original: ~ 34,000 points

Down sampled: ~ 14,000 points

Reduce # of points without compromising structure of scene

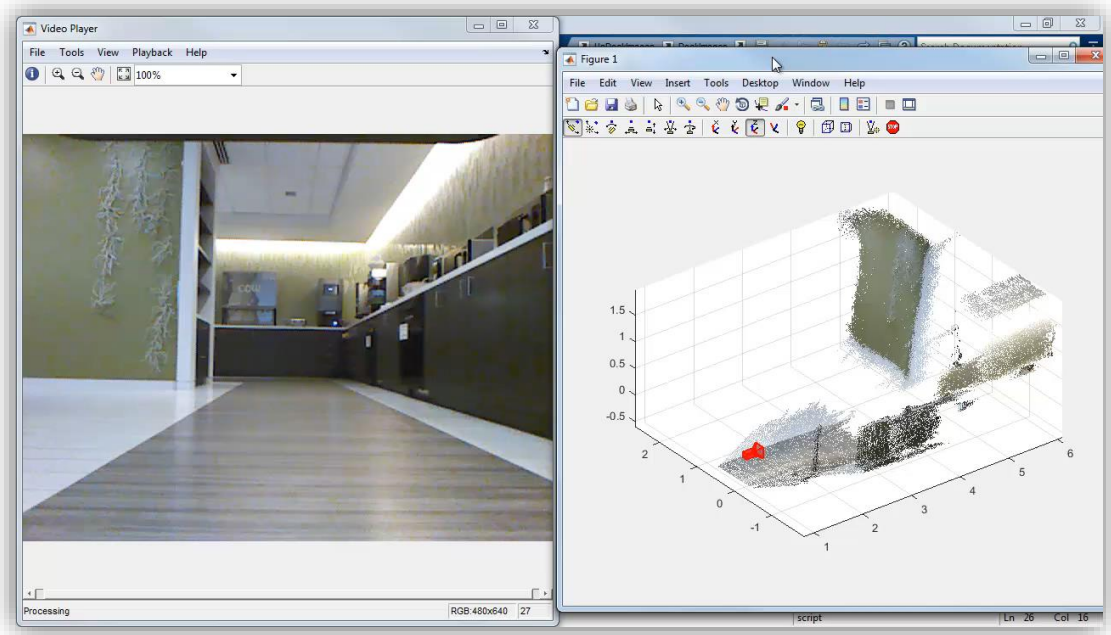
Point Cloud Registration and Stitching

Create 3D maps with series of Lidar point clouds

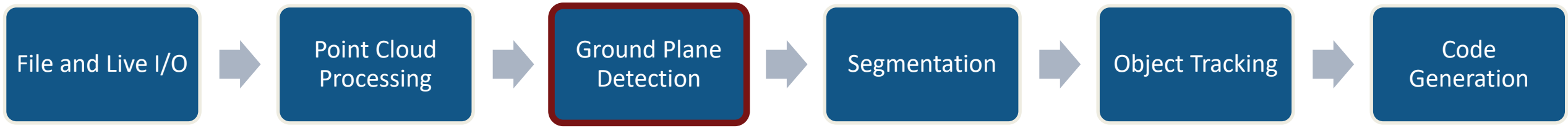


Registration techniques supported:

- Iterative closest point (ICP)
- Coherent point drift (CPD)
- Normal distributions transform (NDT)
  - Popular for automated driving applications



Talk Outline



Segmenting Ground Plane in Point Cloud Data

`pcfitplane`

General Geometric Model Fitting

- *Fit model of plane to points*
- *More general - designed for ALL point clouds (Lidar and RGB-D)*

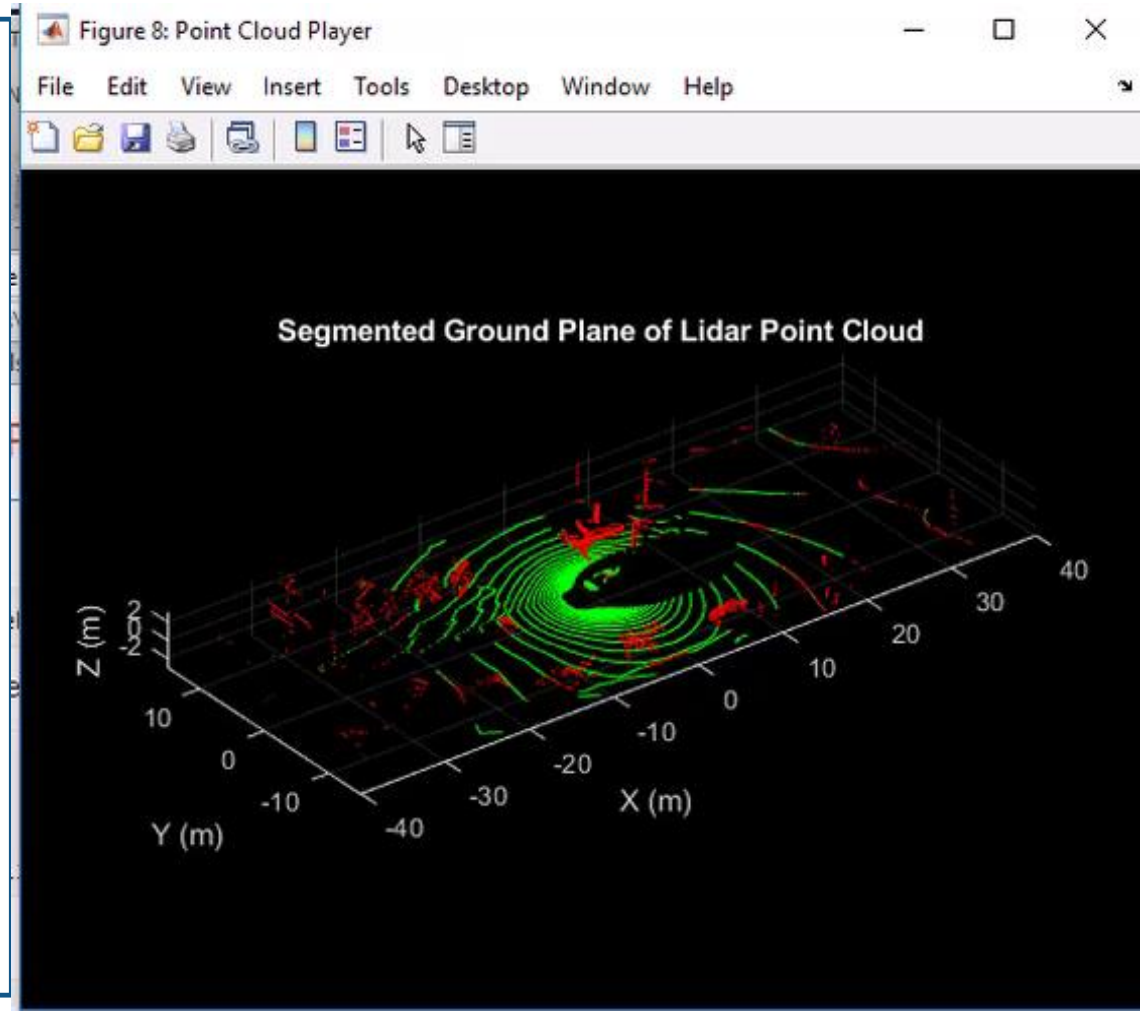
`segmentGroundFromLidarData`

Specialized Ground Plane Segmentation

- *Specialized for organized point clouds*
- *Faster execution*

Ground Plane Segmentation

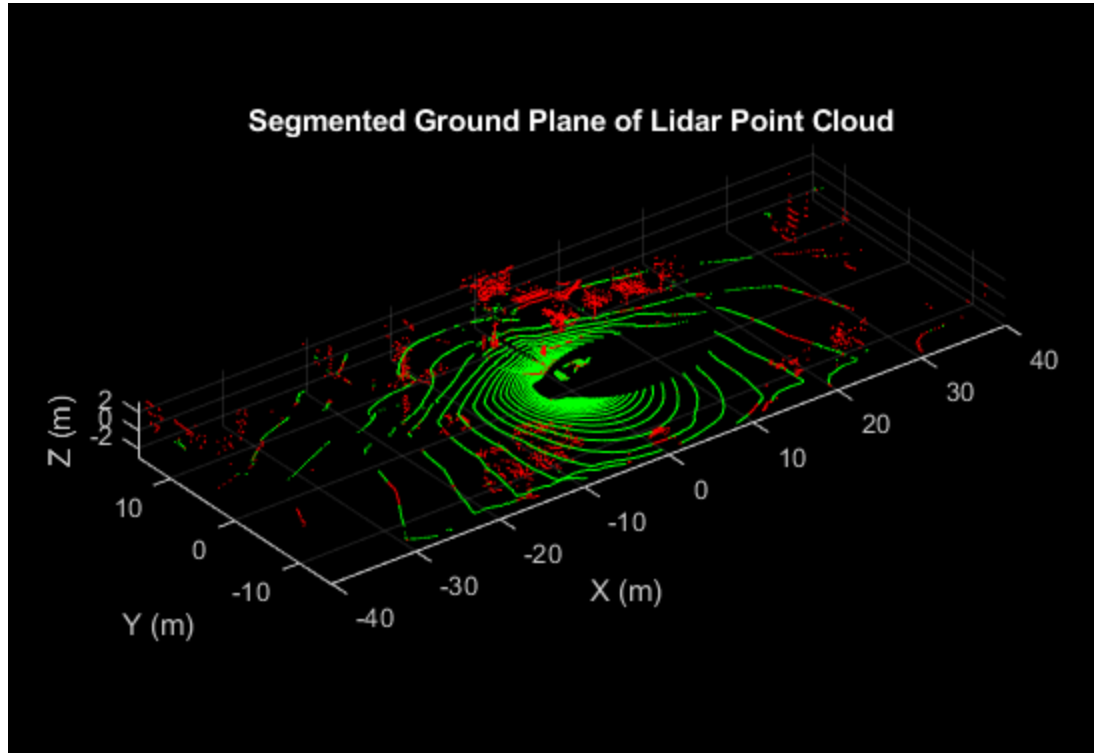
```
% Find the ground points.
 groundPtsIdx = segmentGroundFromLidarData(ptCloud);

% Map color ground points to green.
colorLabels(groundPtsIdx (:)) = greenIdx;

% Map color nonground points to red.
colorLabels(~groundPtsIdx (:)) = redIdx;
```



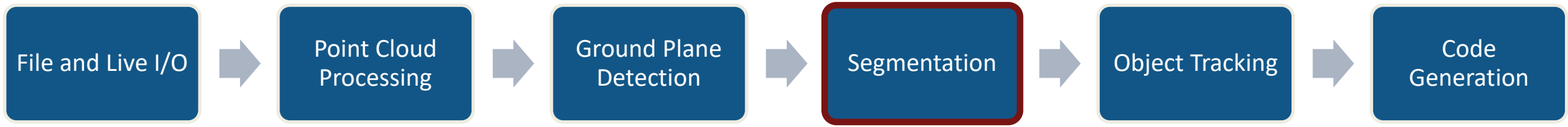
Why is it important to detect the ground plane ?

1. Helps estimate drivable path:
   - Objects that are NOT ground are obstacles

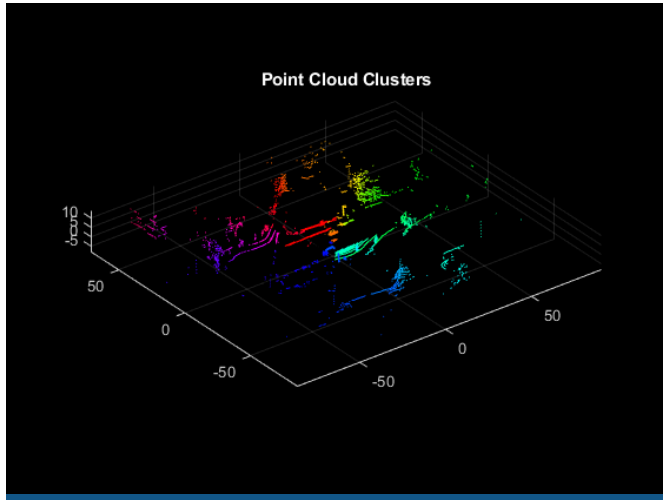
2. Improve processing time
   - Fewer points for downstream processing



Talk Outline

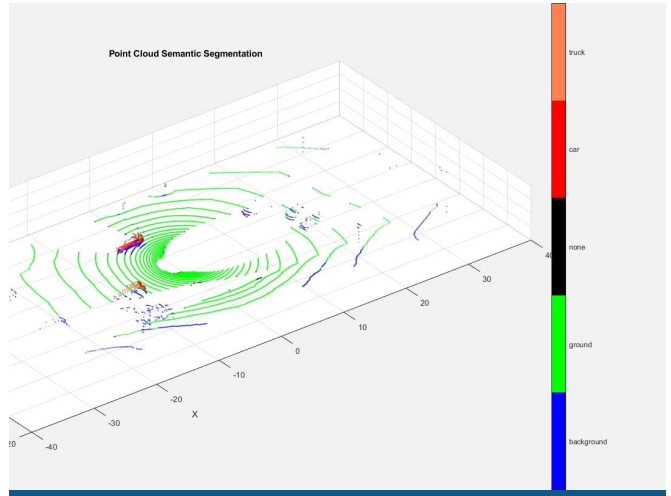


Lidar Segmentation



Clustering

- Assigns labels to co-located points
- Uses distance metric to cluster points
- No object classification
- Fast execution



Semantic Segmentation (Deep Learning)

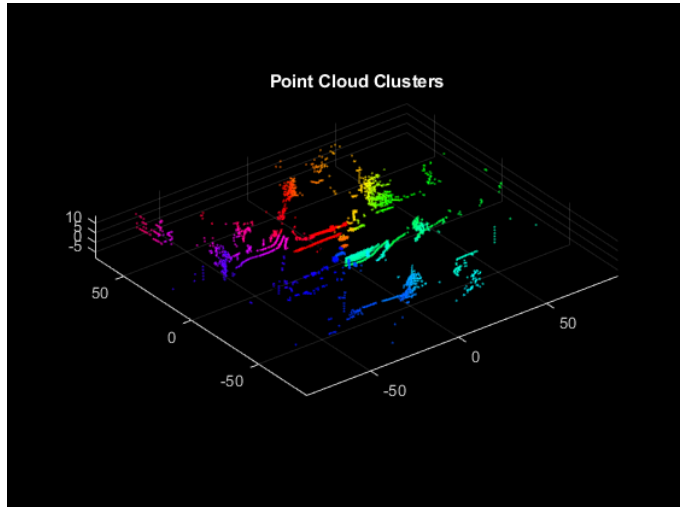
- Deep learning to classify each point
- Longer execution time

Point Cloud Segmentation with Clustering

pcsegdist

Segments point cloud using Euclidean distance (generic)

```
distThreshold = 0.5;
[labels,numClusters] =
pcsegdist(ptCloudWithoutGround,distThreshold);
```



segmentLidar

- Specialized for Lidar (organized point cloud)
- Real-time execution

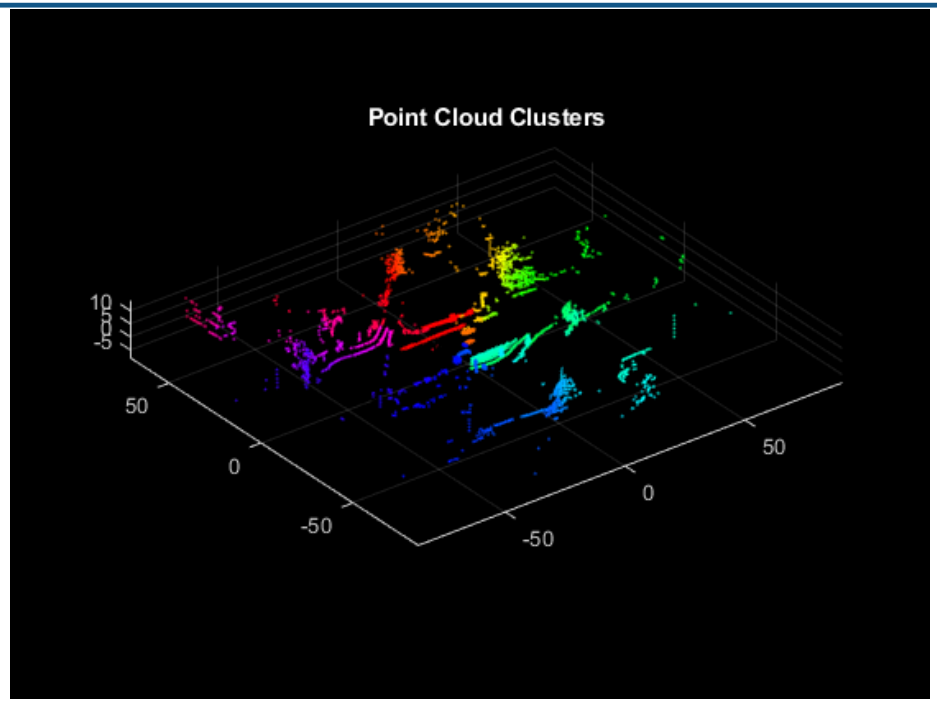
```
distThreshold = 0.5;
[labels,numClusters] =
pcsegdist(ptCloudWithoutGround,distThreshold);
```

Lidar Segmentation with Clustering

pcsegdist

Segments point cloud using Euclidean distance (generic)

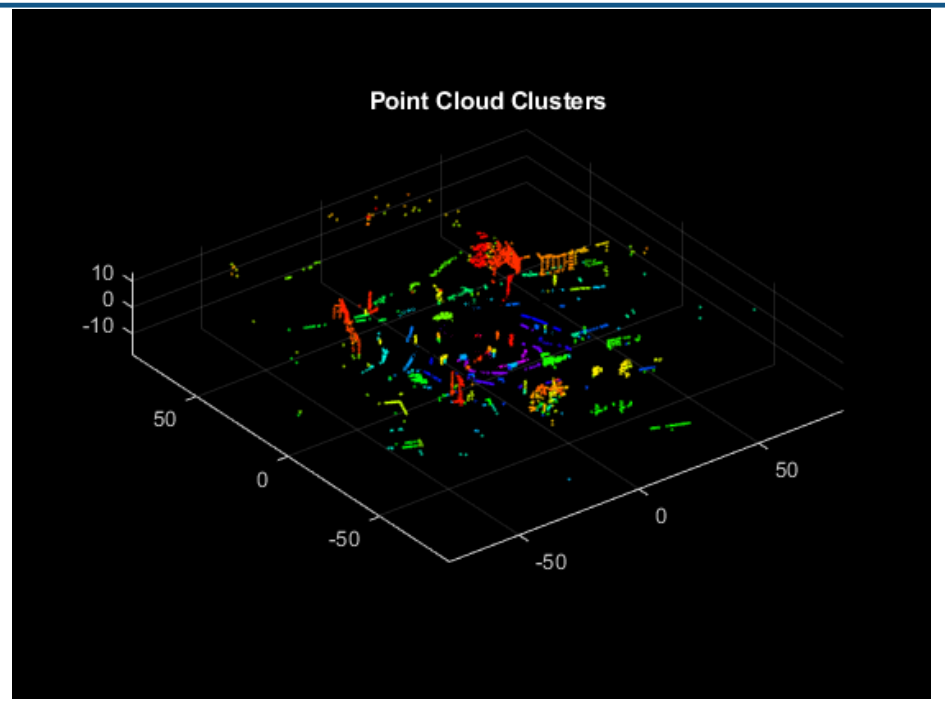
```
distThreshold = 0.5;
[labels,numClusters] =
pcsegdist(ptCloud,distThreshold);
```



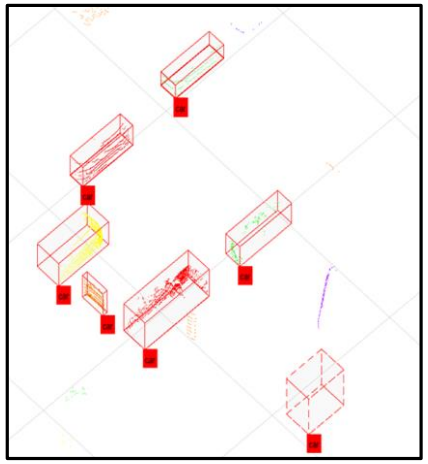
segmentLidar

- Specialized for Lidar (organized)
- Real-time execution

```
distThreshold = 0.5;
[labels,numClusters] =
segmentLidarData(ptCloud,distThreshold);
```

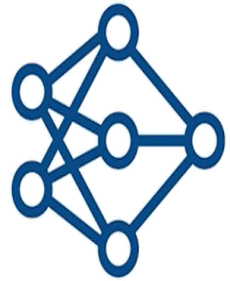


Semantic Segmentation of Lidar using Deep Learning



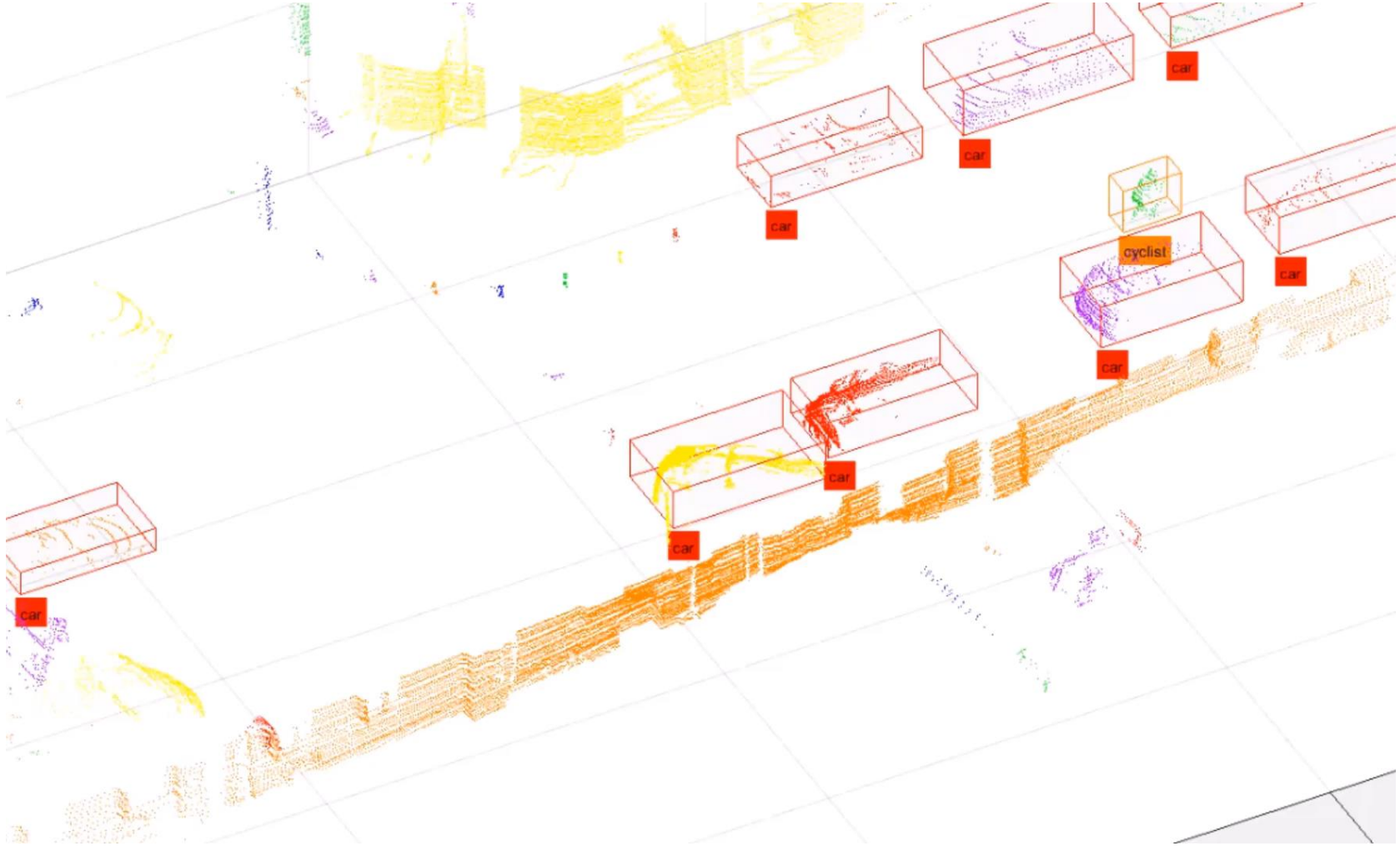
Label Data

Train



Trained DNN

Ground Truth Labeling of Lidar Data



Organize Data for Training

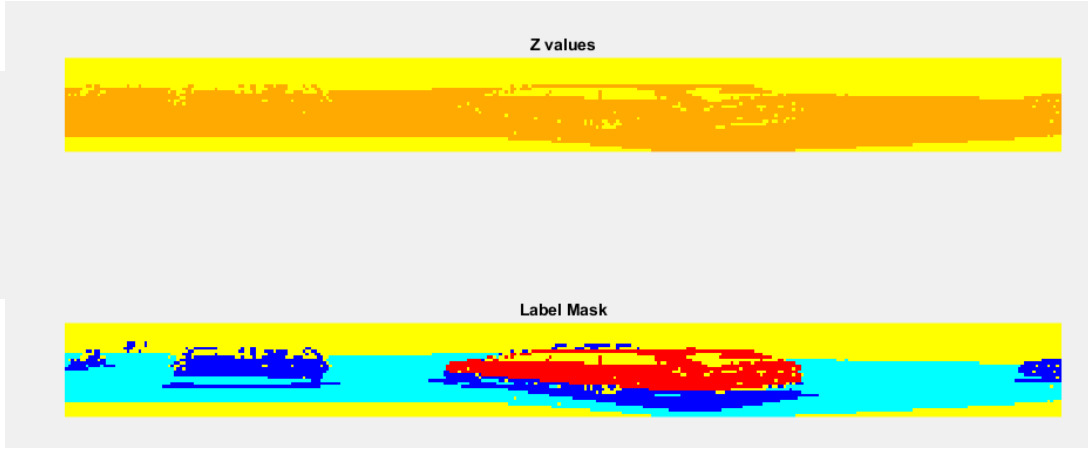
Raw Point Cloud Data

32x1085 double

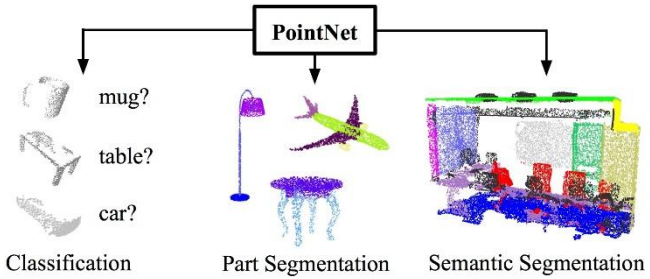
	1	2	3	4
1	NaN	0.1761	0.2727	0.3
2	0.0770	0.1932	0.3064	0.4
3	NaN	NaN	0.2822	0.4
4	0.0834	0.2183	0.2875	0.3
5	NaN	NaN	NaN	N
6	0.0791	0.2409	0.4033	0.5
7	0.0745	0.2363	0.3987	0.5
8	0.0651	0.2266	0.3885	0.5
9	0.0559	0.2177	0.3799	0.5
10	0.0512	0.2130	0.3749	0.5
11	0.0419	0.2037	0.3657	0.5

Project to 2D

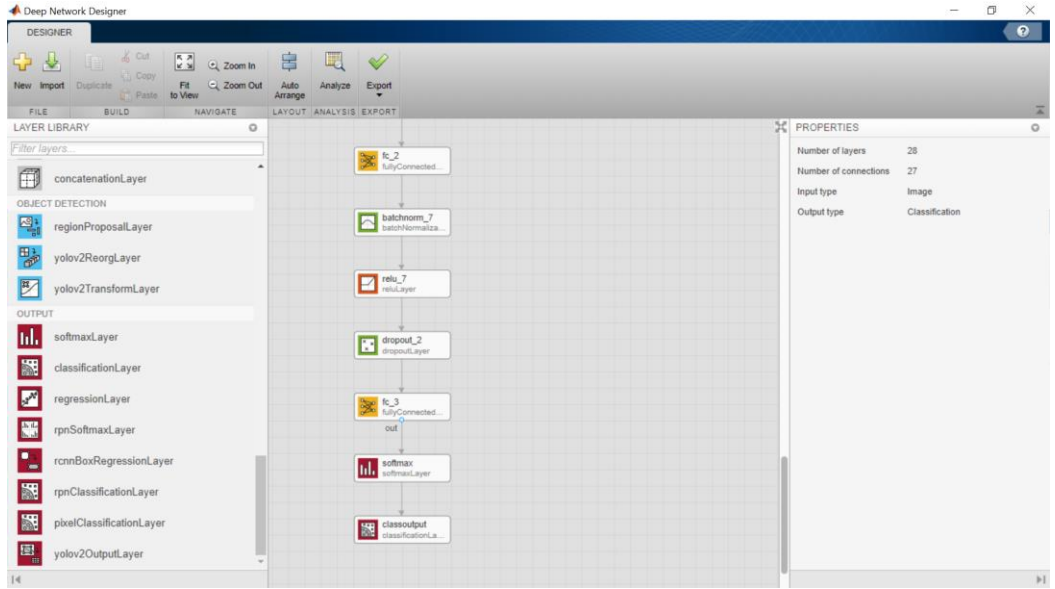
Ground Truth Labels Transformed to Label Mask



Create Network Architecture

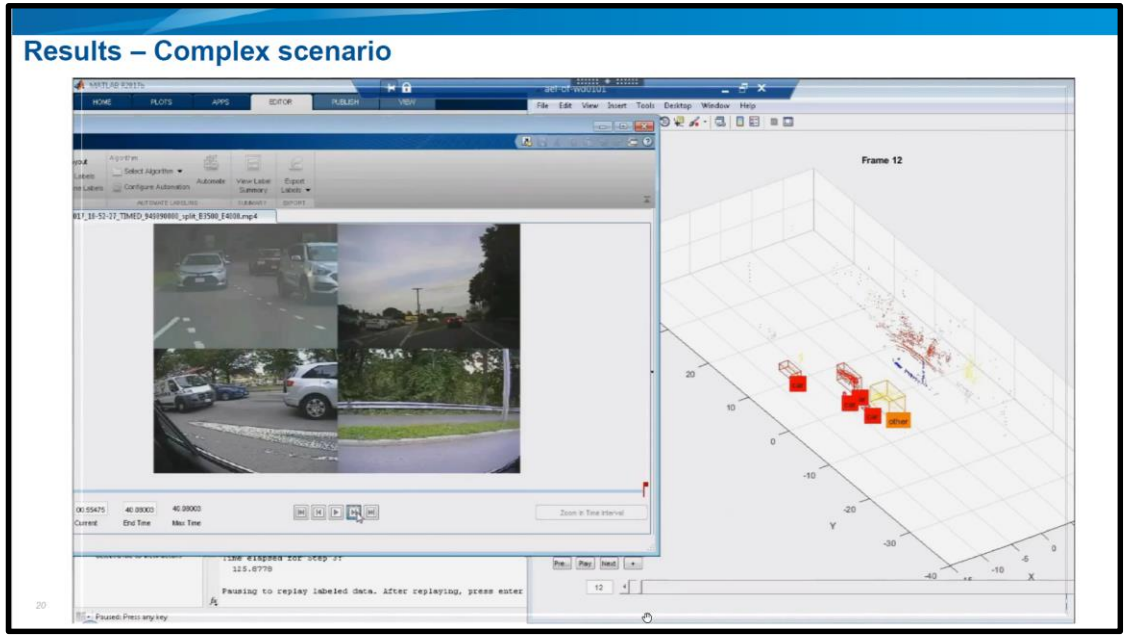


Create Network with App



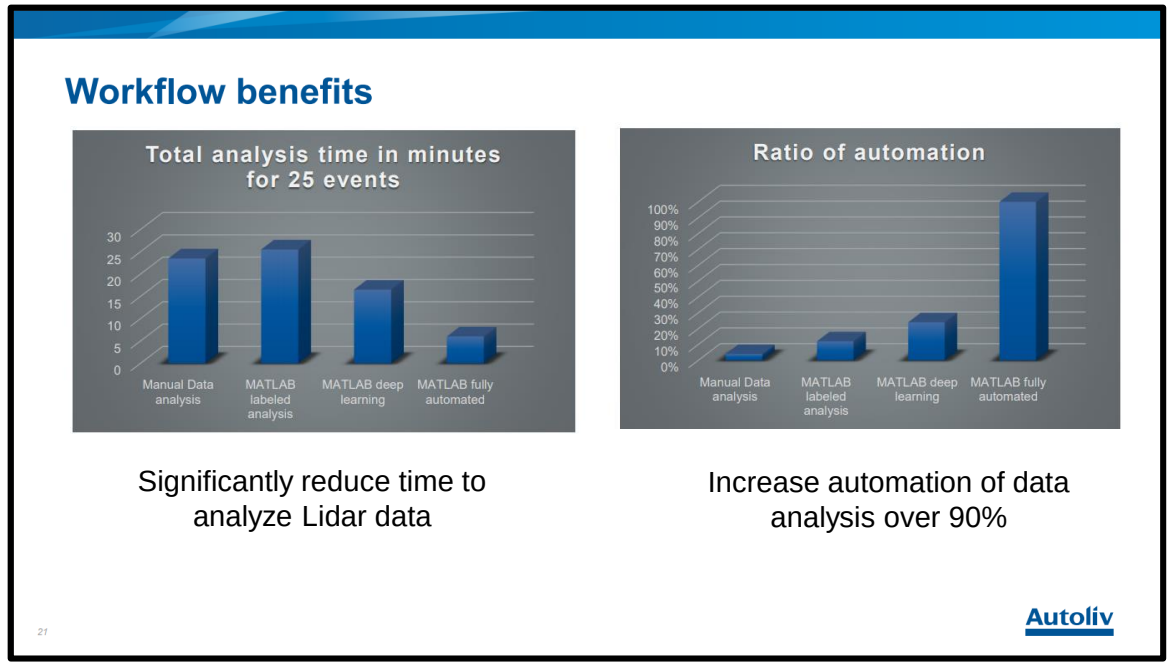
Citation
PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation
Charles R. Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas

Veoneer (Autoliv)
Lidar Based Sensor Verification



Labeling of Lidar for verification of Radar-based automated driving system

- **MATLAB use in project:**
  - Ground truth labeling of Lidar
  - Deep learning for Lidar object detection



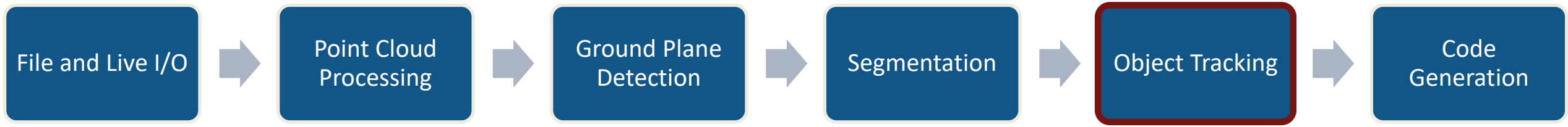
Significantly reduce time to analyze Lidar data

Increase automation of data analysis over 90%

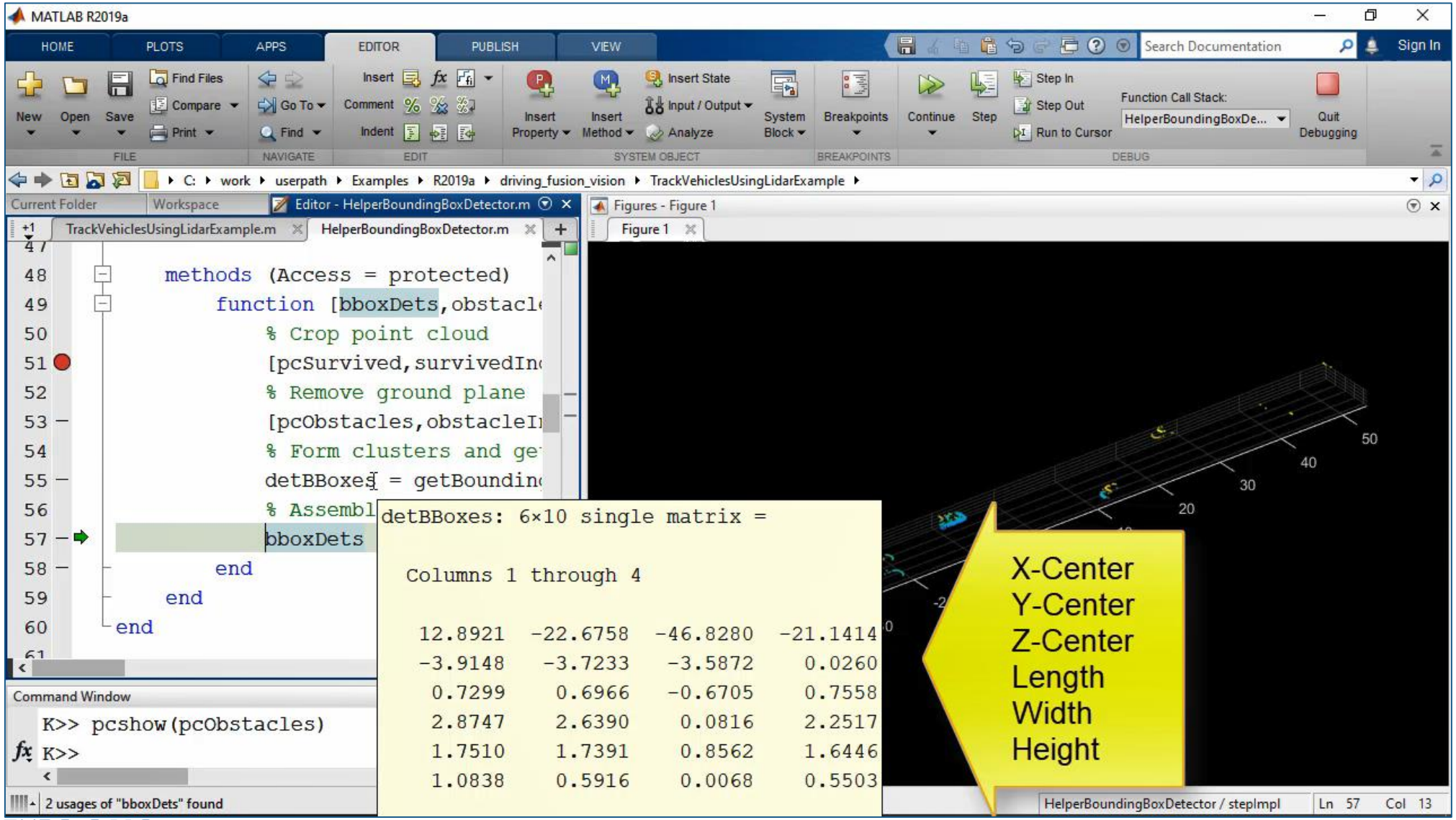
Link to video

LiDAR Based Sensor Verification
Nathan Kurtz (Autoliv), Balakumar Ragunathan (Autoliv) , Arvind Jayaraman (MathWorks)
Video: Automated LiDAR Point-Cloud Annotation for Sensor Verification

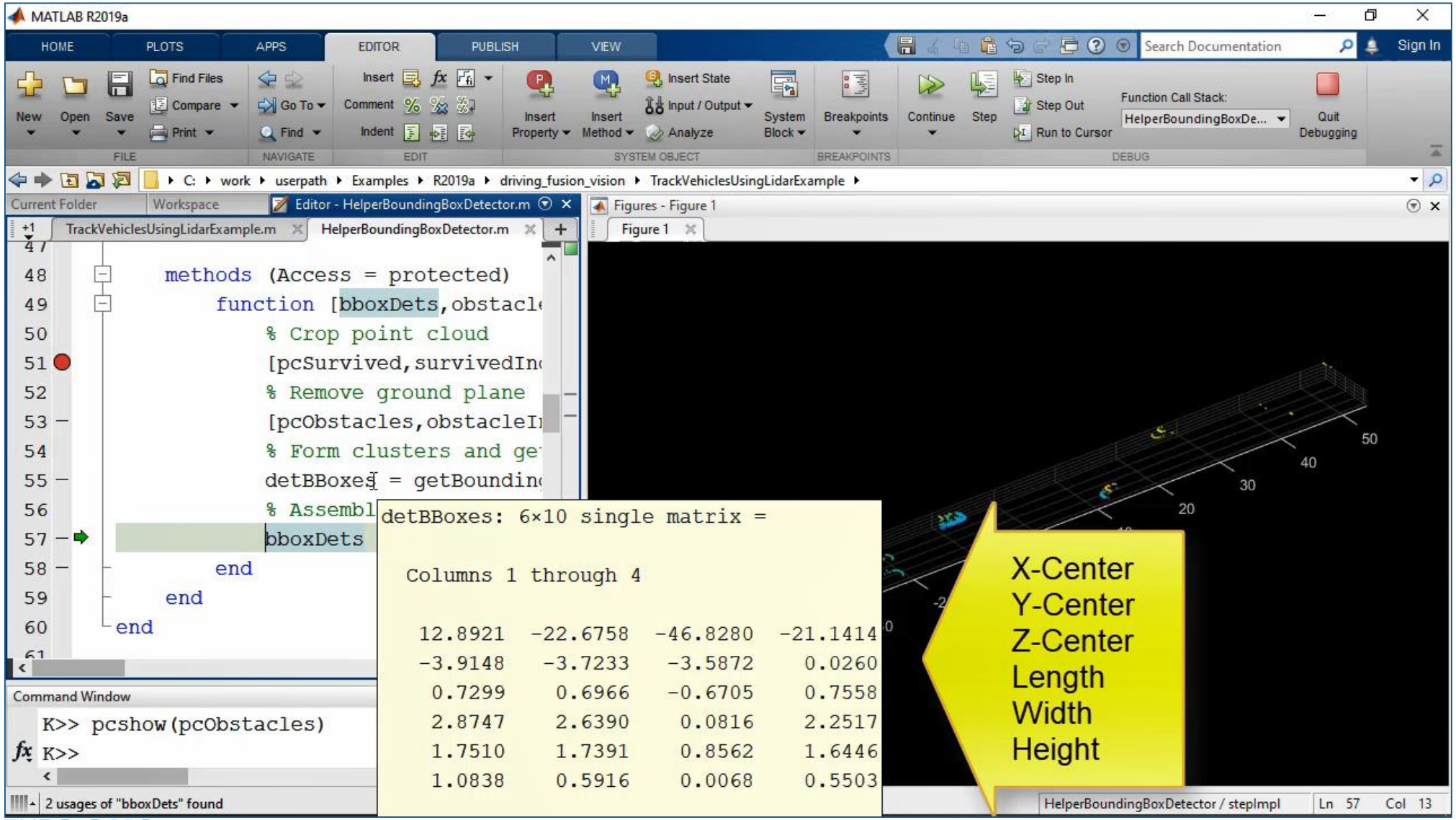
Talk Outline



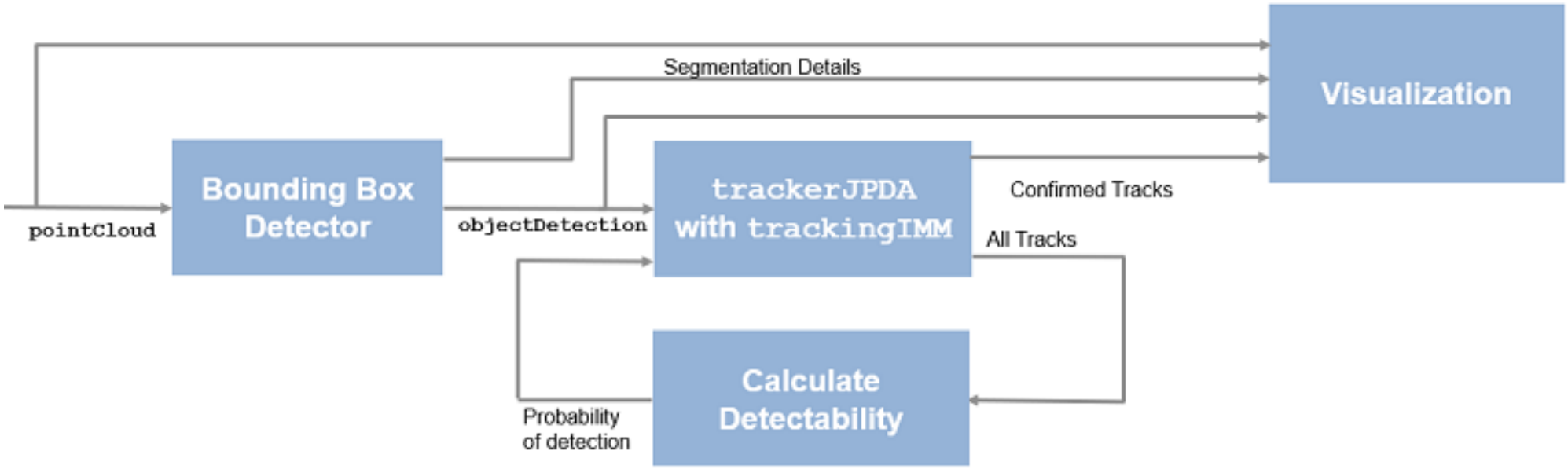
Detect Objects in Lidar



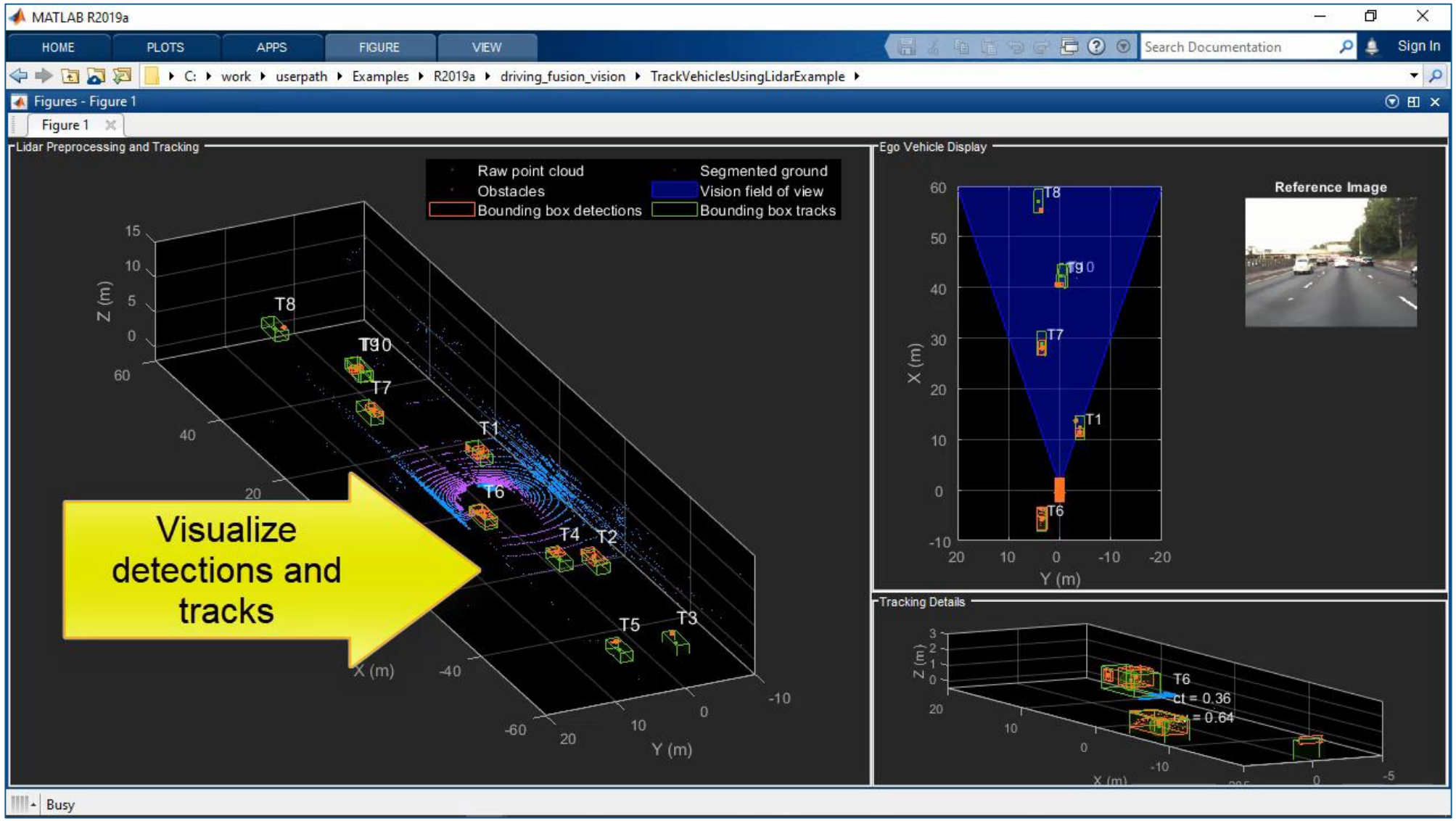
Detect Objects in Lidar



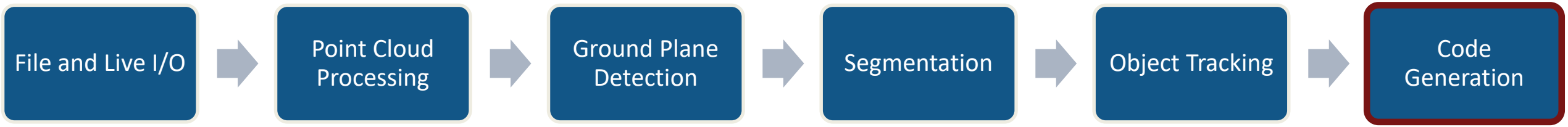
Design Tracker for Lidar Point Cloud



Design Tracker for Lidar Point Cloud

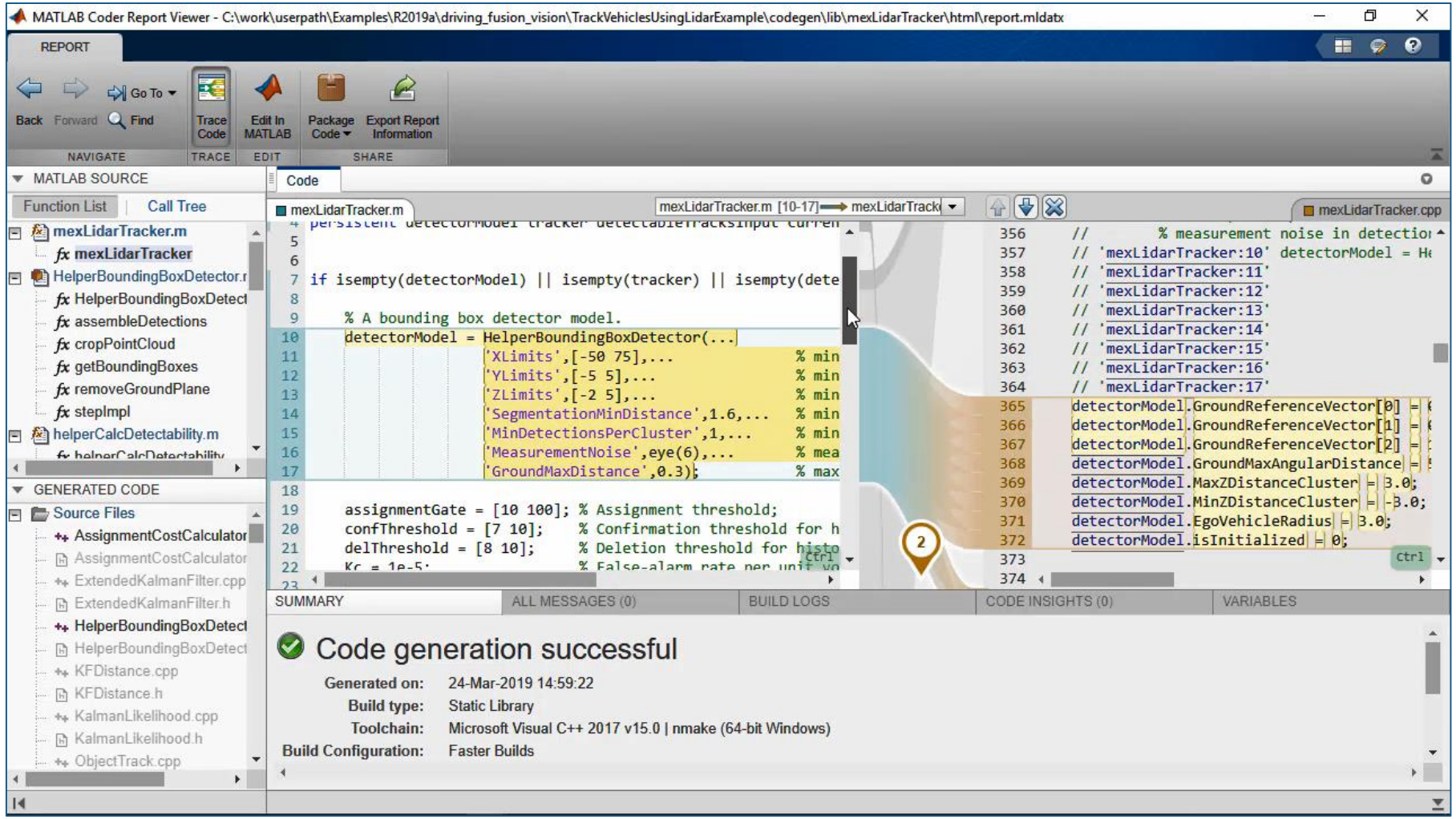


Talk Outline

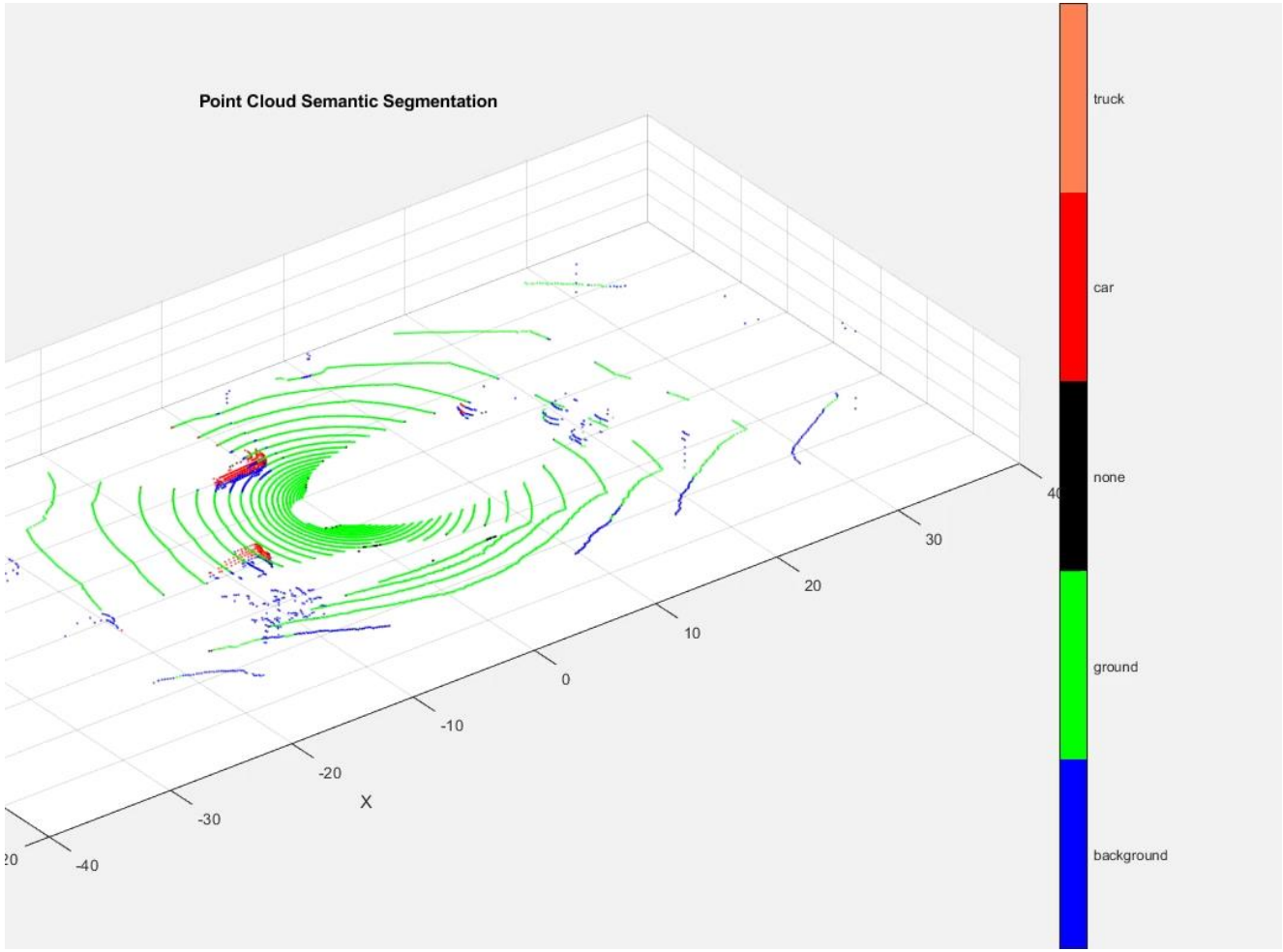


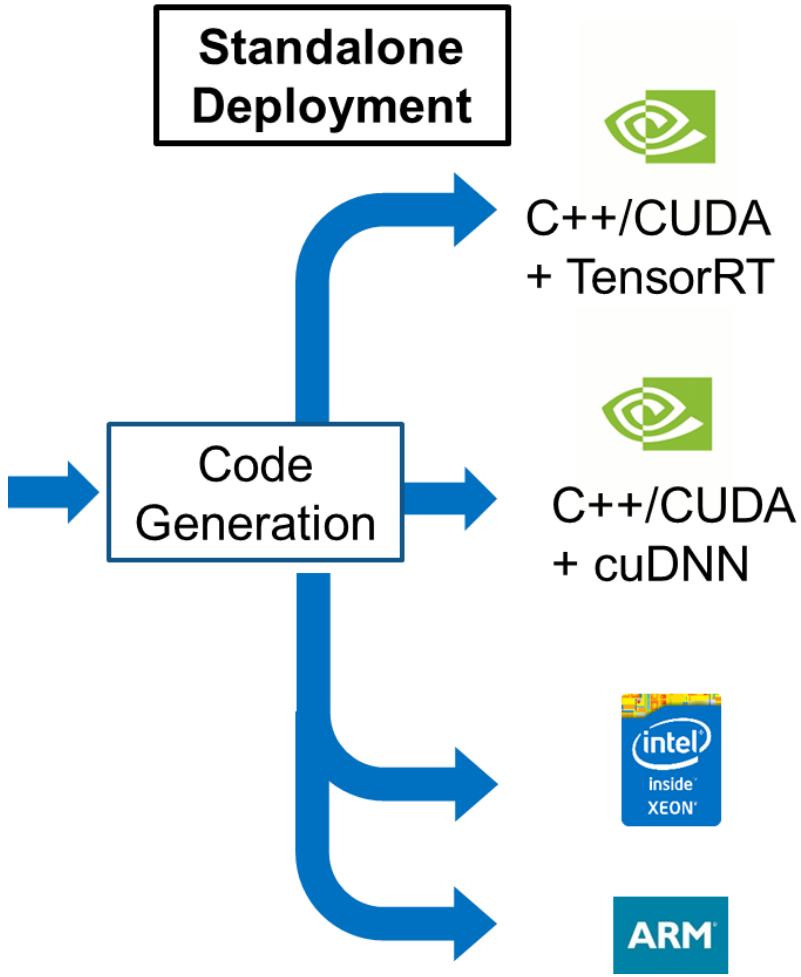
- C++ code generation for lidar detection and tracking
- CUDA code generation for deep learning

Generate C/C++ Code for Lidar Detector and Tracker



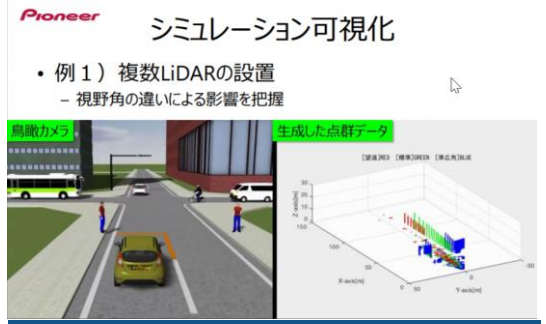
Deep Learning Model Deployment



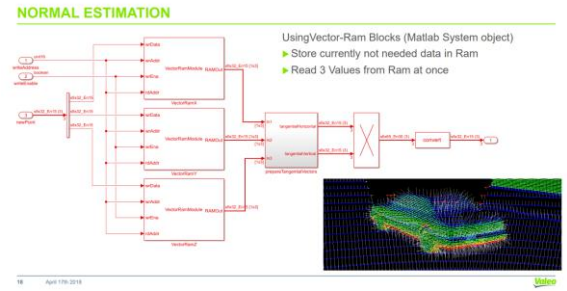


Customer Success Using MATLAB for Lidar Processing

- Lidar sensor design



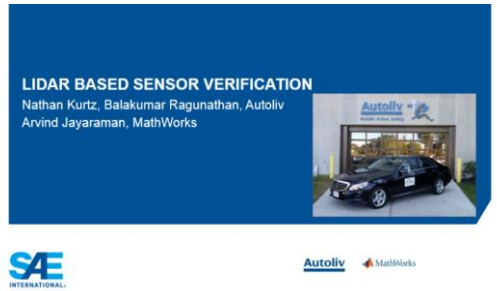
Pioneer



VALEO

- Radar verification with Lidar

  Automated 90% *of data analysis*



Veoneer

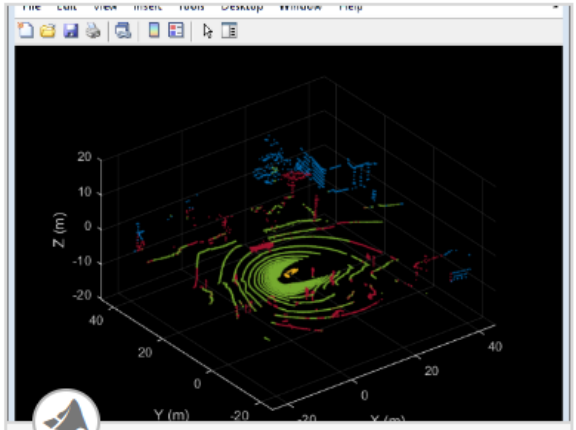
- Perception with Lidar

"We've used both Python and MATLAB to work with lidar sensor data, and I estimate that ***analysis and development was one-and-a-half to two times faster in MATLAB****,"*

Clearpath Robotics

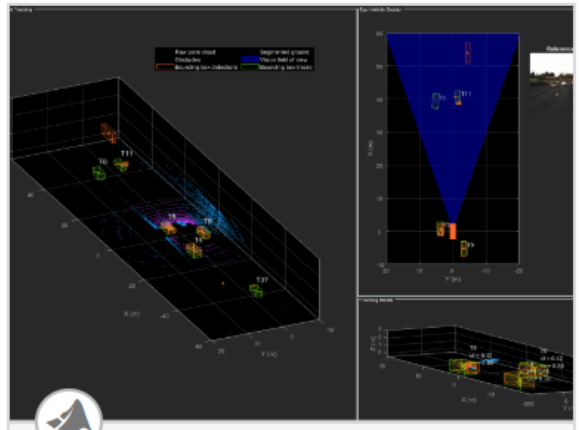
Reference Examples on Lidar Processing

Lidar Processing



Ground Plane and Obstacle Detection Using Lidar

Detect the ground plane and find nearby obstacles in 3-D lidar data.



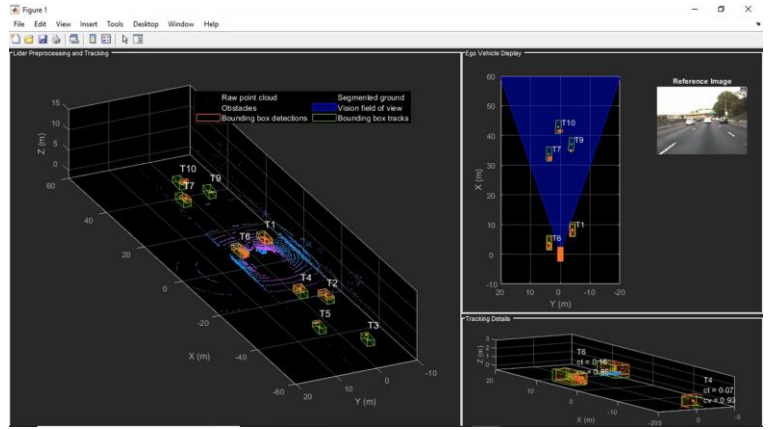
Track Vehicles Using Lidar: From Point Cloud to Track List

Track vehicles from lidar data using a joint probabilistic data association (JPDA) tracker and an interacting multiple model (IMM) approach.

https://www.mathworks.com/help/driving/examples.html

Summary: Lidar Processing for Automated Driving

File and Live I/O → Point Cloud Processing → Ground Plane Detection → Segmentation → Object Tracking → Code Generation



Why use MATLAB for Lidar processing ?

- Library of algorithms for complete workflow
- High-performance visualization tools
- C/C++ and CUDA code generation support