

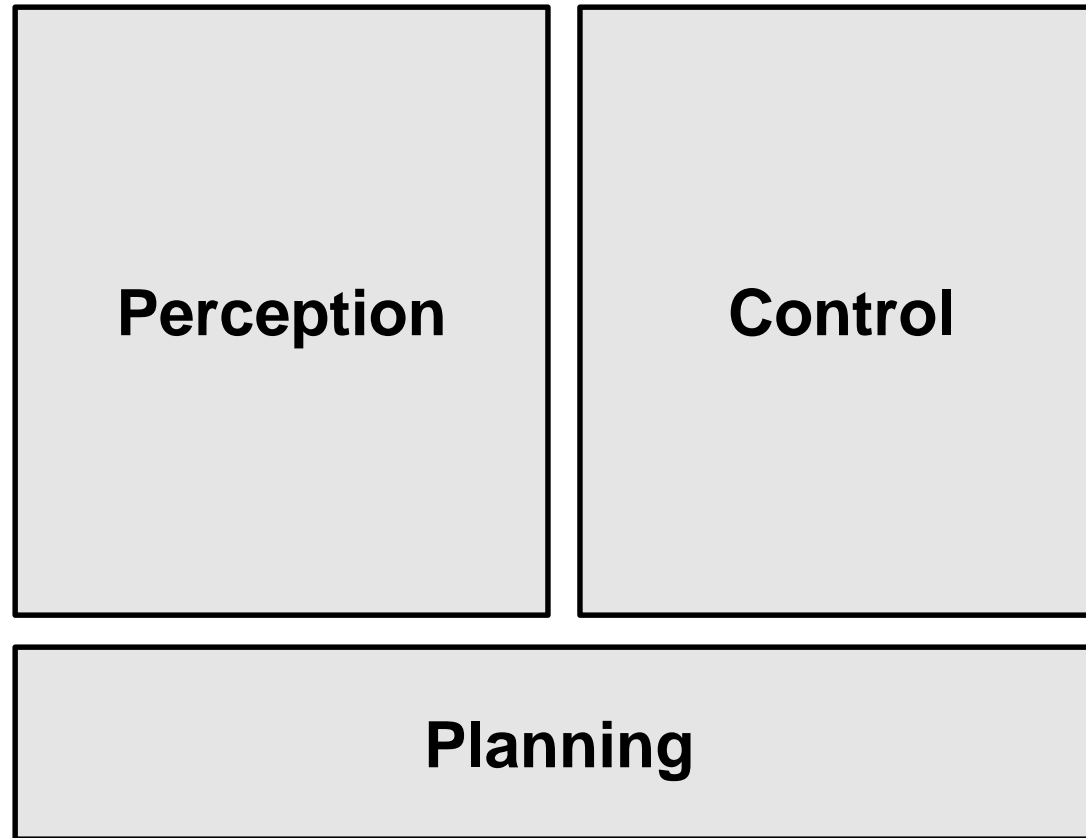
MATLAB EXPO 2018

Sensor Fusion and Tracking for Automated Driving

Abhijit Bhattacharjee
Senior Application Engineer
MathWorks



How can you use MATLAB and Simulink to develop automated driving algorithms?



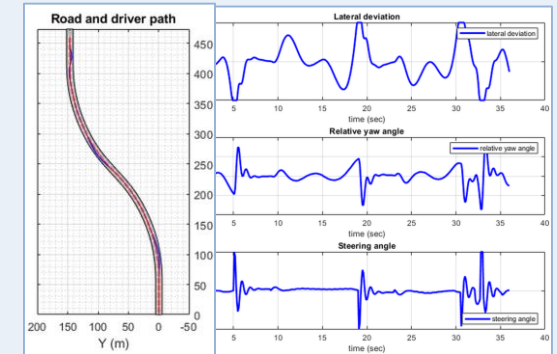
Examples of how you can use MATLAB and Simulink to develop automated driving algorithms

Deep learning



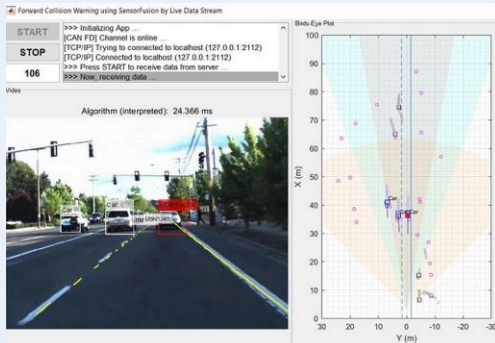
Perception

Sensor models & model predictive control



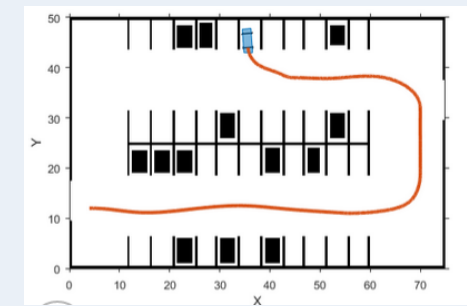
Control

Sensor fusion



Planning

Path planning



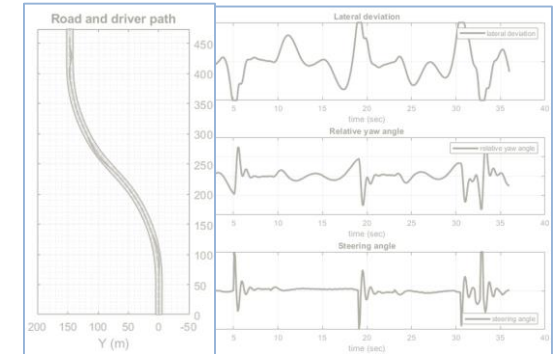
How can you use MATLAB and Simulink to develop perception algorithms?

Deep learning



Perception

Sensor models & model predictive control



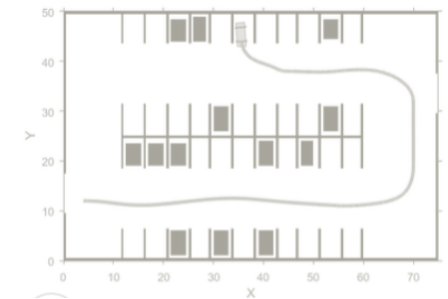
Control

Sensor fusion



Planning

Path planning



Detection drivable space using semantic segmentation



Learn more about developing deep learning perception algorithms with these examples

R2018a

Automate Ground Truth Labeling for Semantic Segmentation

R2017b

Semantic Segmentation Using Deep Learning

R2018a

Code Generation for Semantic Segmentation Network

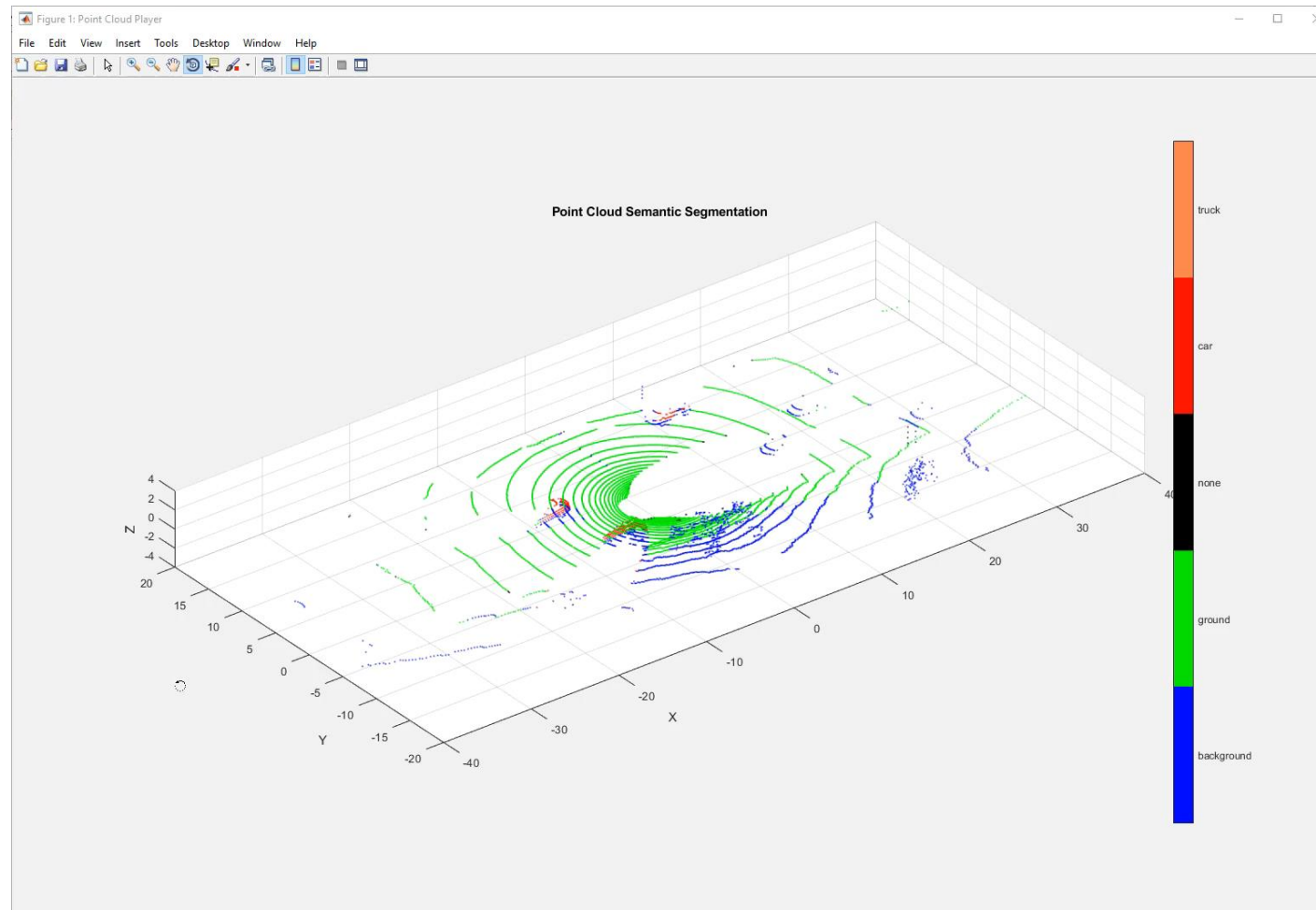
Bicyclist
Pedestrian
Car
Fence
SignSymbol
Tree
Pavement
Road
Pole
Building
Sky

- **Add semantic segmentation automation algorithm to Ground Truth Labeler App**
Automated Driving System Toolbox™

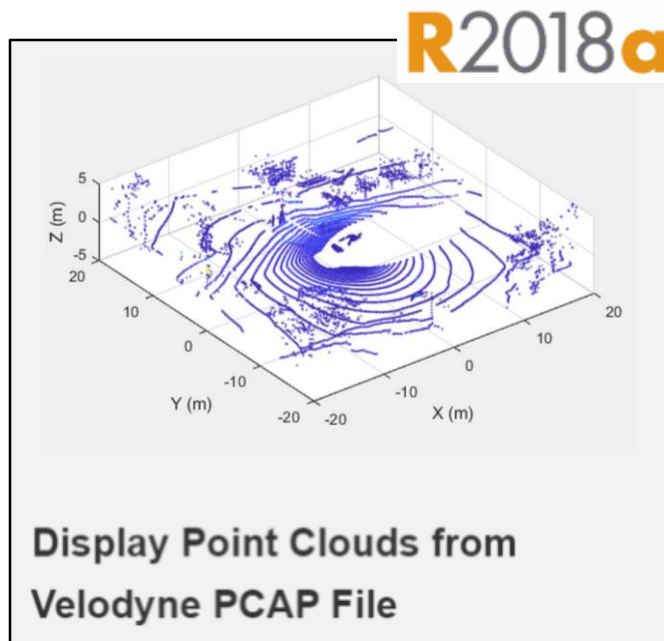
- **Train free space detection network using deep learning**
Computer Vision System Toolbox™

- **Generate CUDA® code to execute directed acyclic graph network on an NVIDIA GPU**
GPU Coder™

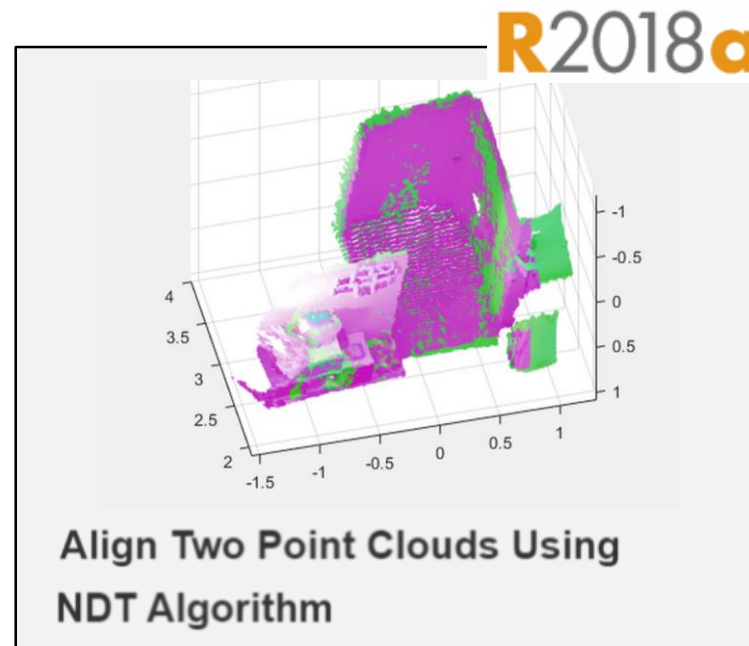
LIDAR Point Cloud Segmentation



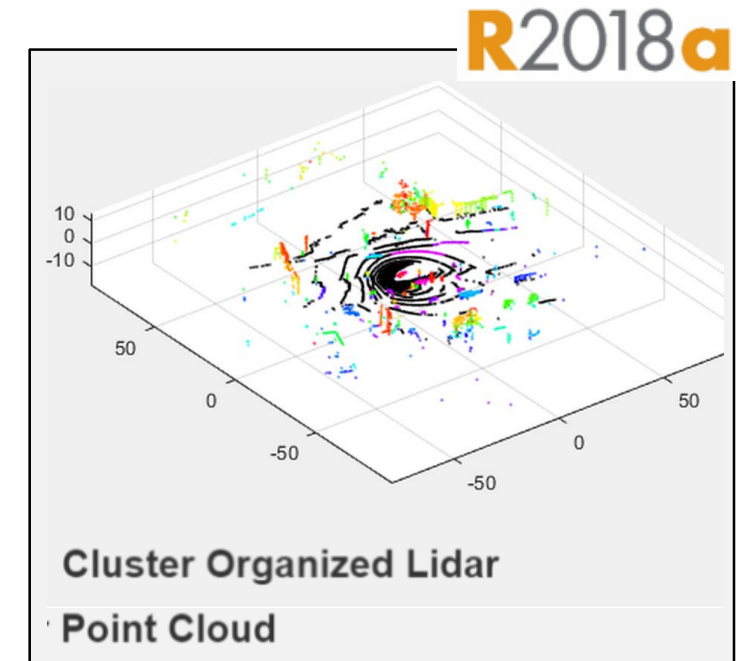
Learn about developing lidar perception algorithms with these examples



- **Read Velodyne files**
`velodyneFileReader`
 Automated Driving System Toolbox™



- **Register** point clouds with Normal Distributions Transform
`pcregisterndt`
 Computer Vision System Toolbox™



- **Segment** lidar point cloud
`segmentLidarData`
 Automated Driving System Toolbox™

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

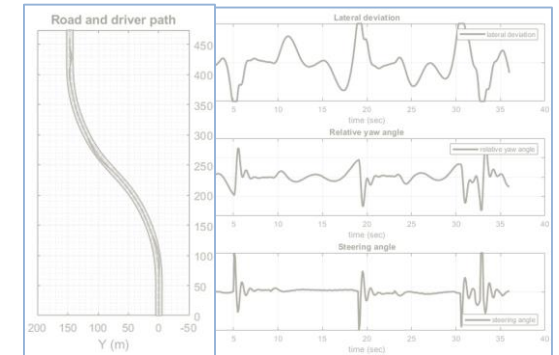
Deep learning



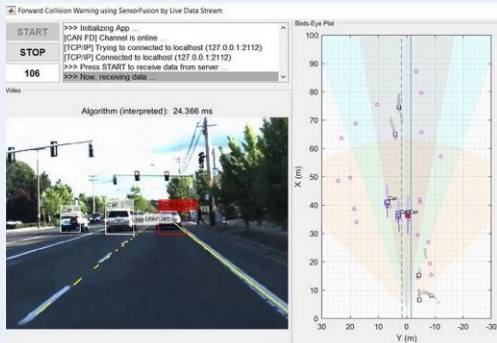
Perception

Control

Sensor models & model predictive control

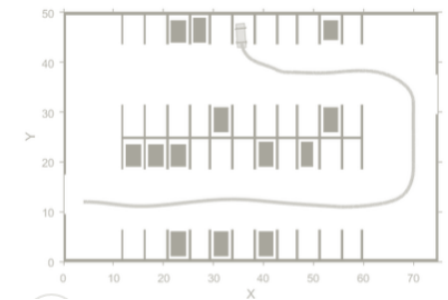


Sensor fusion

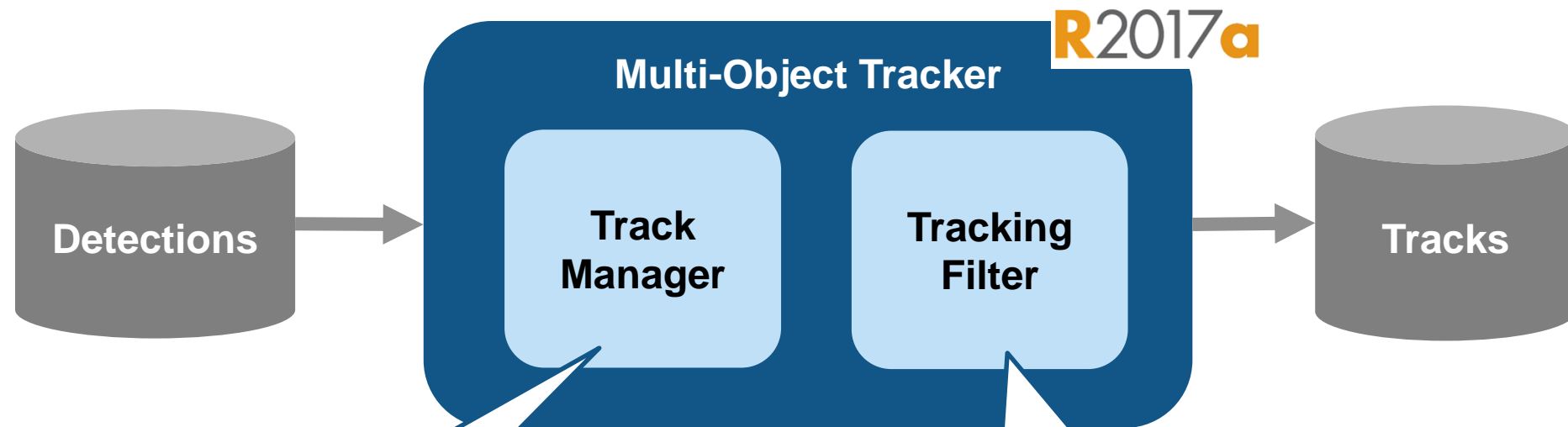


Planning

Path planning



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



- 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

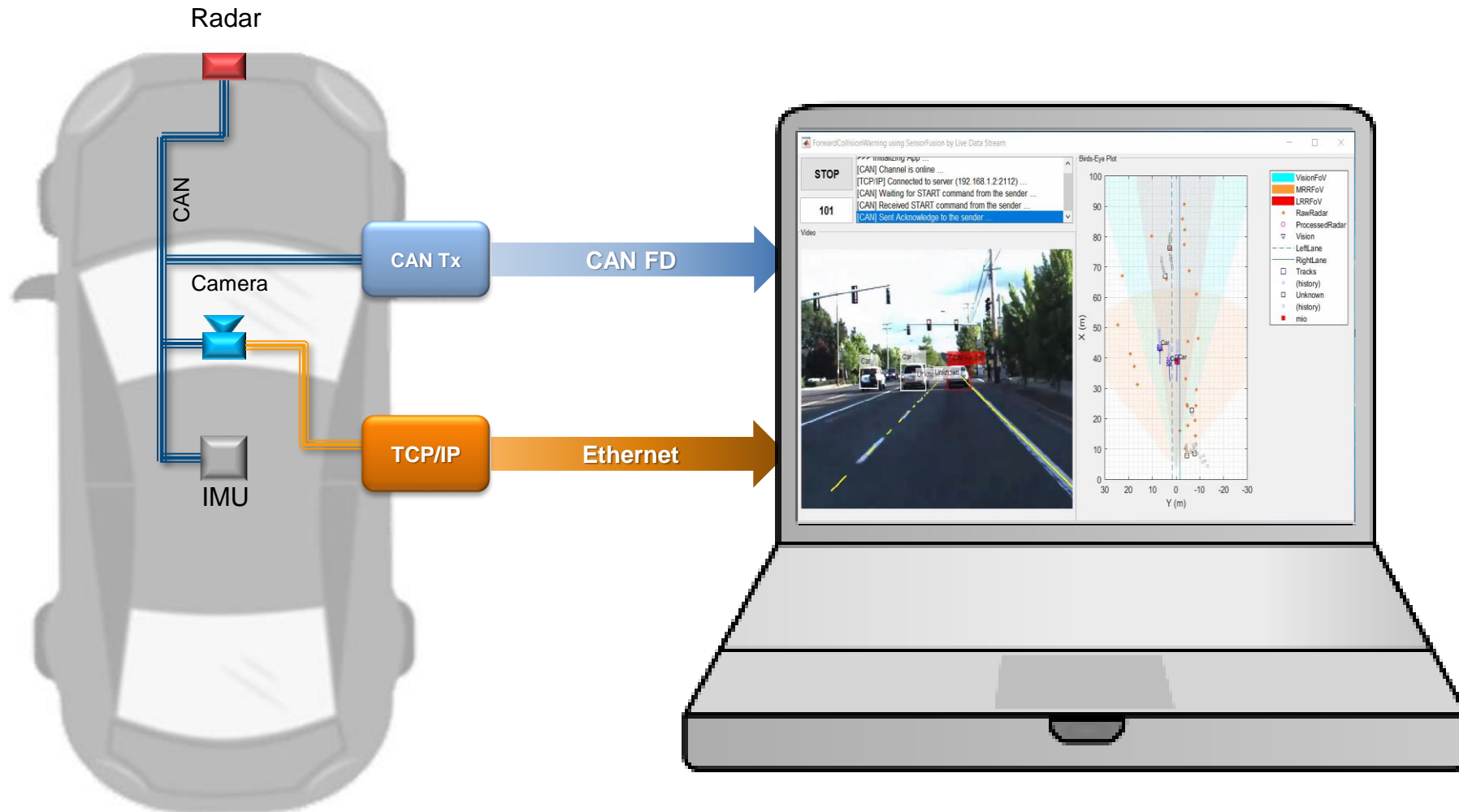
Videos and Webinars

Some common questions from automated driving engineers

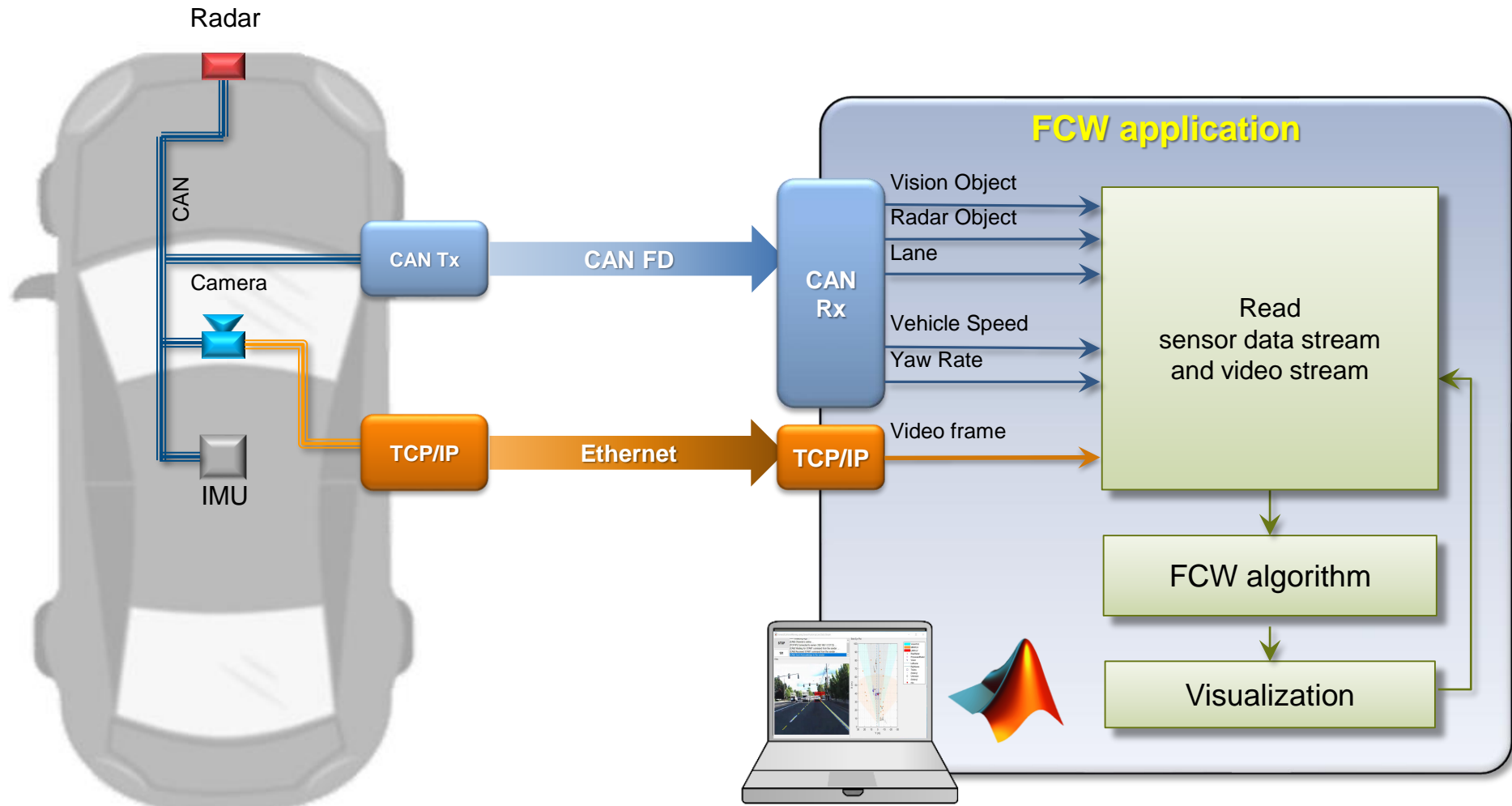
19:27

Introduction to Automated Driving System Toolbox

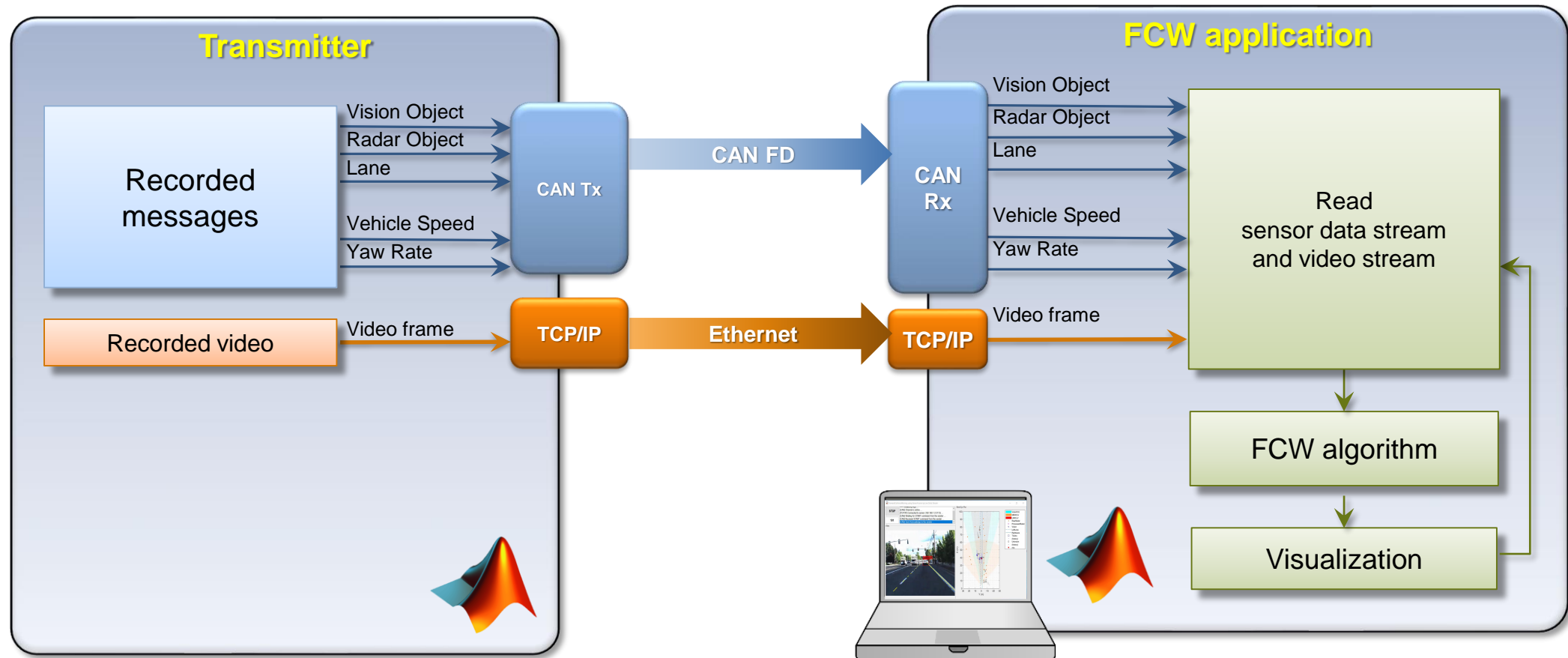
How can I test my sensor fusion algorithm with live data?



Test forward collision warning algorithm with live data from vehicle



Test forward collision warning algorithm with live data from “surrogate” vehicle



Send live CAN FD and TCP/IP data

MATLAB Command Window

```

To get started, type one of t
For product information, visi

[Timer] Timer is ready ...
[CAN FD] Initializing channel
[CAN FD] Channel is online ..
[TCP/IP] Waiting for Client C
[TCP/IP] Connected to Client
>>> Now, sending data ...
Press Enter to end
        
```

Forward Collision Warning using SensorFusion by Live Data Stream

START >> Initializing App ...
[CAN FD] Channel is online ...
[TCP/IP] Trying to connect to localhost (127.0.0.1:2112)
[TCP/IP] Connected to localhost (127.0.0.1:2112)
>> Press START to receive data from server ...
>> Now, receiving data ...

STOP

114

Detections received over CAN FD

Birds-Eye Plot

- VisionFoV
- MRRFoV
- LRRFoV
- Radar
- Vision
- LeftLane
- RightLane
- Tracks
- (history)
- Unknown
- (history)
- mio

Video received over TCP/IP

Algorithm (interpreted)

Toolbox CAN FD functions, Instrumentation System Toolbox capabilities are used

In this part of the example, run start to actively decode and execute the functions

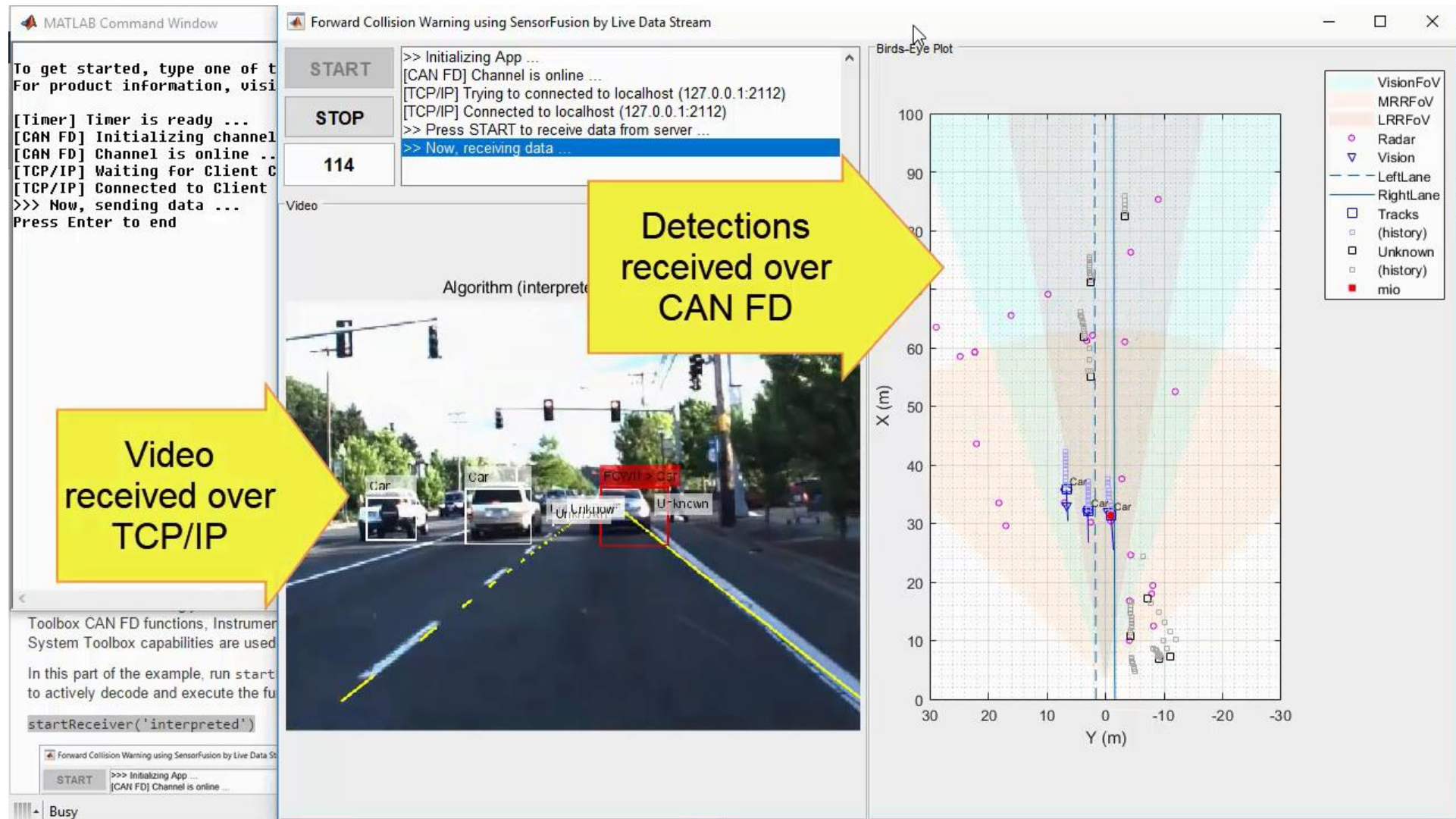
```
startReceiver('interpreted')
```

Forward Collision Warning using SensorFusion by Live Data Stream

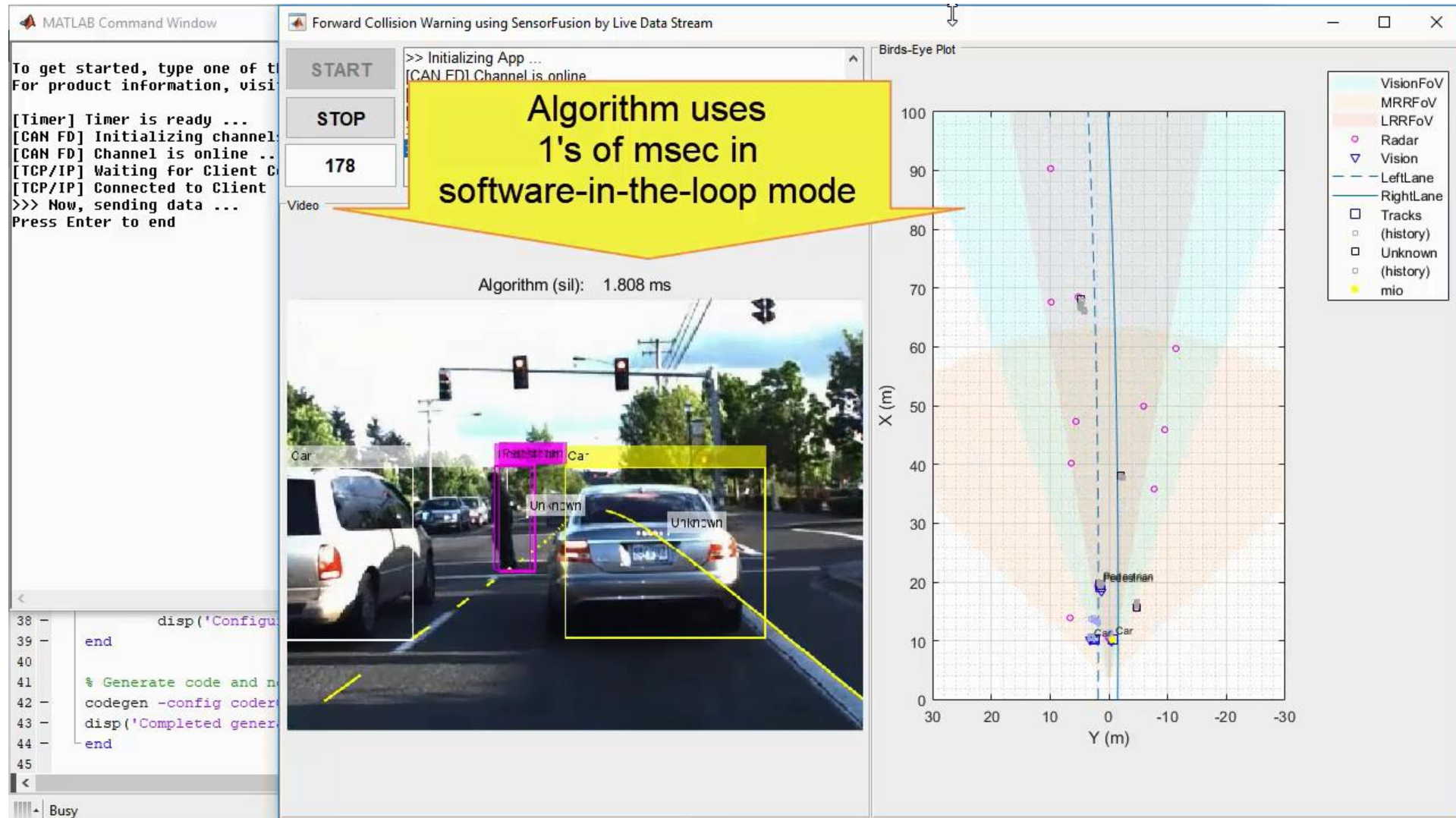
START >>> Initializing App ...
[CAN FD] Channel is online ...

Busy

Receive live CAN FD and TCP/IP data



Stream live CAN FD and TCP/IP data into compiled algorithm code



Learn more about developing sensor fusion algorithms

R2017a

Forward Collision Warning Using Sensor Fusion

R2017a

Code Generation for Tracking and Sensor Fusion

R2018a

Executing a Forward Collision Warning Application with Live CAN...

- **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



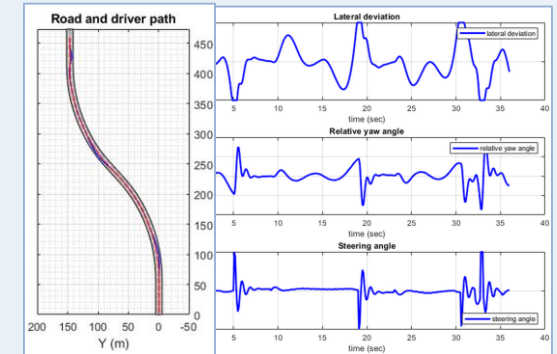
Perception

Sensor fusion



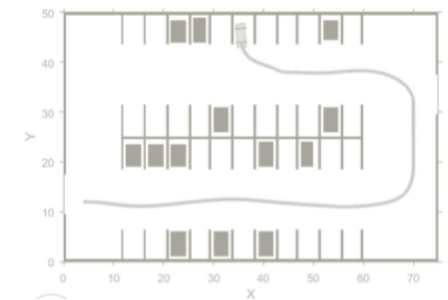
Control

Sensor models & model predictive control



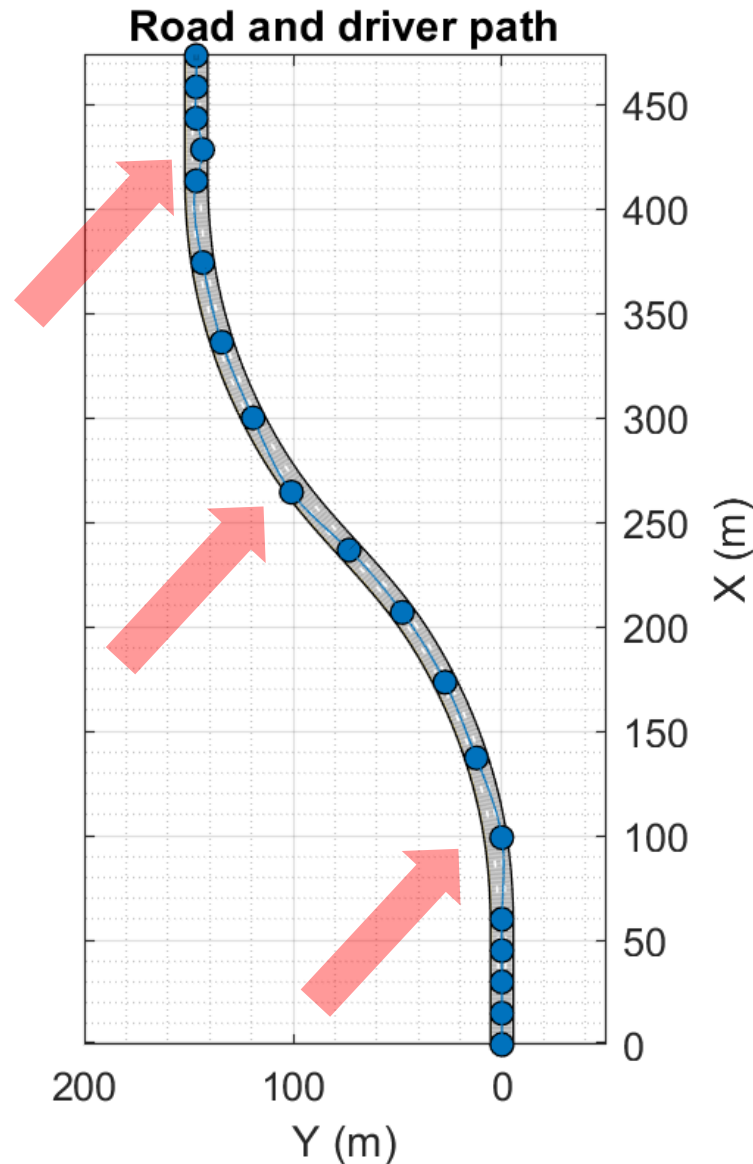
Planning

Path planning

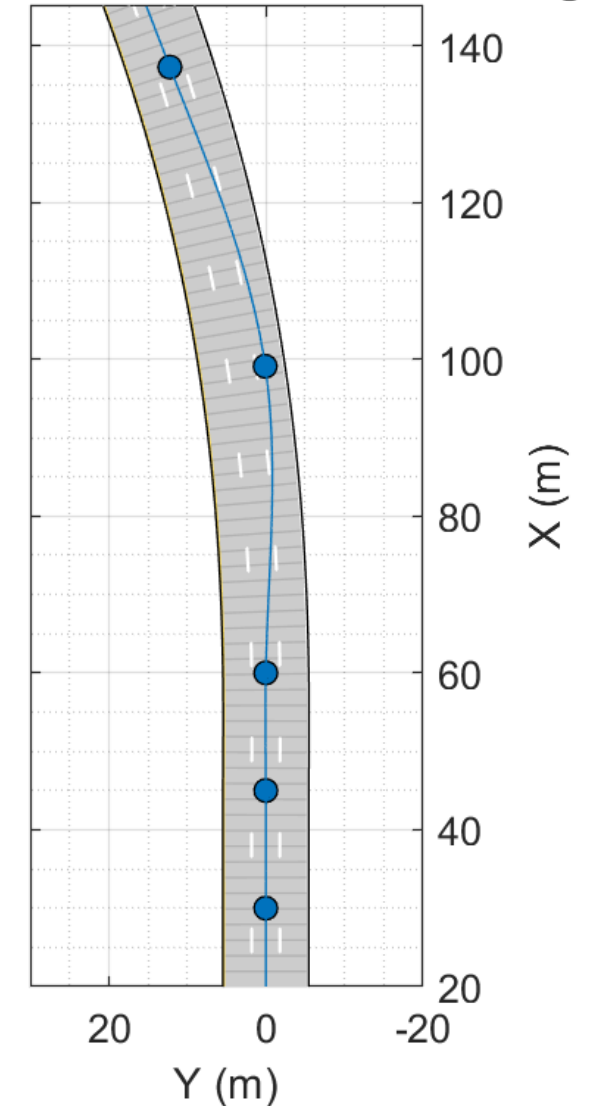


Create highway double curve with drivingScenario

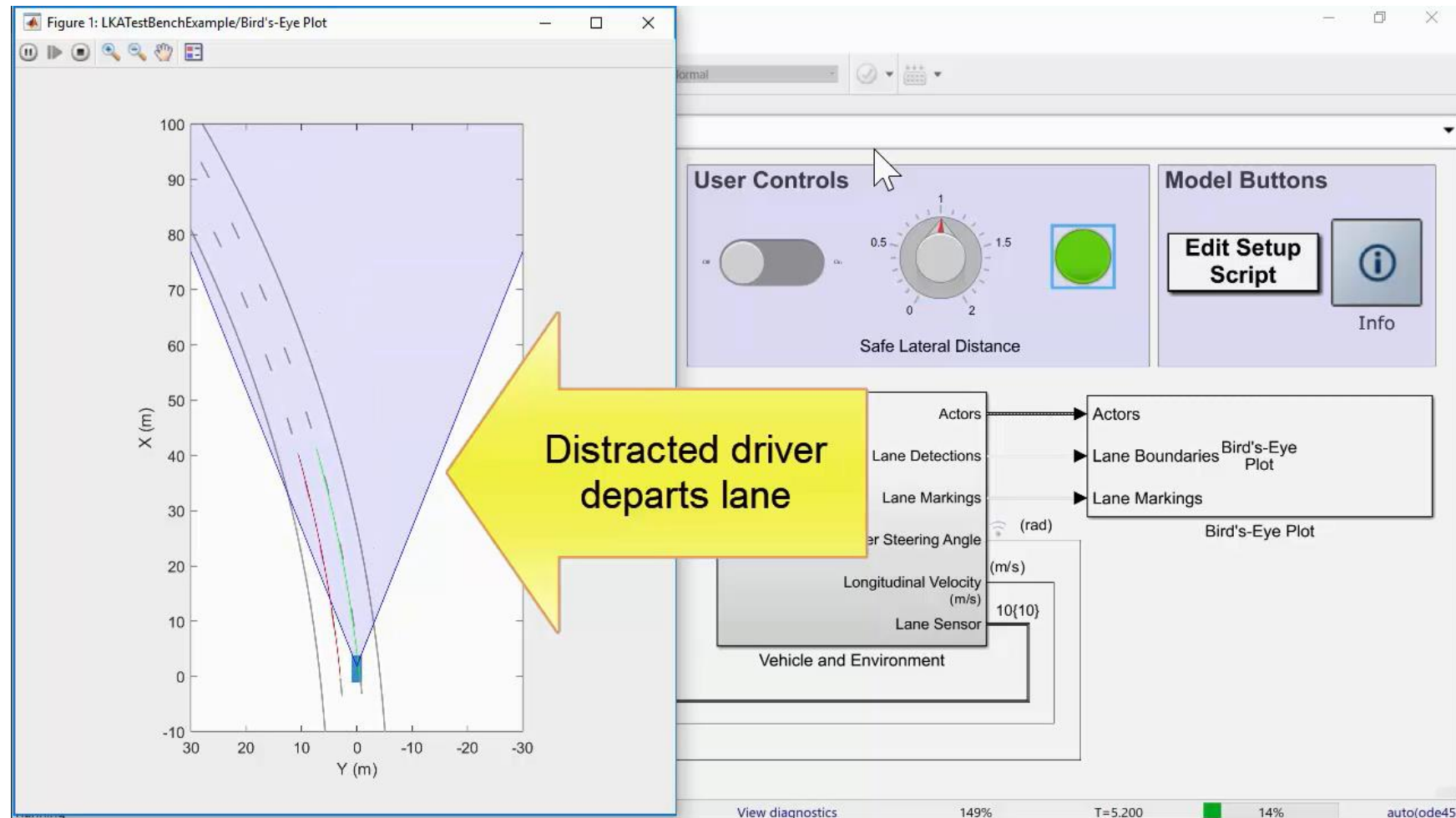
- Driver waypoints simulate distraction at curvature changes



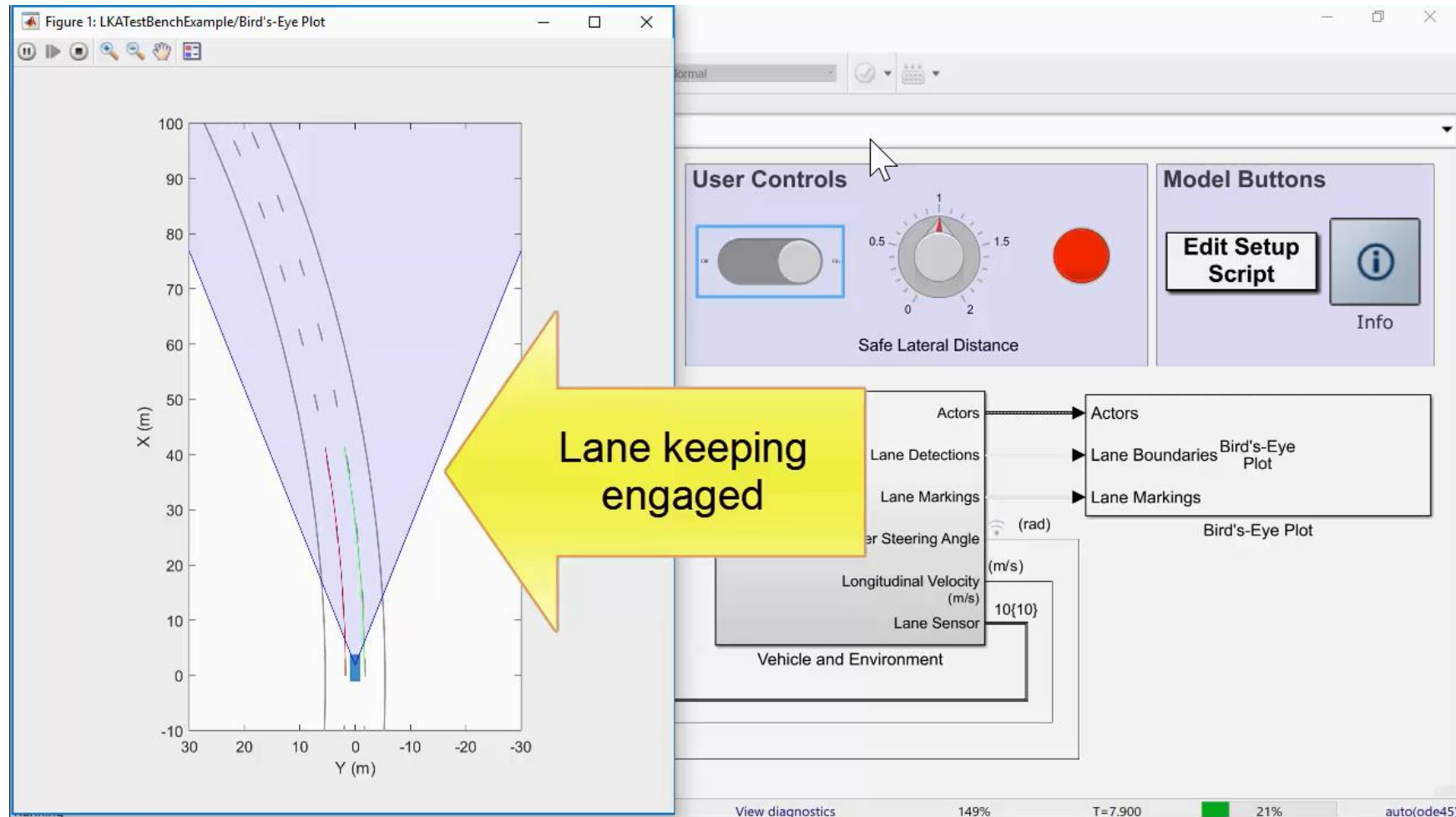
Driver distracted at curvature change



Simulate distracted driver

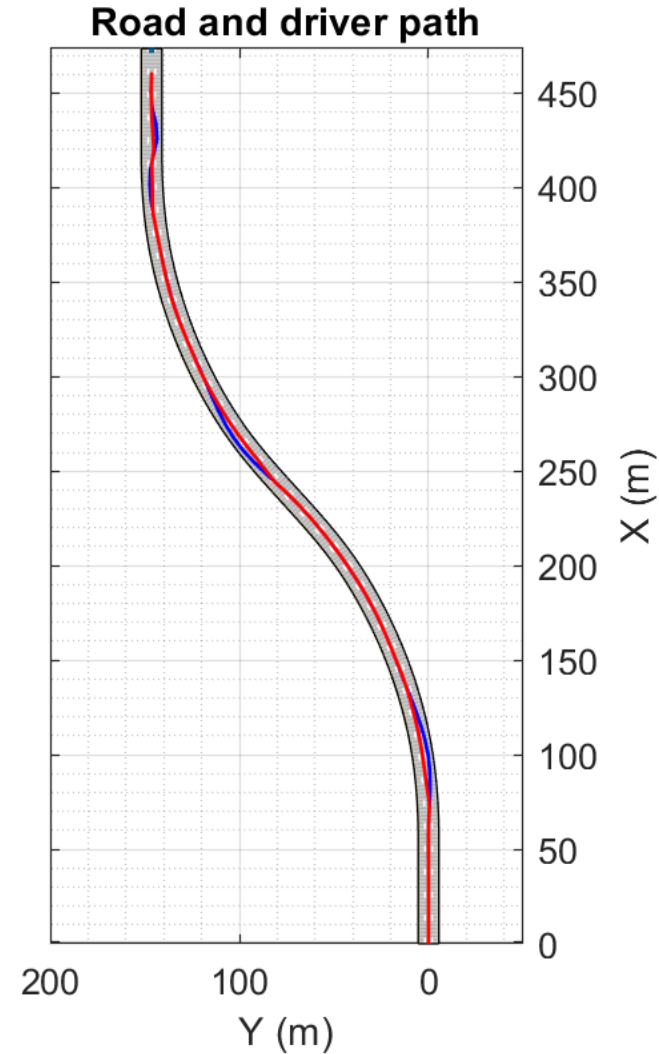


Simulate lane keep assist at distraction events

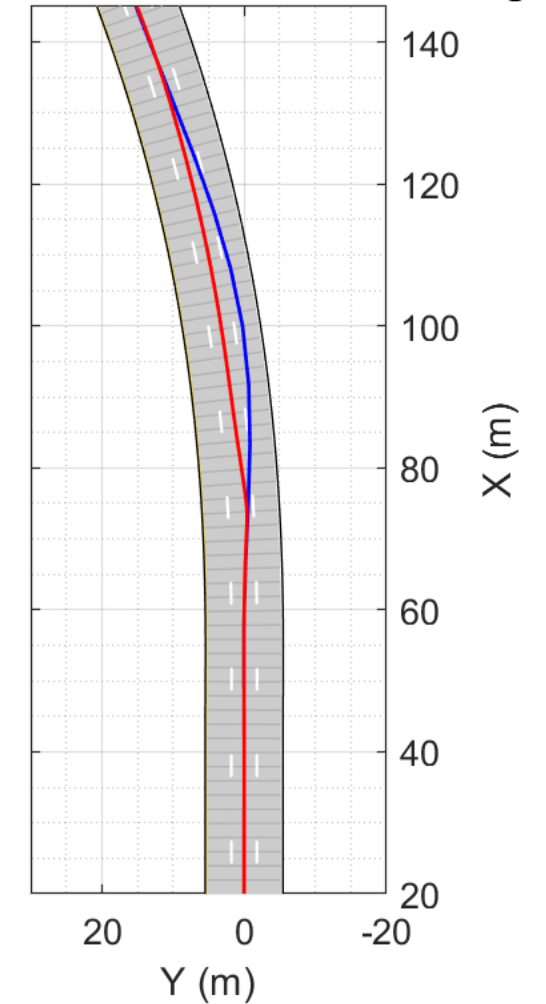


Compare distracted and assisted results

- Detect lane departure and maintain lane during distraction

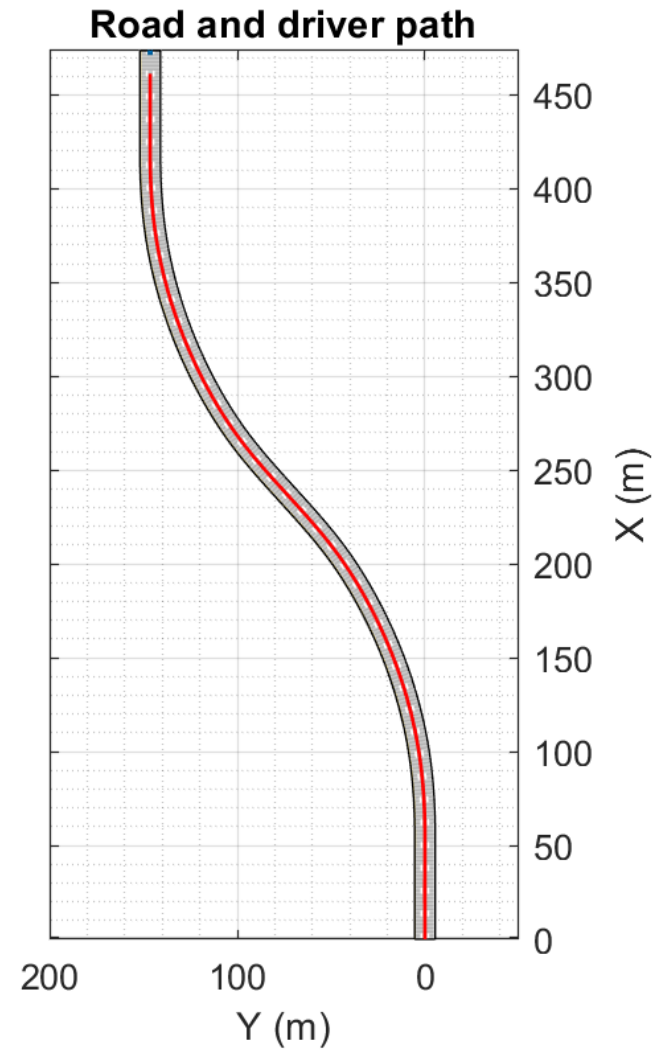


Driver assisted at curvature change

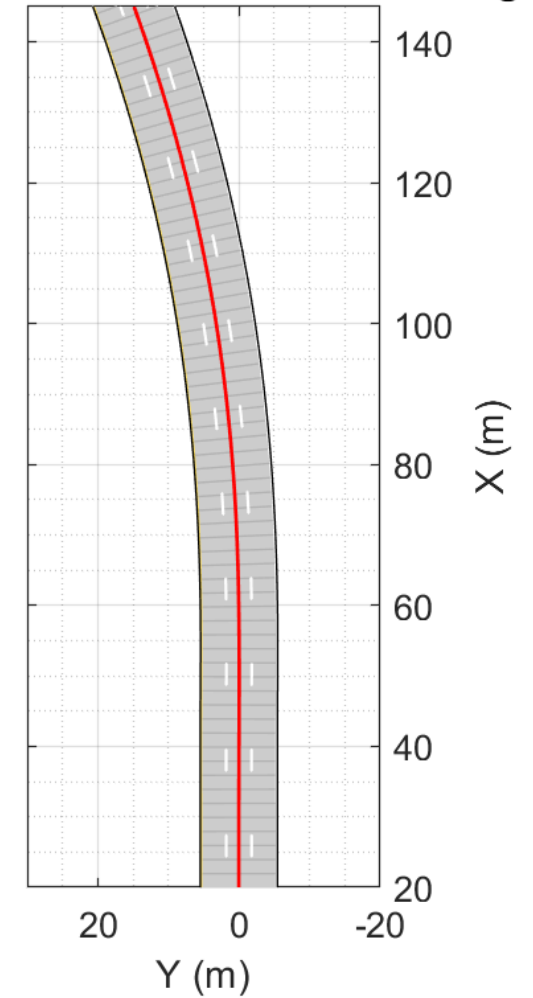


Explore lane following results

- Vehicle stays within lane boundaries



Driver assisted at curvature change



Graphically edit scenarios with Driving Scenario Designer

The screenshot displays the Driving Scenario Designer software interface. The main window is titled "Driving Scenario Designer - untitled* - Scenario Canvas". The interface is divided into several sections:

- DESIGNER Toolbar:** Contains icons for New, Open, Save, Add Road, Add Actor, Add Camera, Add Radar, Go to Start, Step Back, Run, Step Forward, Settings, Repeat, Default Layout, and Export.
- Left Panel:** Includes a "Roads" section with a dropdown menu set to "1", and fields for Name, Width (m) (set to 6), and Bank Angle (deg) (set to 0). Below this is a "Road Centers" table.
- Scenario Canvas:** A 2D plot showing a road layout. The Y-axis ranges from 0 to 200, and the X-axis ranges from 100 to -80. A blue road is shown, with a yellow arrow pointing to it and the text "Graphically edit scenario".
- Ego Centric View:** A 3D view showing a blue car (the ego vehicle) on a road.

	x (m)	y (m)	z (m)
1	0	0	0
2	15	0	0
3	30	0	0
4	45	0	0
5	60	0	0
6	99.10...	3.0779	0
7	131.2...	19.80...	0
8	173.4...	27.24...	0
9	206.9...	47.74...	0
10	236.7...	73.22...	0
11	266.6...	98.70...	0
12	300.0...	119.1...	0
13	336.2...	134.2...	0
14	374.4...	143.3...	0
15	413.5...	146.4...	0
16	428.5...	146.4...	0
17	443.5...	146.4...	0
18	458.5...	146.4...	0
19	473.5...	146.4...	0

Export MATLAB code to generate scenarios

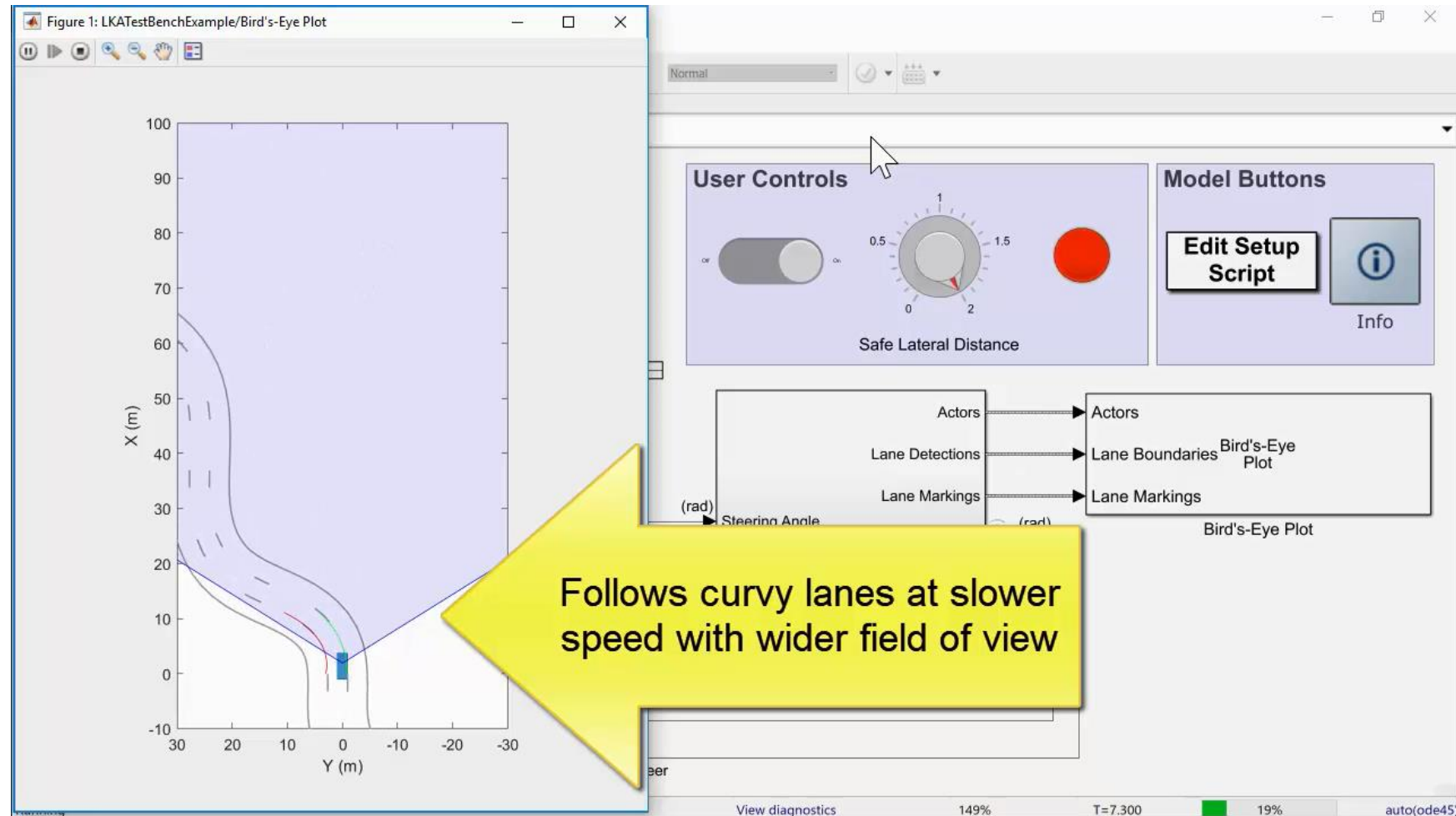
The screenshot displays the 'Driving Scenario Designer - untitled* - Scenario Canvas' window. The interface includes a top toolbar with various icons for file operations, scenario management, and simulation. A yellow arrow points to the 'Export' button (a green checkmark icon) in the toolbar, with the text 'Export to MATLAB code' overlaid on it. A context menu is open over the 'Export' button, listing the following options:

- Export MATLAB Function**
Generate MATLAB function for your driving scenario and sensors
- Export Sensor Data**
Export sensor data from last simulation run to base workspace

The main workspace is divided into three panels:

- Left Panel (Actors):** Shows a list of actors, currently containing '1: (ego car)'. Below the list are fields for 'Name' and 'Class' (set to 'Car'), and expandable sections for 'Actor Properties', 'Radar Cross Section', and 'Trajectory'.
- Center Panel (Scenario Canvas):** A 2D plot with X (m) on the vertical axis (ranging from 0 to 200) and Y (m) on the horizontal axis (ranging from 100 to -80). It shows a grey road path with several blue circular markers representing the ego car's trajectory.
- Right Panel:** A 3D perspective view of the ego car, shown as a blue rectangular block on a grey road surface.

Explore what is required to follow high curvature paths



Learn about synthesizing sensor detections to develop control algorithms with these examples

R2017b

Adaptive Cruise Control with Sensor Fusion

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

R2018a

Lane Keeping Assist with Lane Detection

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

R2018a

Generate Synthetic Detections from an Interactive Driving Scenario

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

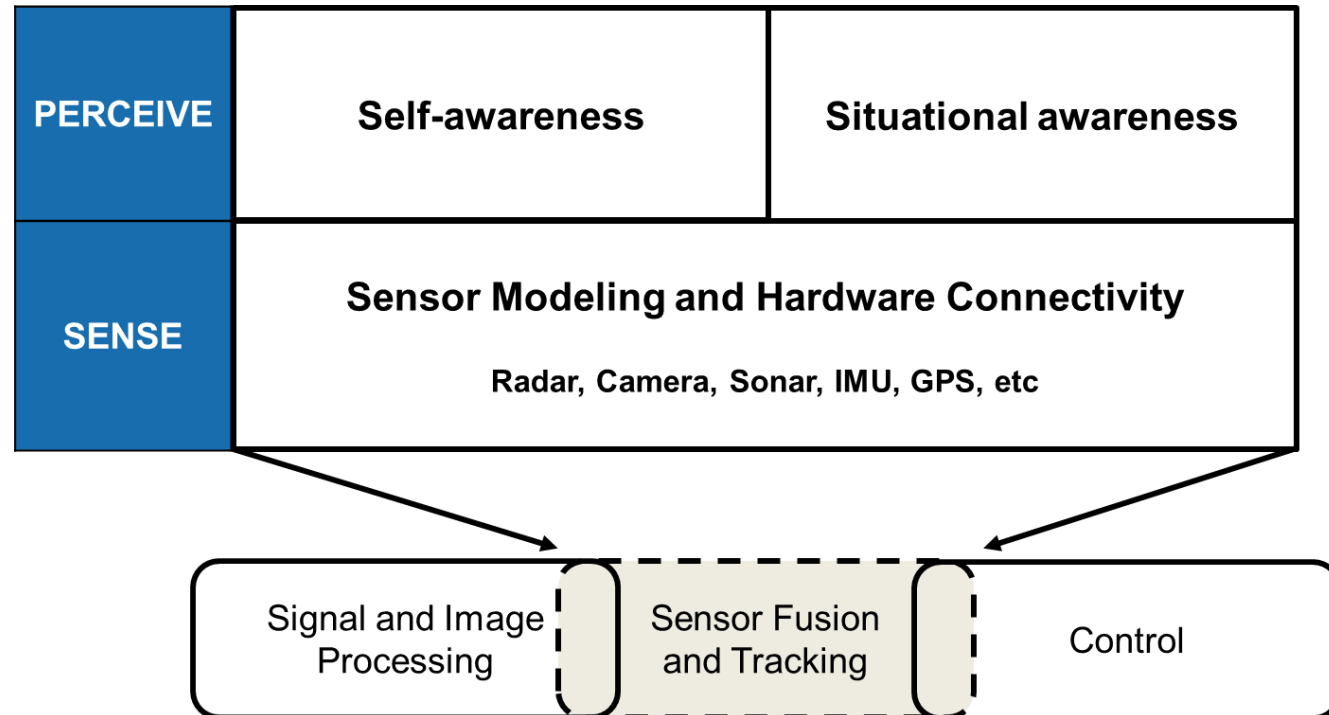
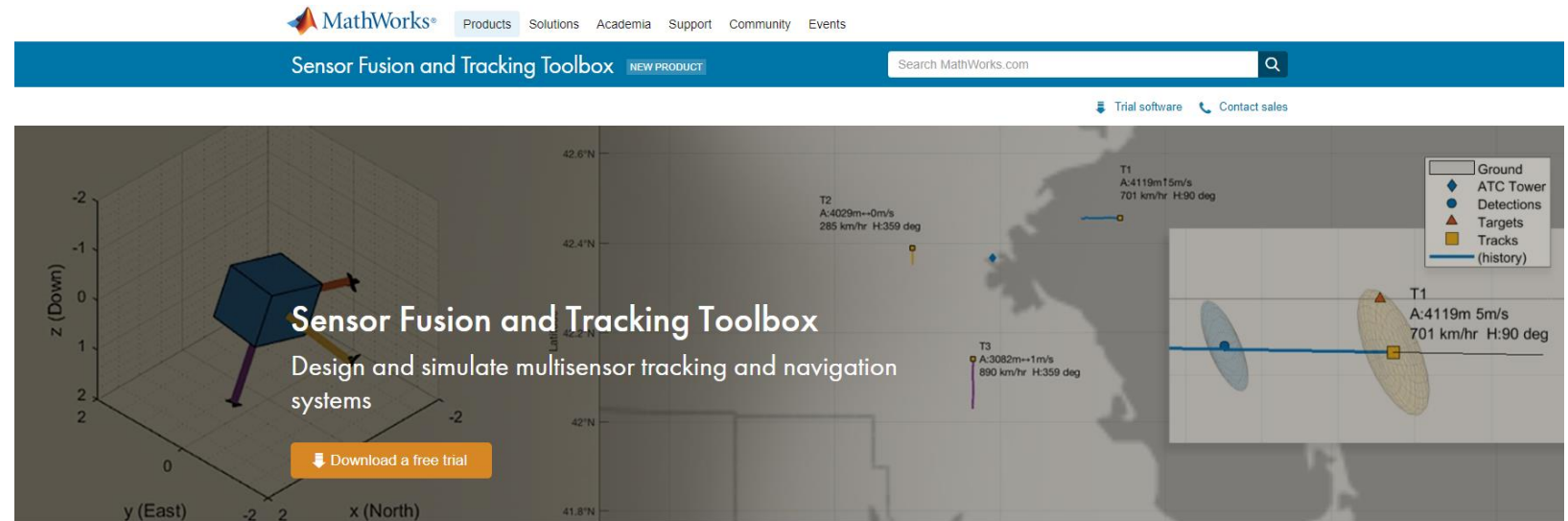
Sensor Fusion and Tracking Toolbox

Extending the Automated Driving Workflow

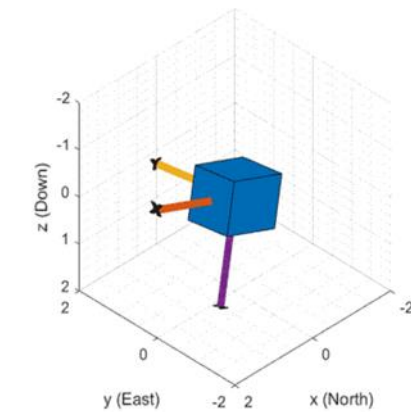
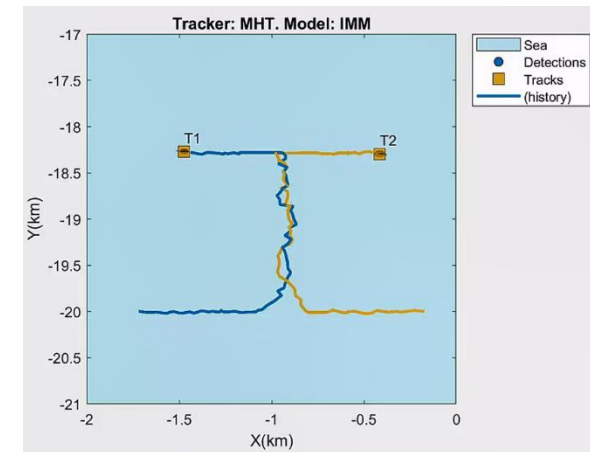
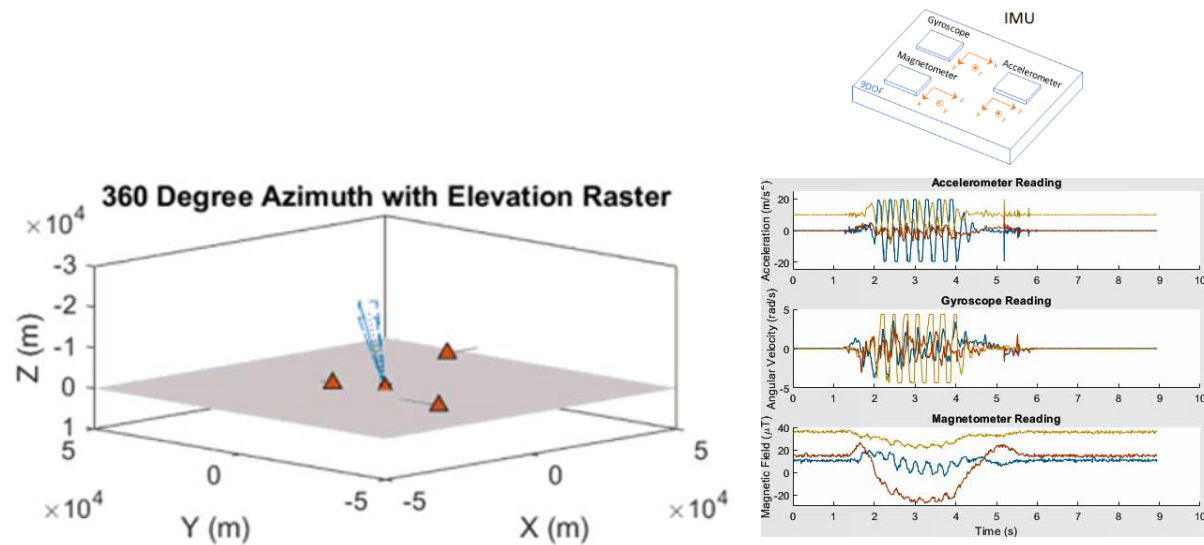
- Localization for pose, orientation, and position
- Tracker and fusion algorithm development
- Metrics – accuracy and track assignment

In this part:

- Scope
- Overview
- Applications

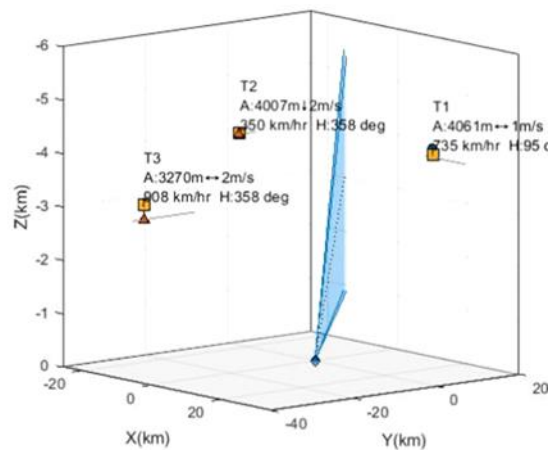


Overview: Sensor Fusion and Tracking Toolbox

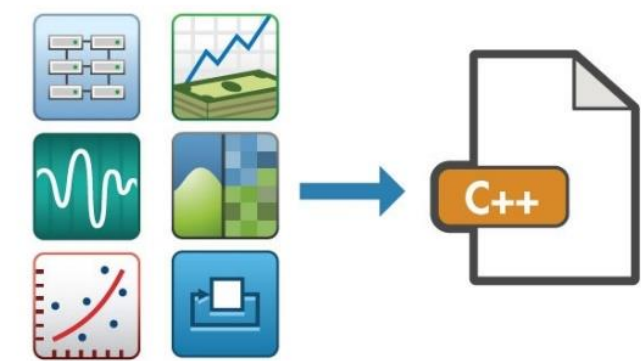


Scenarios and Sensors Simulation

Tracking and Localization Algorithms



TrackID	AssignedTruthID	TotalLength
2	NaN	1
3	NaN	1
4	8	55
5	7	56
6	NaN	2

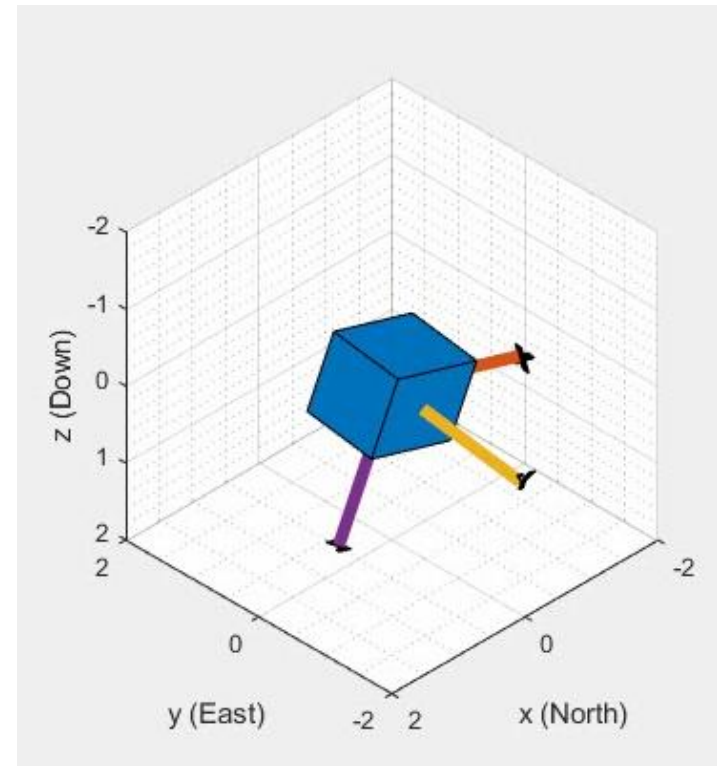


Visualization and Metrics

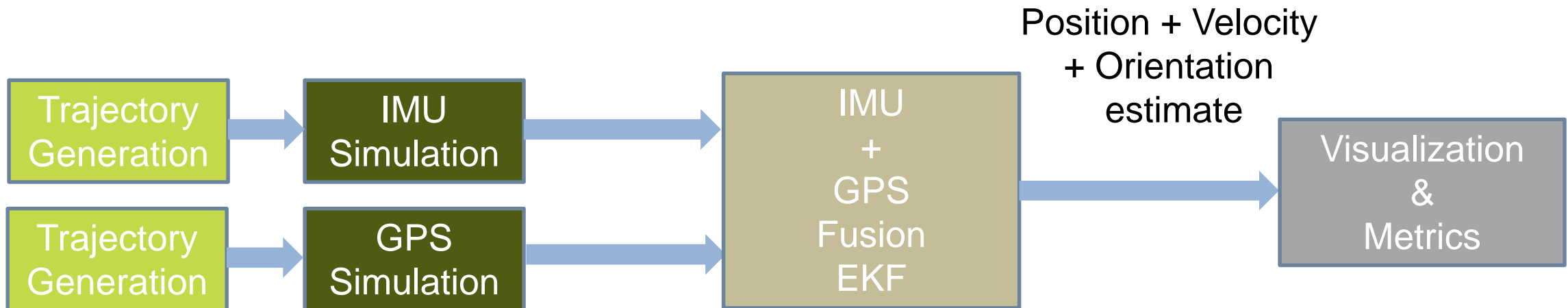
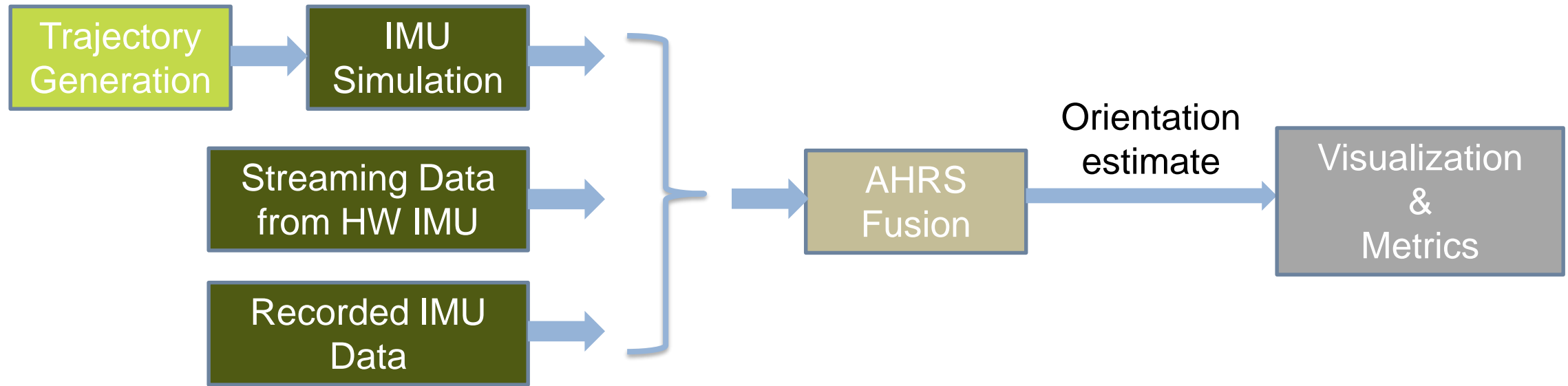
Code Generation

Localization

- Workflows
- IMU and GPS Sensor Models
- Orientation estimation (*6-DOF, 9-DOF, 10-DOF*)
- Pose estimation (*IMU+GPS, IMU+Visual Odometry*)



Localization Algorithm Development Workflow

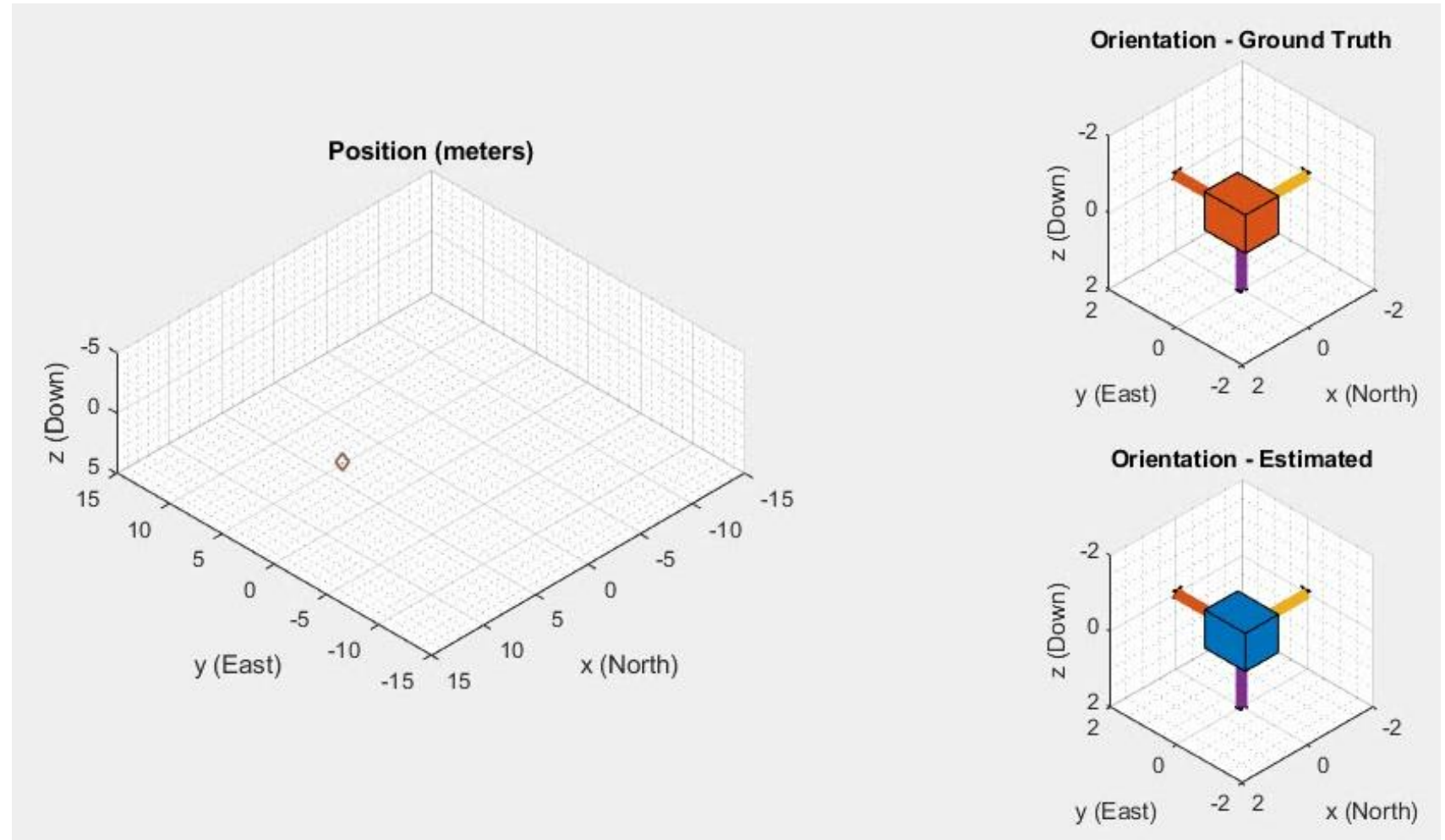


Estimate Orientation, Position and Velocity of Ground Vehicles

R2018b

IMU + GPS Fusion

Non-holonomic Motion Constraint



Estimate Position of Ground Vehicle without GPS

IMU + Camera Fusion

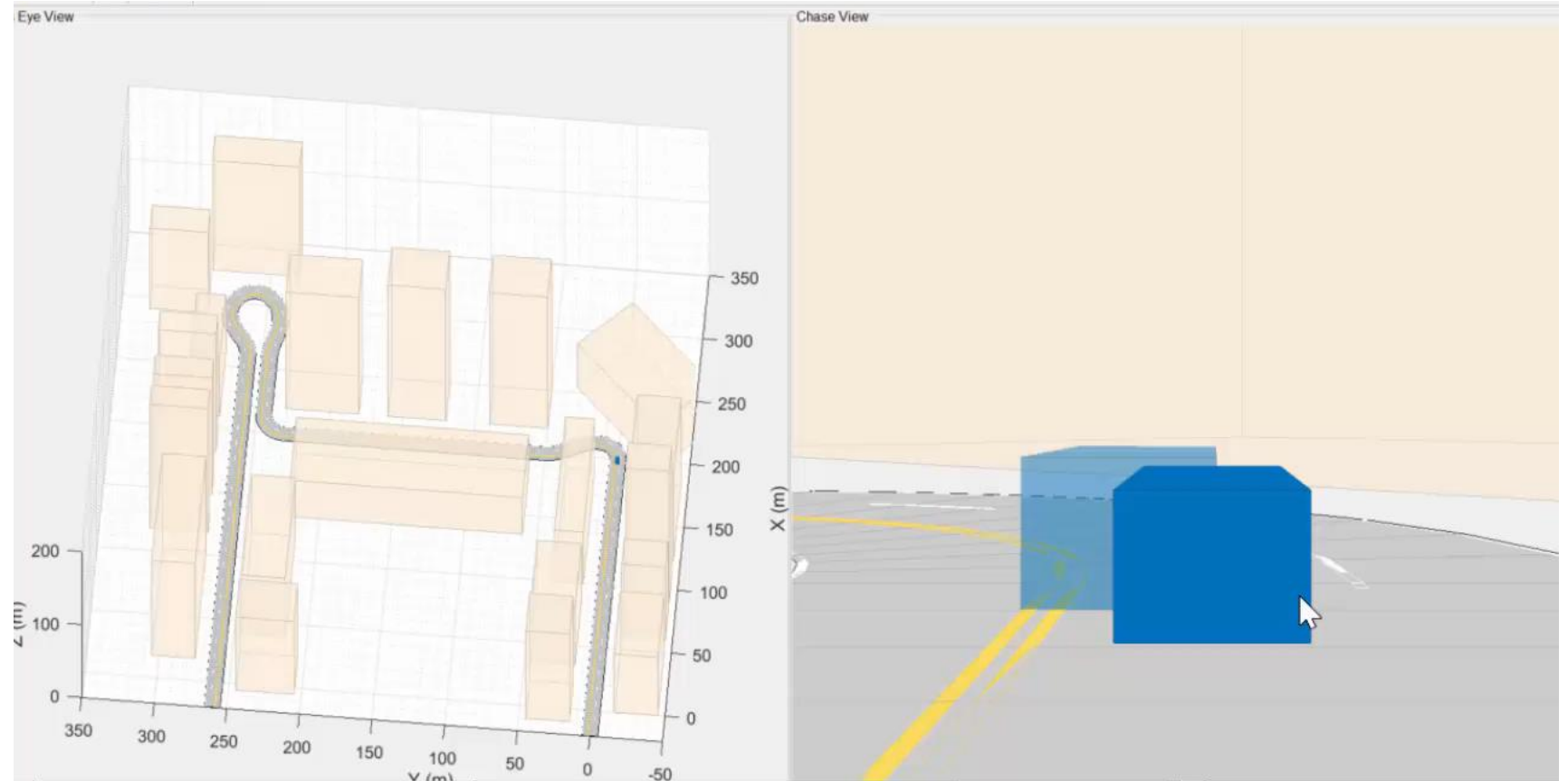
Fuse:

Accelerometer

Gyroscope

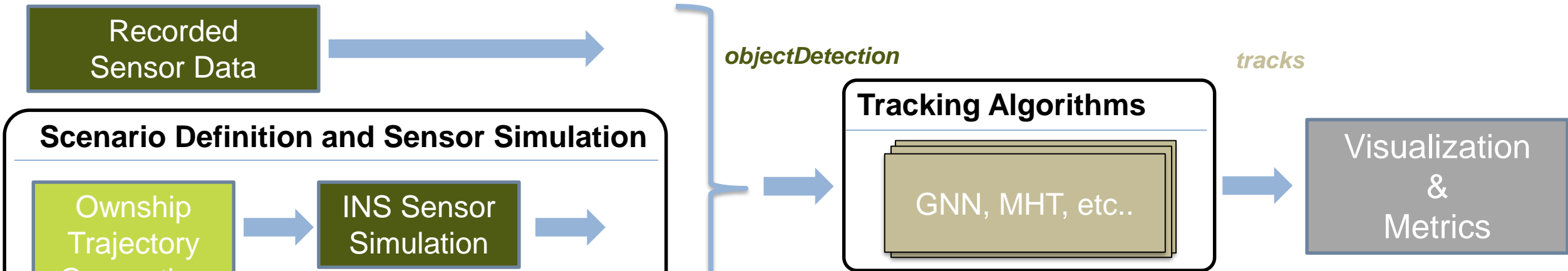
Camera (Visual Odometry)

Unconstrained Motion

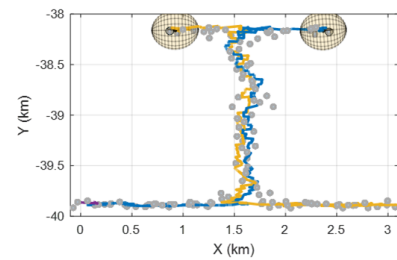
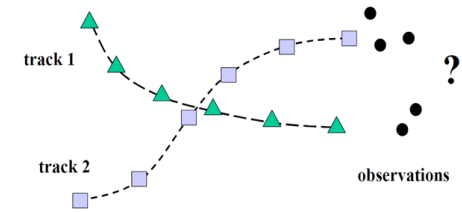
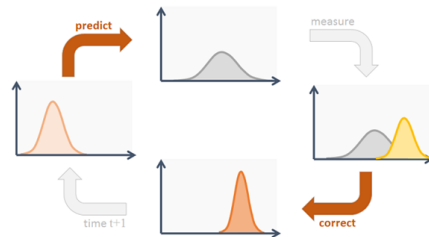


Multi-Target Tracking

Tracking Algorithm Development Workflow



A rich library of tracking algorithms



Filters

- Alpha Beta filter
- Kalman filters
 - Linear, EKF, UKF, CKF, MSCEKF
- Particle filter
- Multiple models
 - GSF, IMM

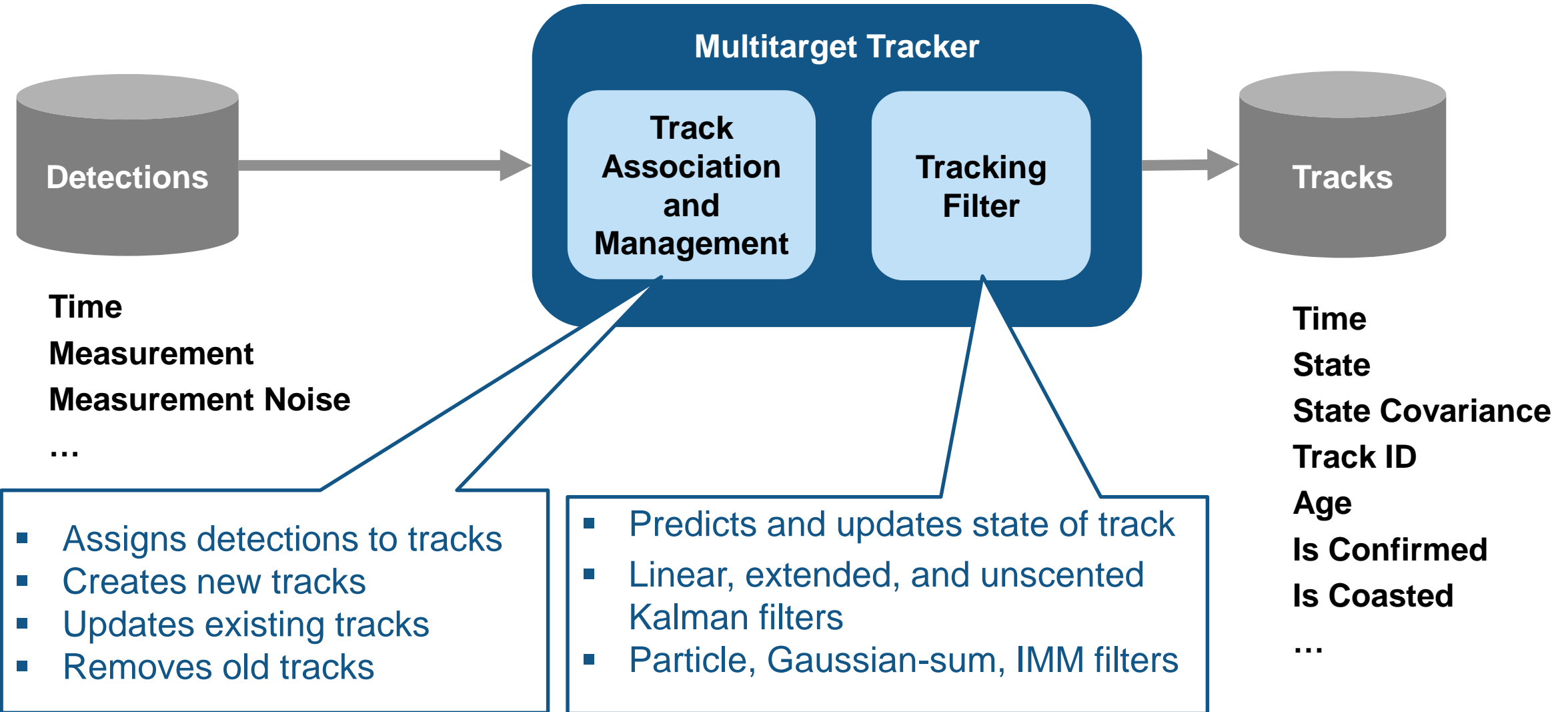
Data Association

- 2D assignment
- S-D assignment
- K-best assignment

Trackers

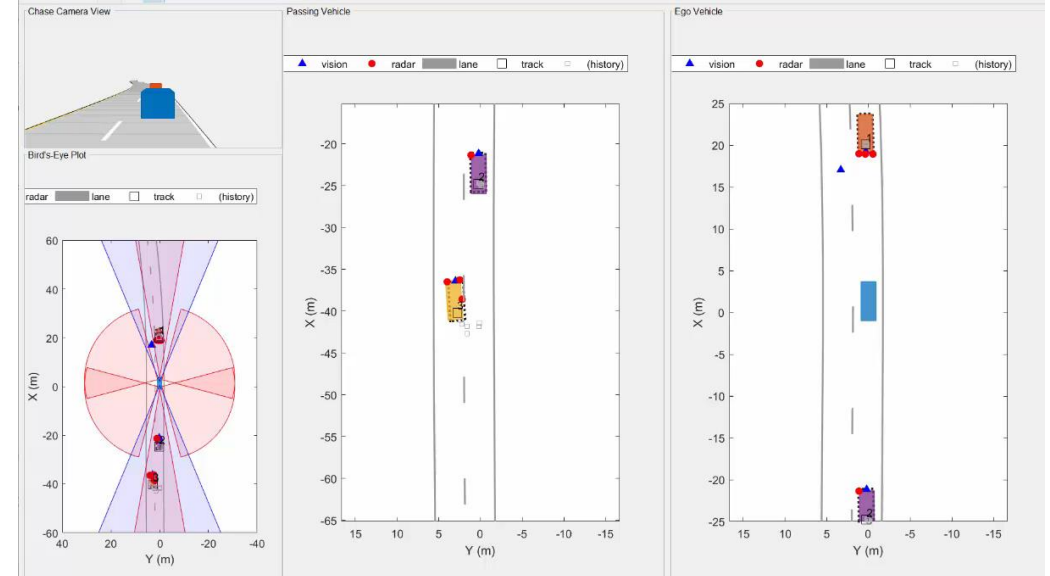
- GNN
- MHT (track-oriented)
- Trackers components
 - History and score logic
 - etc.....

Difference Between Multitarget Tracker and Kalman Filter



Tracking extended objects

- High resolution sensors (radar, lidar)
- Close-range targets



Point object vs. Extended object

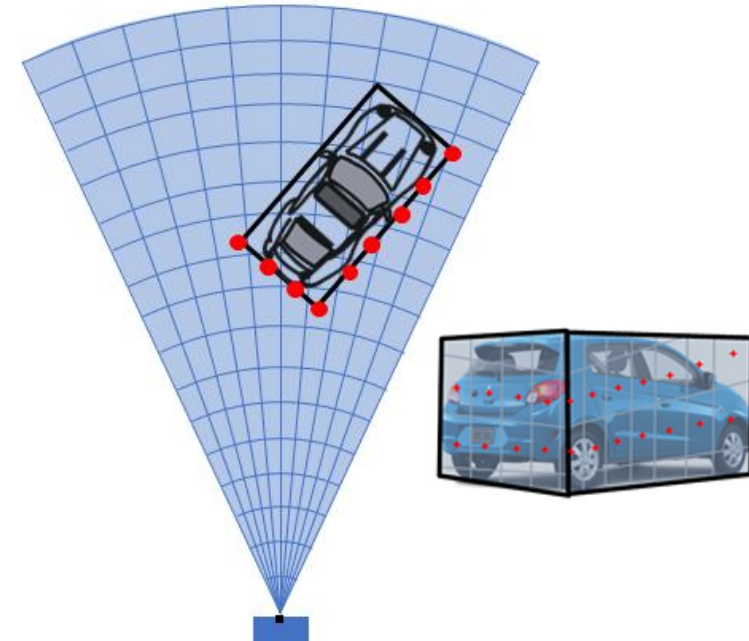
- **Point object**

- Distant object represented as a single point
- One detection per object per scan

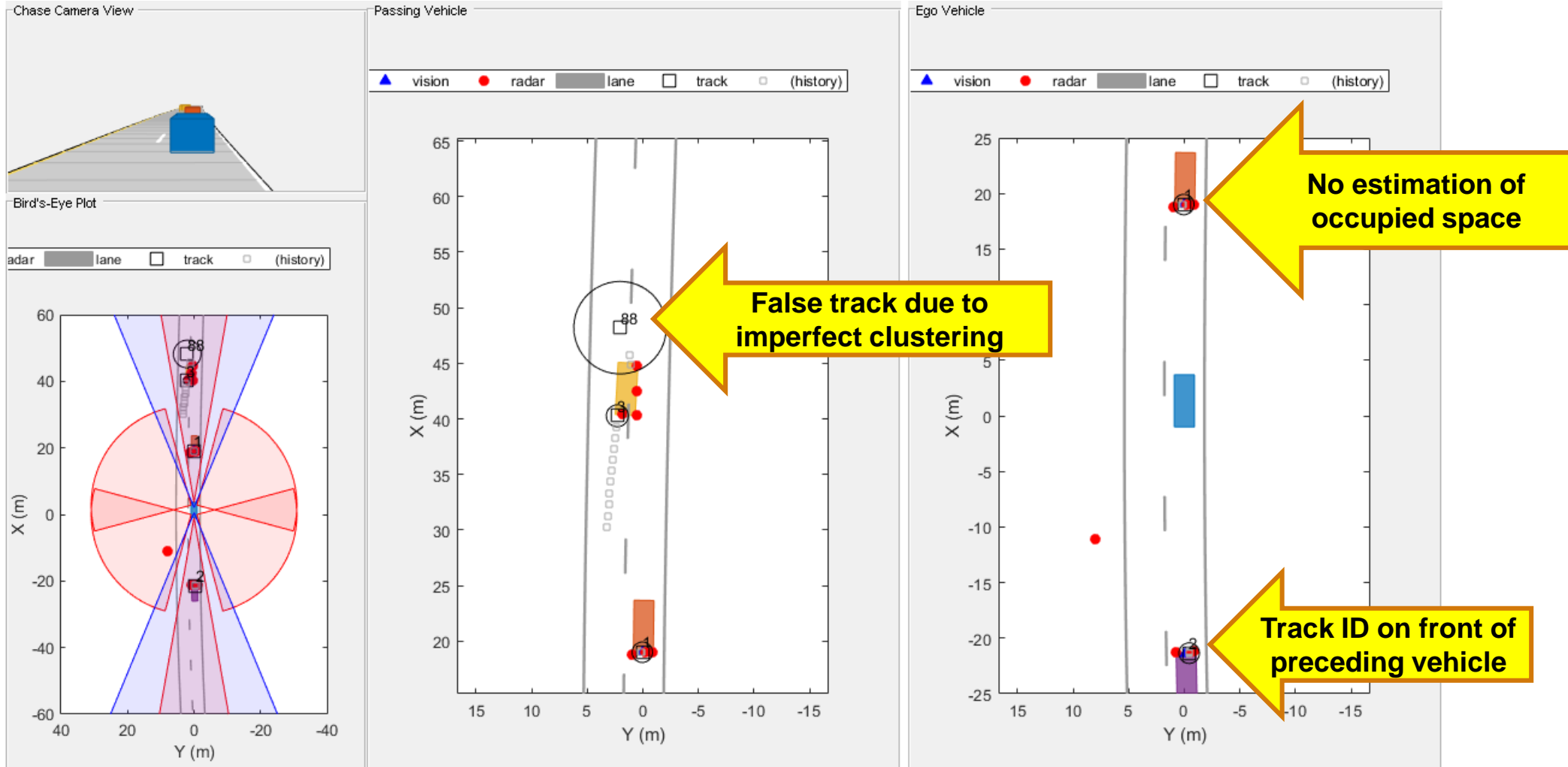


- **Extended object**

- High resolution sensors generate multiple detections per object per scan



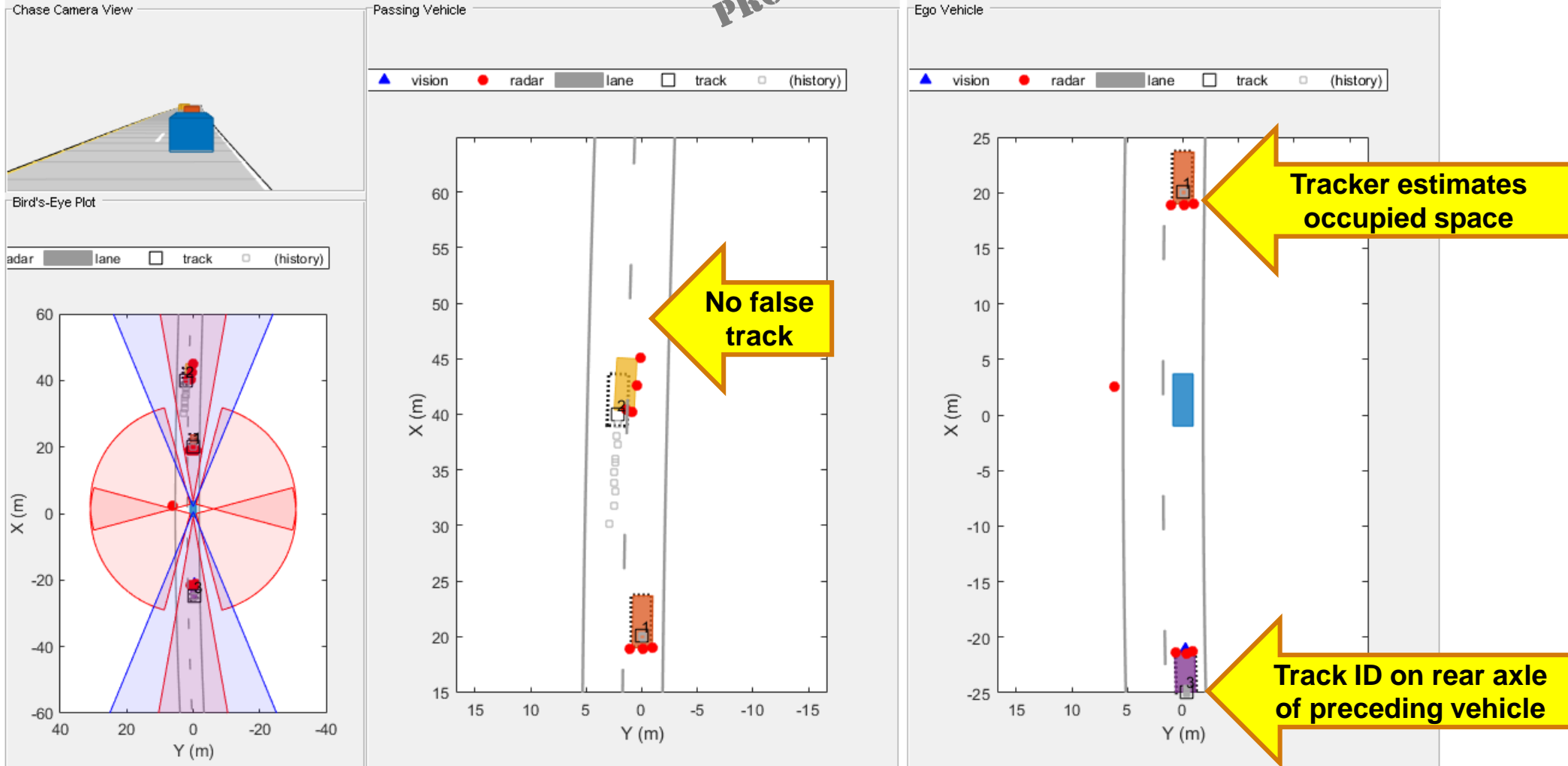
Tracking Extended Objects with Point Trackers



Point cloud changing with aspect angle -> challenging clustering problem

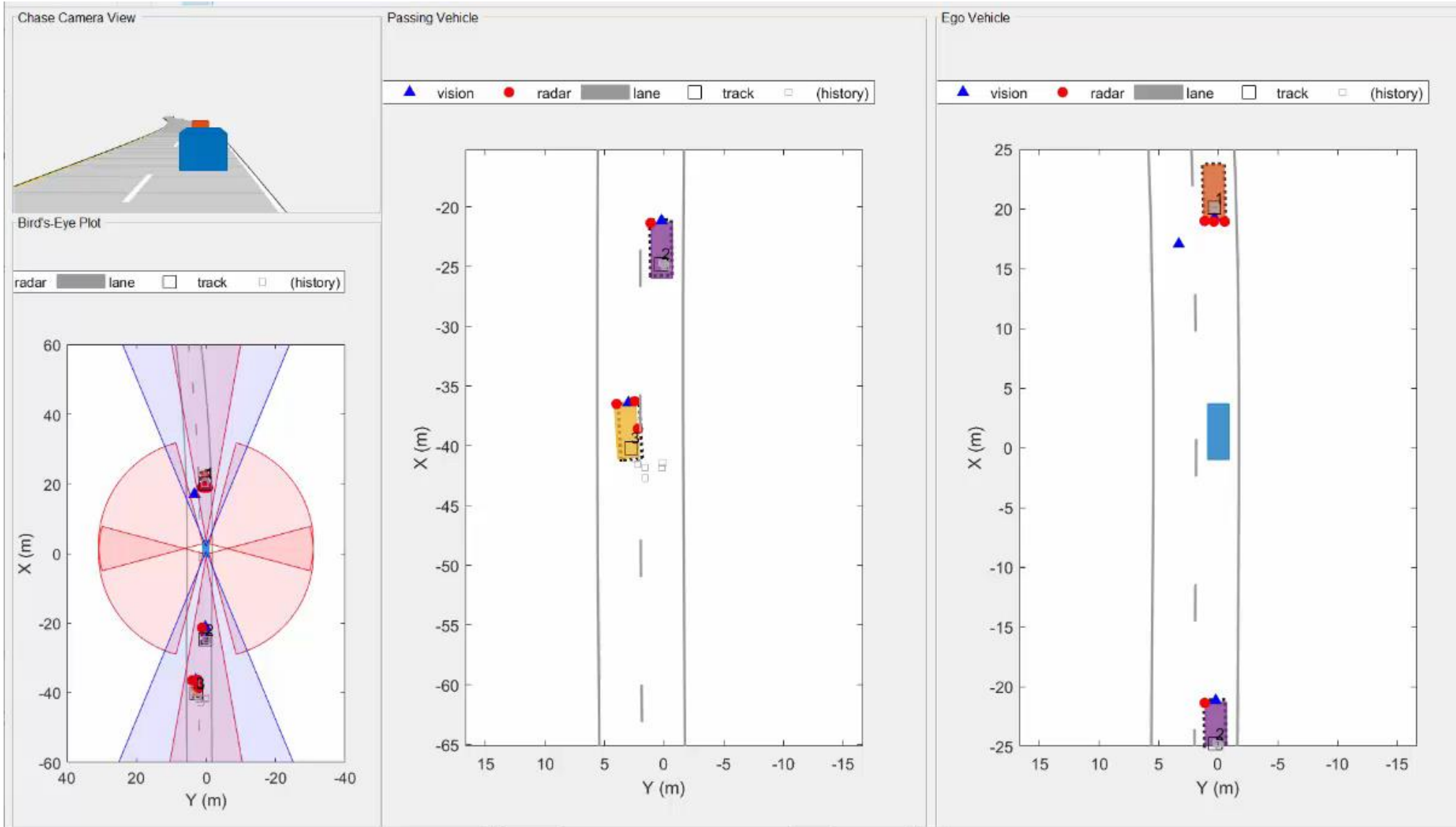
Tracking Extended Objects: Estimate Position, Velocity, and Shape

PROTOTYPE



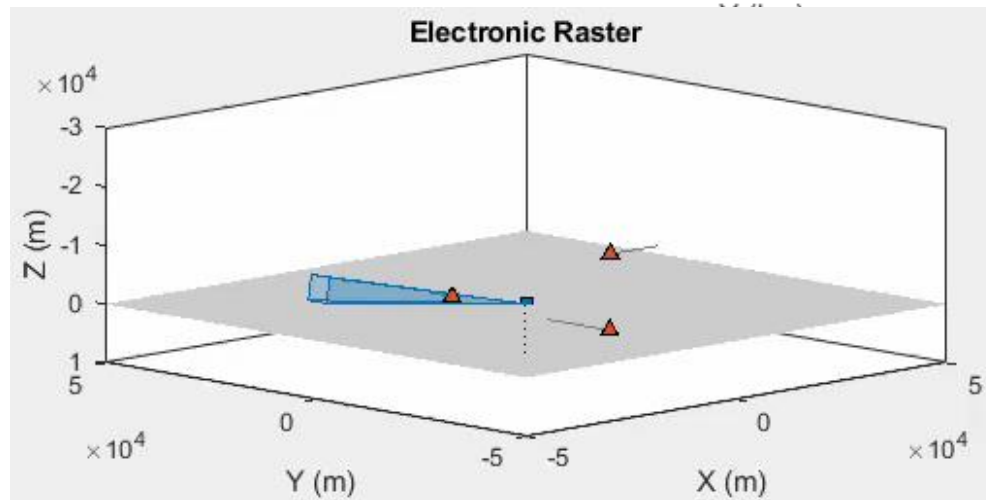
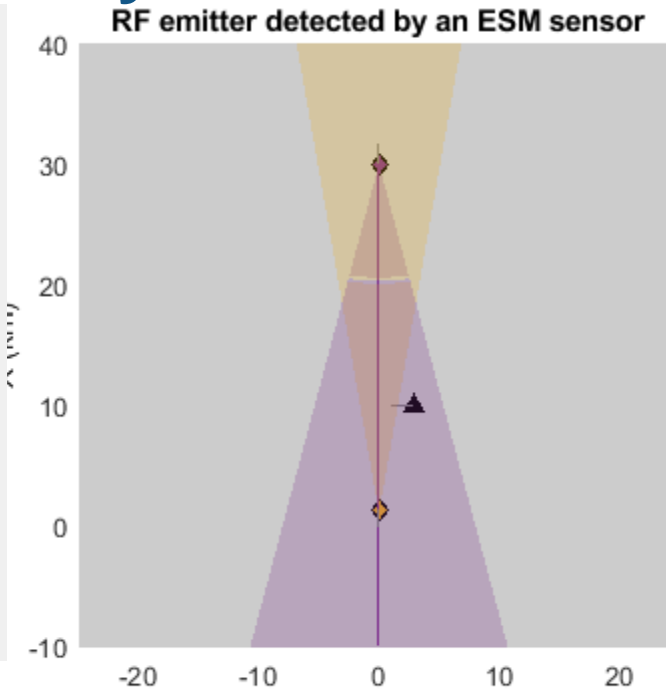
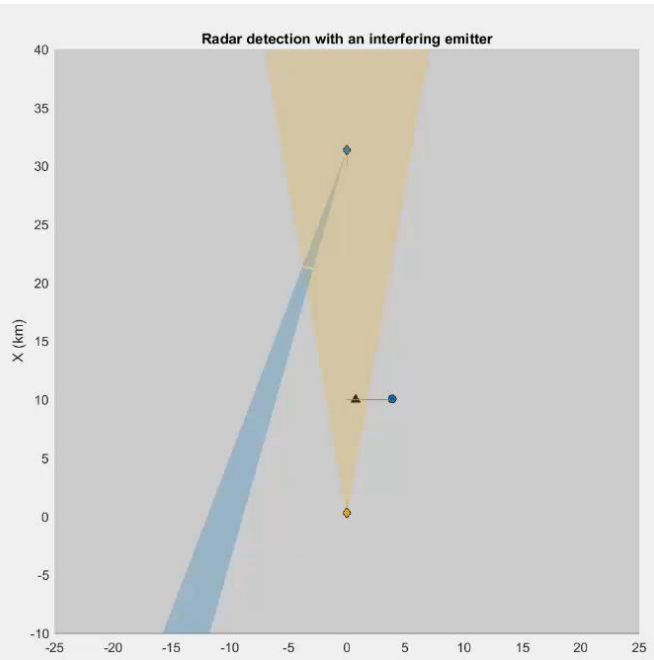
No clustering needed, Consistent tracks

Tracking Extended Objects: Estimate Position, Velocity, and Shape



Sensor Models Beyond Automated Driving

Sensor Models Beyond Automated Driving



Sensor Models

IMU, GPS, RADAR, ESM, and EO/IR

Model various sensors, including: IMU (accelerometer, gyroscope, magnetometer), GPS receivers, radar, sonar, and channels as separate objects.

Functions

> GPS

> IMU

> Infrared

> INS

> Radar

> Sonar

Getting Started



[Products](#)
[Solutions](#)
[Academia](#)
[Support](#)
[Community](#)
[Events](#)

Sensor Fusion and Tracking Toolbox **NEW PRODUCT**

Search MathWorks.com



[Trial software](#)
[Contact sales](#)

Sensor Fusion and Tracking Toolbox
Design and simulate multisensor tracking and navigation systems

[Download a free trial](#)

T2
 A:4029m→0m/s
 285 km/hr H:359 deg

T3
 A:3082m→1m/s
 890 km/hr H:359 deg

T1
 A:4119m↑5m/s
 701 km/hr H:90 deg

T1
 A:4119m 5m/s
 701 km/hr H:90 deg

- Ground
- ATC Tower
- Detections
- Targets
- Tracks
- (history)