Principles of Robot Autonomy I

Multi-sensor perception and sensor fusion II

Daniel Watzenig

Today's lecture

- Aim
  - Introduce the topic of sensor fusion for multi-sensor perception
  - Learn about fundamental ideas of multi-object tracking for autonomous systems
  - Sensor fusion of radar and visual data (high-level fusion)

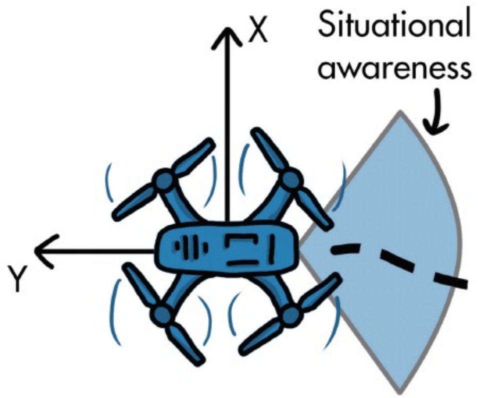
- Topics
  - Introduction: Understanding tracking filters, measurement noise, and process noise
  - Single-object tracking: Using a tracker to determine position and motion of a remote object
  - Multi-object tracking: Overcoming the challenges of tracking several objects at once

- Readings
  - Blackman S. S., and Popoli R., *Design and analysis of modern tracking systems*, 1999.
  - MathWorks, *Multi-Object Tracking for Autonomous Systems and Surveillance Systems*, 2020.

Part 1: Introduction

Perception – critical component of autonomous system

Multi-object tracking and ***sensor fusion*** are core of a perception system



The system needs to be able to maintain situational awareness.

Part 1: Introduction

Multi-object tracking - the core is the ability to estimate the motion of each object separately

Estimation filters - different types are used in tracking

- the most fundamental and simple filter is the **Kalman filter**

Kalman filter - uses a two-step process to estimate state:

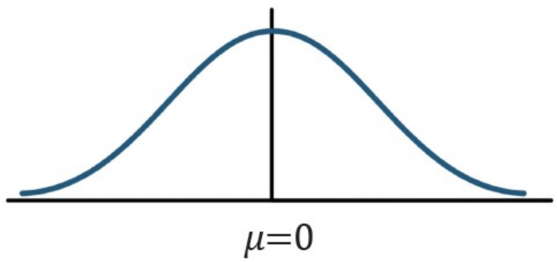
1. predict

2. correct

1. Linear system model: $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$

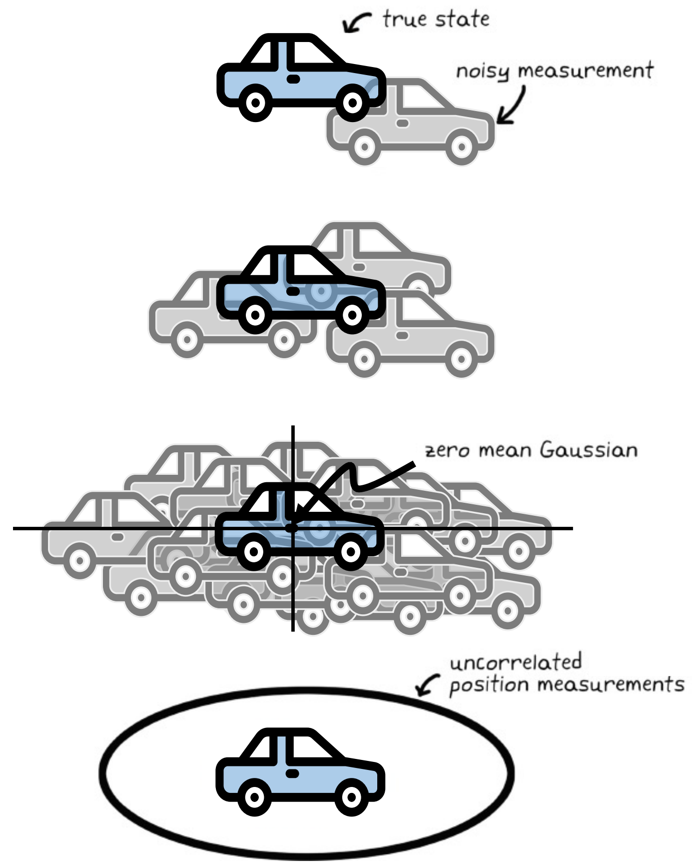
$$z_t = C_t x_t + \delta_t$$

2. Gaussian noise distribution:

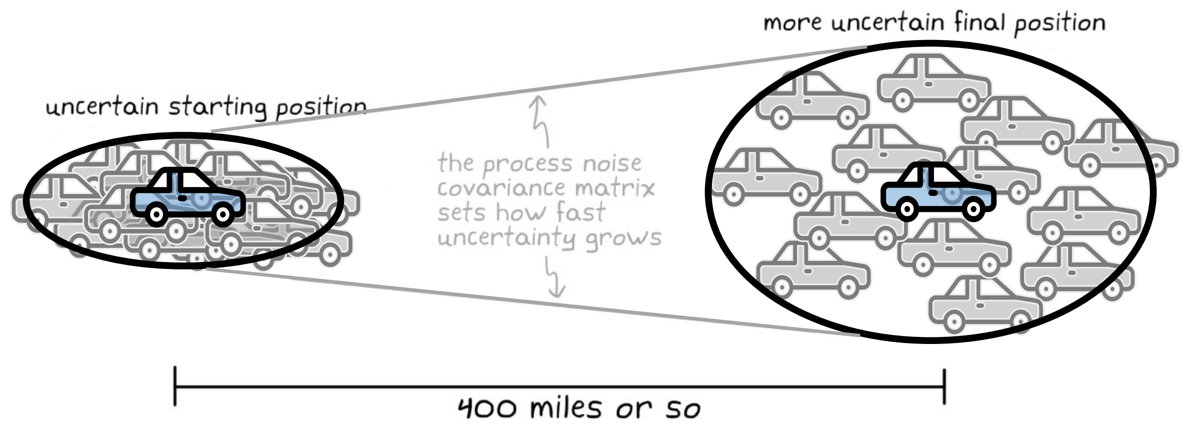


Part 1: Introduction

Measurement noise

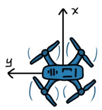


Process noise



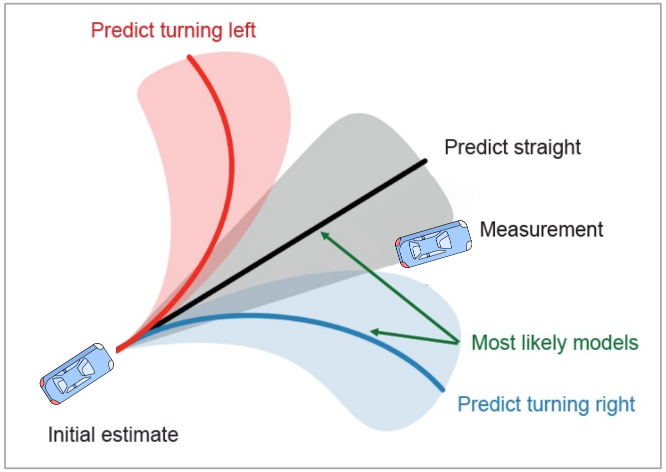
Combining prediction and measurement to get more accurate and more reliable estimates of a system state.

Part 2: Single-object tracking

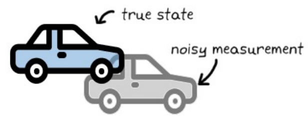
from → **What is the state of the system?**  to → **What is the state of a remote object?**  → Task is to determine state (e.g., position or velocity) by fusing the results from sensors and models

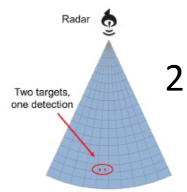
Tracking becomes more challenging:

a) Predicting the state of a tracked object

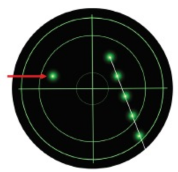


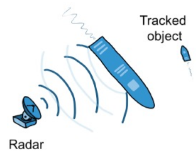
b) Challenges in remote measurements

1. Inaccurate measurements 



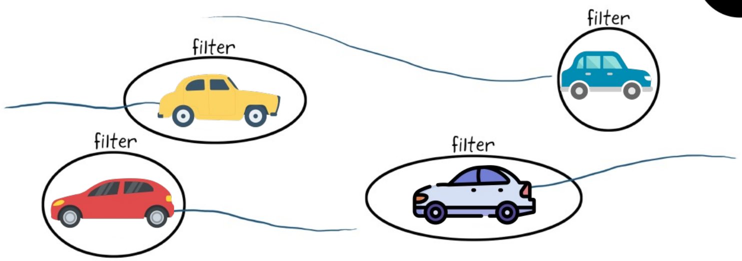
2. Ambiguous measurements


3. False alarms

4. Target occlusion 

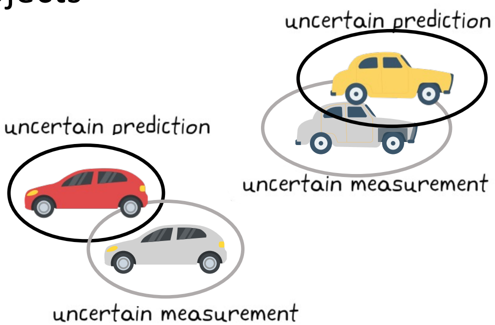
Part 3: Multi-object tracking

Is the tracking of multiple objects at the same time tougher than tracking a single object?

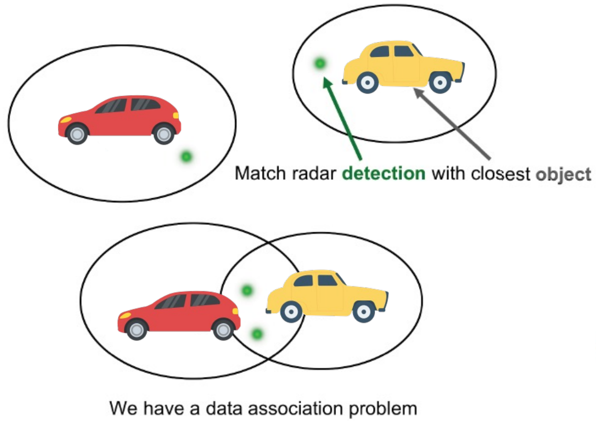


The difficulty of multi-object tracking

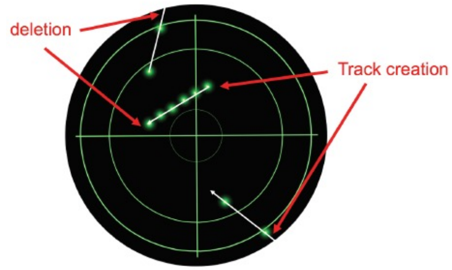
1. Uncertainties in predictions and in measurements of the objects



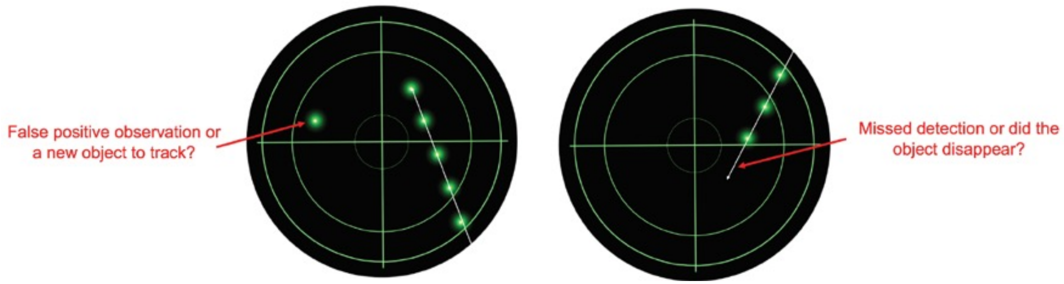
2. Data association problem



3. Track maintenance



4. Track maintenance due to uncertainties



Part 3: Multi-object tracking

When tracking multiple objects:

- What are the ways to approach the data association problem?
- What are the ways to address the track maintenance problem?

Multi-object tracking flow chart

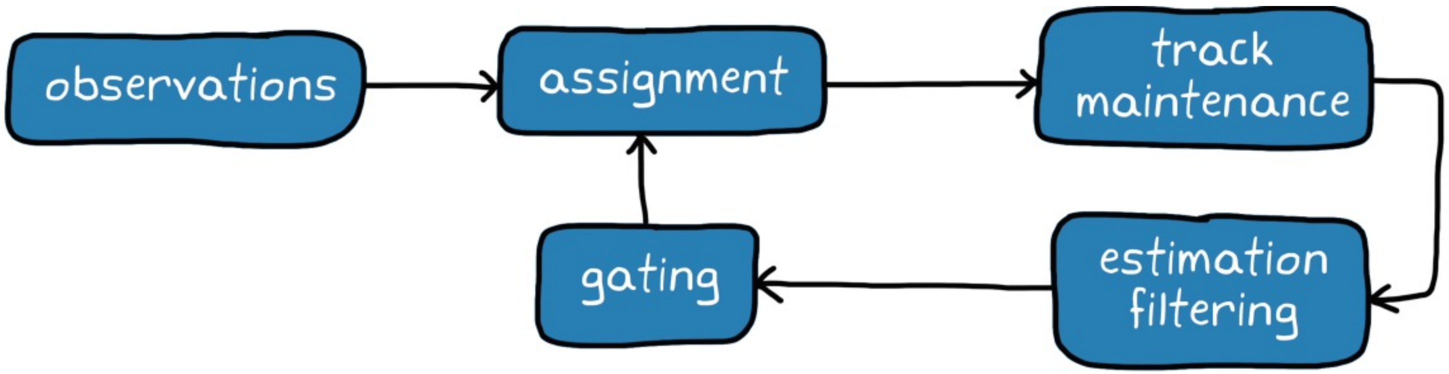


Figure adapted from Design and Analysis of Modern Tracking Systems by Samuel Blackman and Robert Popoli (Artech House Radar Library).

Part 3: Multi-object tracking

Recall: Kalman Filter from previous lectures

Description of the system and the measurement models:

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

$$z_t = C_t x_t + \delta_t$$

- Independent process noise ϵ_t is $\mathcal{N}(0, R_t)$
- Independent measurement noise δ_t is $\mathcal{N}(0, Q_t)$

Kalman filter equations

Prediction

Project state ahead

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

Project covariance ahead

$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

Correction

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

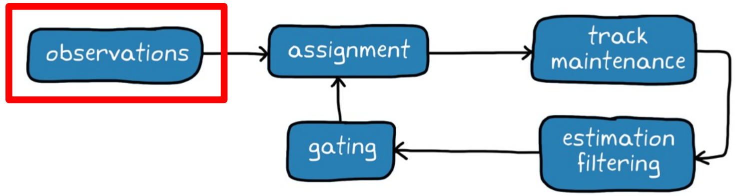
Update estimate with new measurement

$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$

Update covariance

$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

Part 3: Multi-object tracking



Observation or detection occurs when the sensor measures an object

Measurement vector within the Kalman filter structure:

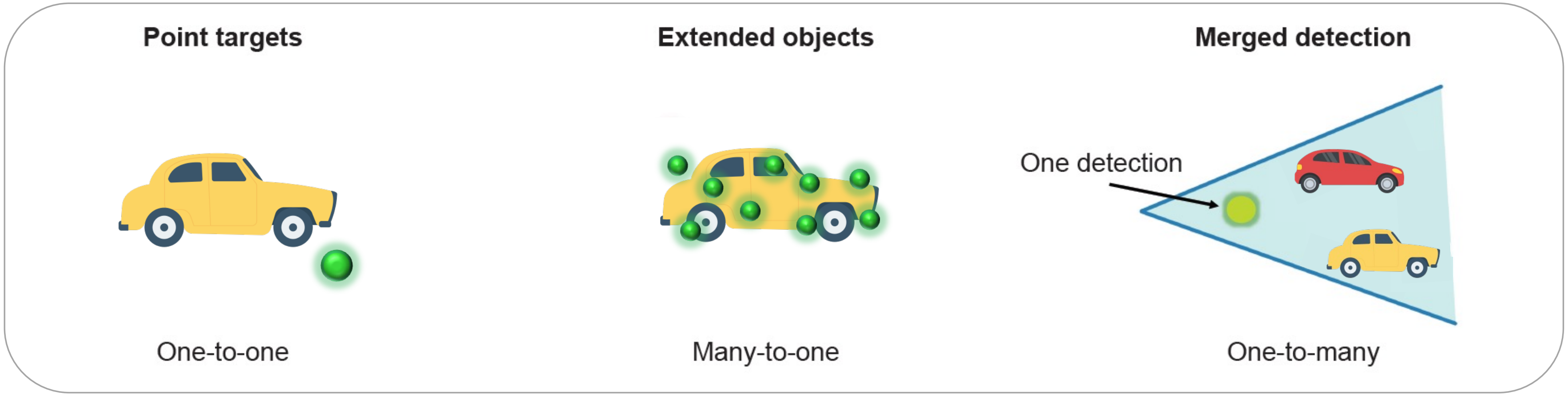
$$z_t = \begin{bmatrix} \text{speed} \\ \text{heading} \\ \dots \end{bmatrix}_t = \begin{bmatrix} 400 \\ 17 \\ \dots \end{bmatrix}_t$$

Measured quantities, e.g., speed or heading

Measured attributes, e.g., color or car type

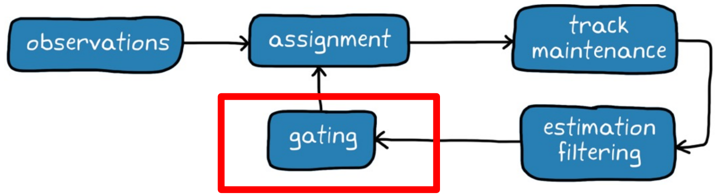
Useful for improving the track confirmation performance

Types of observations



*Note: only point targets will be discussed

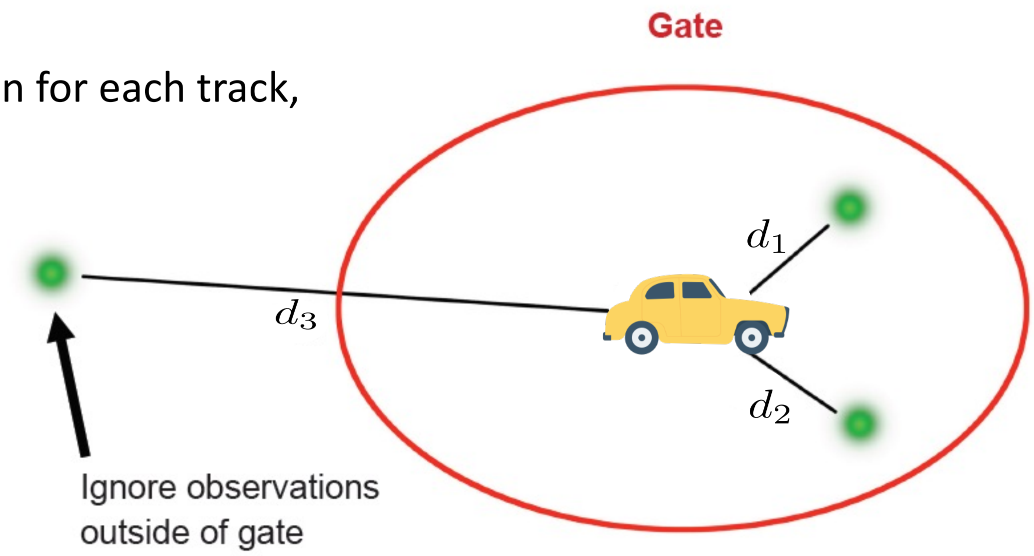
Part 3: Multi-object tracking



Computational challenge to look at every observation and consider how likely it is to be assigned to every track

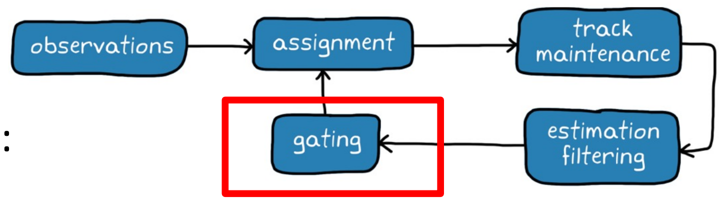
Gating - screening mechanism

- ignoring observations outside of a specific region for each track,
- speeds up the assignment process.



Gating impacts the assignment algorithms – they consider only the observations that are worth looking at.

Part 3: Multi-object tracking



- Residual (or innovation) vector and its covariance matrix are defined as:

$$\tilde{z}_t = z_t - C_t\bar{\mu}_t$$

$$S_t = C_t\bar{\Sigma}_t C_t^T + Q_t$$

- If the measurement is of dimension *M*, the *M*-dimensional Gaussian probability density for the residual is:

$$p(\tilde{z}_t) = det((2\pi)^M S_t)^{-\frac{1}{2}} exp(-\frac{1}{2}\tilde{z}_t^T S_t^{-1}\tilde{z}_t)$$

1. Rectangular gates

The simpliest gating technique – an observation satisfies the gates of a given track if all elements $\tilde{z}_l$ of the residual vector $\tilde{z}_t$ satisfy:

$$\boxed{|\tilde{z}_l| \leq K_{Gl}\sigma_r}$$

➢ Residual standard deviation is defined as:

$$\sigma_r = \sqrt{\sigma_o^2 + \sigma_p^2}$$

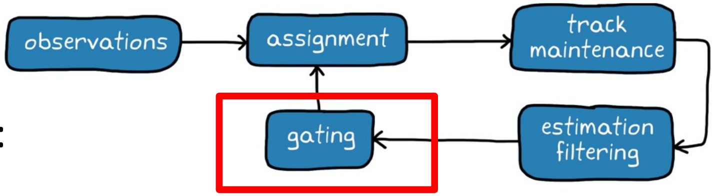
Prediction variance (appropriate diagonal element taken from the KF covariance matrix)

Measurement variance

➢ Typical choice of rectangular gating coefficients is:

$$K_{Gl} \geq 3.0$$

Part 3: Multi-object tracking



- Residual (or innovation) vector and its covariance matrix are defined as:

$$\tilde{z}_t = z_t - C_t \bar{\mu}_t$$

$$S_t = C_t \bar{\Sigma}_t C_t^T + Q_t$$

- If the measurement is of dimension *M*, the *M*-dimensional Gaussian probability density for the residual is:

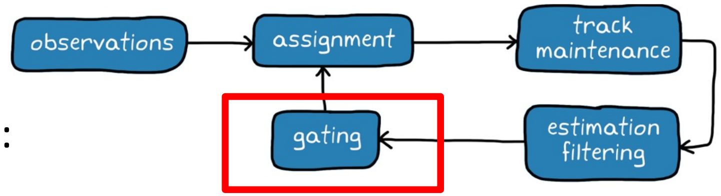
$$p(\tilde{z}_t) = det((2\pi)^M S_t)^{-\frac{1}{2}} exp(-\frac{1}{2} \tilde{z}_t^T S_t^{-1} \tilde{z}_t)$$

2. Ellipsoidal gates

The measurements will be in the area $\boxed{d^2 = \tilde{z}_t^T S_t^{-1} \tilde{z}_t \leq G}$ with a probability defined by the gate threshold *G*.

- This area is called **validation gate**. The shape of the validation gate is a hyper-ellipsoid (an ellipse in 2d)
- *G* is taken from the inverse χ^2cumulative distribution at a level α and *M* degrees of freedom
- Typical values for α are 0.95 or 0.99
- The validation gate is a **region of acceptance** such that $100(1-\alpha)\%$ of true measurements are **rejected**.

Part 3: Multi-object tracking



- Residual (or innovation) vector and its covariance matrix are defined as:

$$\tilde{z}_t = z_t - C_t \bar{\mu}_t$$

$$S_t = C_t \bar{\Sigma}_t C_t^T + Q_t$$

- If the measurement is of dimension *M*, the *M*-dimensional Gaussian probability density for the residual is:

$$p(\tilde{z}_t) = det((2\pi)^M S_t)^{-\frac{1}{2}} exp(-\frac{1}{2} \tilde{z}_t^T S_t^{-1} \tilde{z}_t)$$

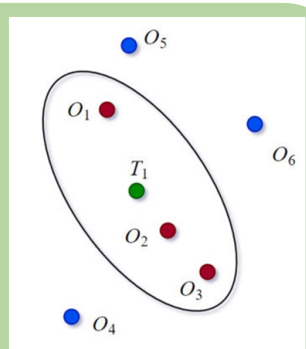
2. Ellipsoidal gates

The measurements will be in the area $\boxed{d^2 = \tilde{z}_t^T S_t^{-1} \tilde{z}_t \leq G}$ with a probability defined by the gate threshold *G*.

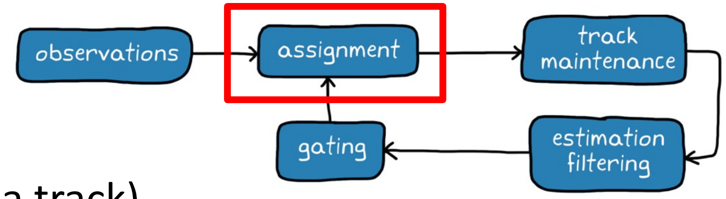
- If $\mathrm{d}^2 \leq \mathrm{G}$: detection is inside the gate of the track, and it will be considered for association.
- If $\mathrm{d}^2 > \mathrm{G}$: the possibility of the detection associated with the track is removed.

Example: T_1 is the predicted track estimate, while $O_1 - O_6$ are six detections.

Based on the gating result, O_1, O_2 and O_3 are within the validation gate.



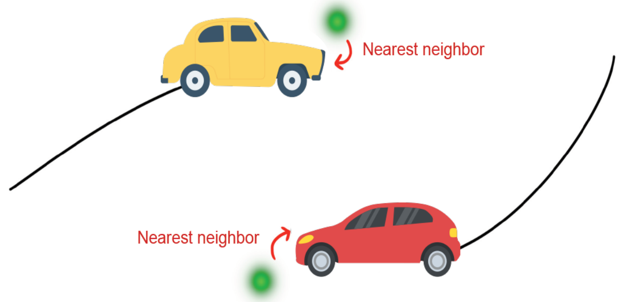
Part 3: Multi-object tracking



One or more sensors generate multiple detections from multiple targets in a scan.

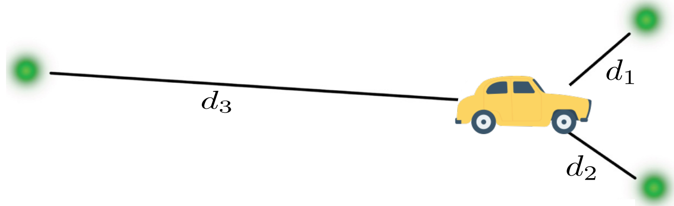
Assignment is the process of matching an observation to a tracked object (a track).

Assigning detections is very challenging:
- the number of targets or detections is large
- conflicts between different assignment hypotheses



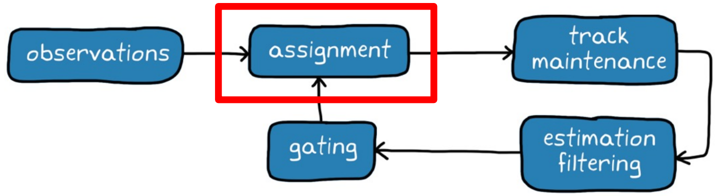
Assignment problems, depending on the dimension, are categorized into:
- 2-D assignment problem – assigns n targets to m observations
- S-D assignment problem – assigns n targets to a set ($m1$, $m2$, $m3$, …) of observations

2-D assignment approaches will be explained:
- ✓ GNN – adopts a global nearest data assignment approach
- ✓ JPDA - adopts a joint probability data association approach



More about other assignment algorithms, e.g.,
https://www.mathworks.com/help/fusion/multi-object-trackers.html

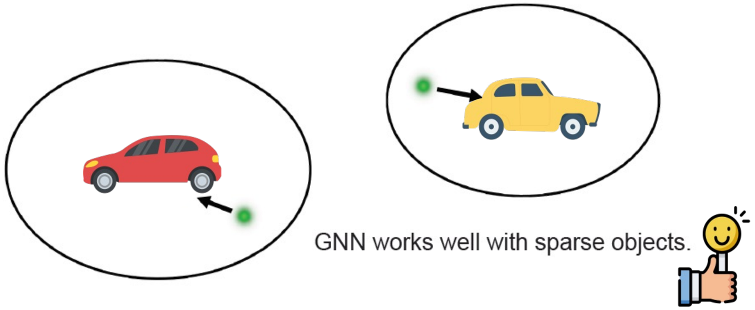
Part 3: Multi-object tracking



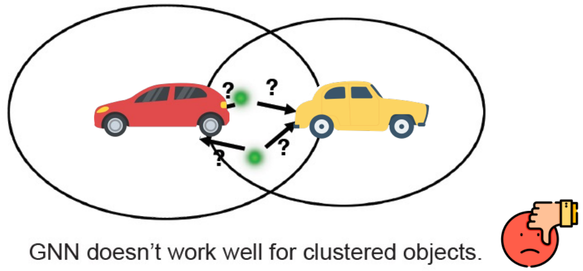
1. *Global Nearest Neighbor (GNN) Method*

- a single hypothesis assignment method
- the goal is to:
  - assign the global nearest observations to existing tracks and
  - create new track hypotheses for unassigned detections

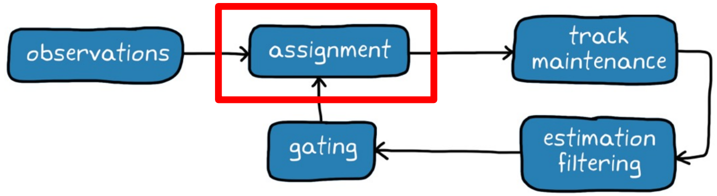
GNN assignment problem:
- easily solved if there are no conflicts of association between tracks.
- tracker assigns a track to its nearest neighbor



Conflict situations:
- when there is more than one observation within a track's validation gate or
- an observation is in the gates of more than one track



Part 3: Multi-object tracking



1. *Global Nearest Neighbor (GNN) Method*

➢ To resolve the conflicts, the tracker must evaluate a cost matrix.

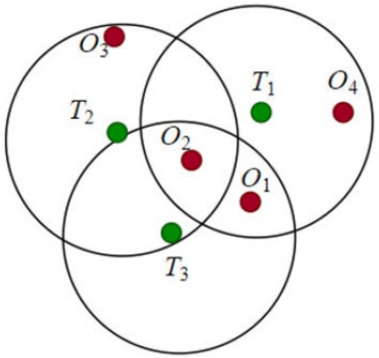
Define a *generalized statistical distance* d_{Gij}^2 between observation *j* to track *i* as:

$$d_{Gij}^2 = d_{ij}^2 + \ln[|S_{ij}|]$$

logarithm of the determinant of the residual covariance matrix (used to penalize tracks with greater prediction uncertainty)

Mahalanobis distance

Example:



Tracks	Observations			
	O_1	O_2	O_3	O_4
T_1	9	6	X	**6**
T_2	X	**3**	10	X
T_3	**8**	4	X	X

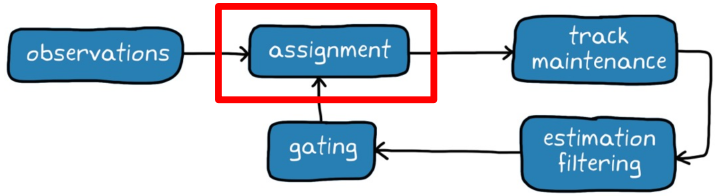
- Table shows a hypothetical cost matrix
- Optimal solutions are highlighted,
- Non-allowed assignments denoted by X.

Detection O_3:

➢ unassigned,

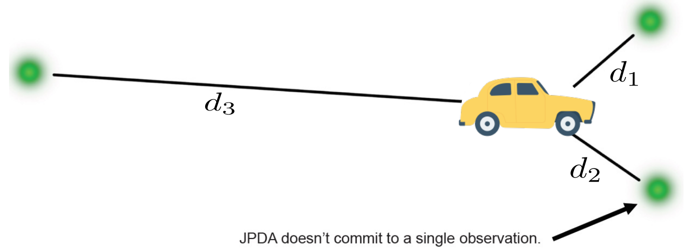
➢ the tracker creates a new tentative track

Part 3: Multi-object tracking



2. *Joint Probabilistic Data Association (JPDA) Method*

- applies a soft assignment,
- detections within the validation gate of a track make a weighted contributions to the track based on their probability of association.



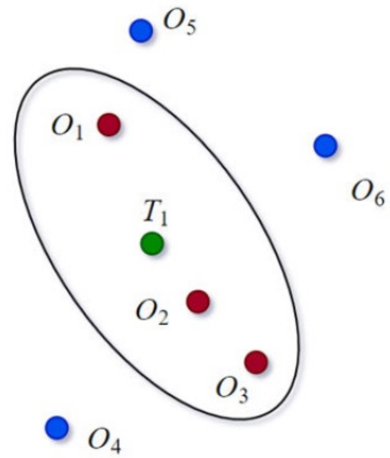
Example: JPDA tracker calculates the possibility of association between track T_1 and observations O_1, O_2 and O_3

Weighted sum of the residuals associated with track T_1:

$$\tilde{z}_1 = \sum_{j=1}^{3} p_{1j} \tilde{z}_{1j}$$

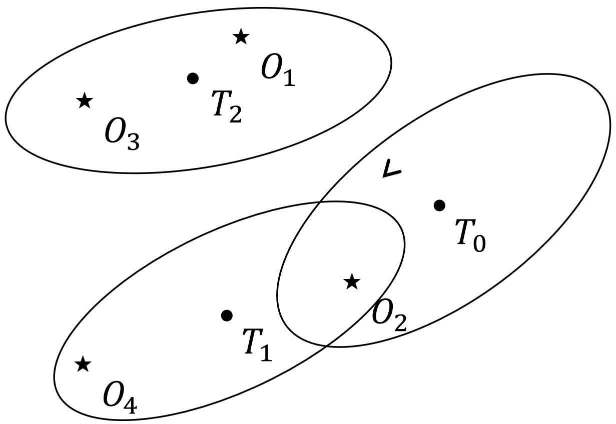
$\tilde{z}_1$ is used to update track T_1 in the correction step of the tracking filter

- p_{11}, p_{12}, p_{13} are association probabilities of the three observations
- $\tilde{z}_{11}, \tilde{z}_{12}, \tilde{z}_{13}$ are residuals relative to the track T_1



Part 3: Multi-object tracking

JPDA example



Association matrix

		Track number			
		-1	**0**	**1**	**2**
Measurement number	**1**	1	0	0	1
	2	1	1	1	0
	3	1	0	0	1
	4	1	0	1	0

Hypotheses 0…N

p=0.1

		Track number			
		-1	**0**	**1**	**2**
Measurement number	**1**	1	0	0	0
	2	1	0	0	0
	3	1	0	0	0
	4	1	0	0	0

p=0.4

		Track number			
		-1	**0**	**1**	**2**
Measurement number	**1**	0	0	0	1
	2	1	0	0	0
	3	1	0	0	0
	4	1	0	0	0

p=0.3

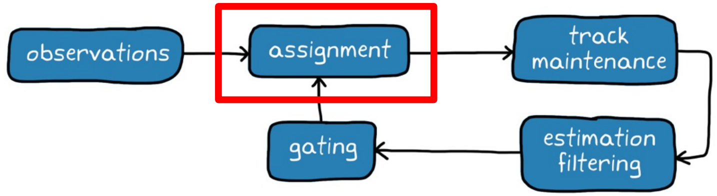
					2
					0
					0
					0
	4	1	0	0	0

Probabilities of measurement-to-track associations

		Track number			
		-1	**0**	**1**	**2**
Measurement number	**1**		0	0	0.7
	2		0.6	0.3	0
	3		0	0	0.2
	4		0	0.5	0

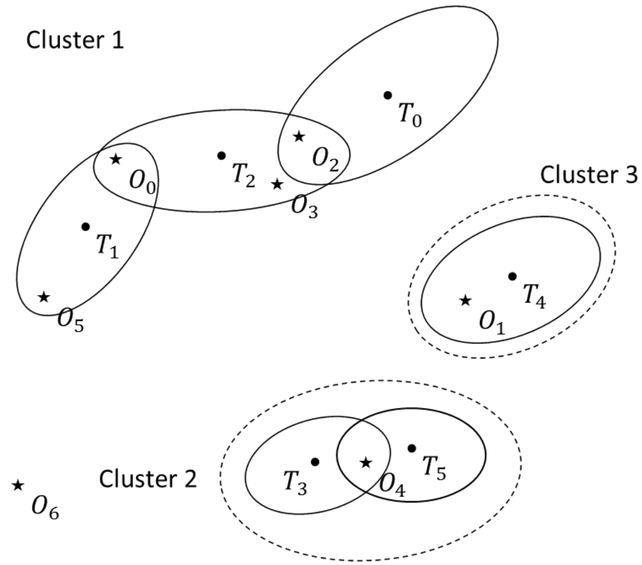
Example for track 1:
- 30 % that O_2 is correct
- 50 % that O_4 is correct
- 20 % that no measurement is correct

Part 3: Multi-object tracking

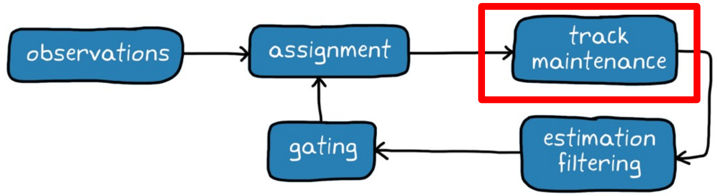


JPDA takeaways

- Bayes-based data association
- Fuse measurements **weighted by the probability** of measurement-to-track association
- Probability depends on:
  - Innovation, predicted measurement, and innovation covariance
  - Modeled probability of track detection
  - Modeled probability of clutter
  - Nonparametric version is used where all numbers of clutter measurements are equally likely
- Too many hypotheses? Clustering
- MHT, Extended object tracking (probability hypotheses densities, phd)



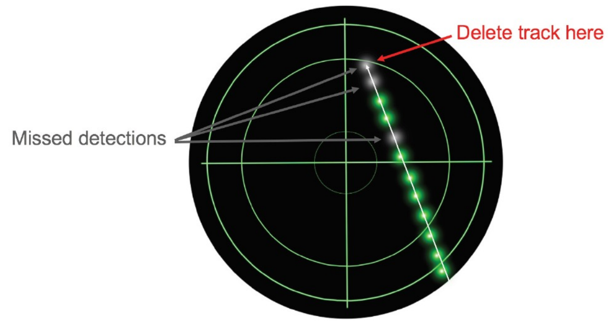
Part 3: Multi-object tracking



Track maintenance algorithms - used to delete and create tracks

1. Deleting a track

- only if it has not been assigned to a detection at least M times during the last N updates,
- M and N are tuning parameters.

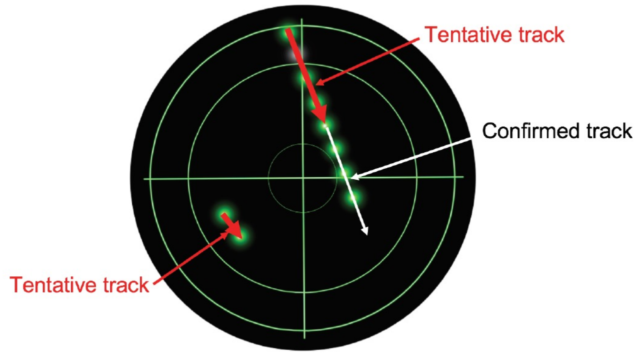


2. Creating a track

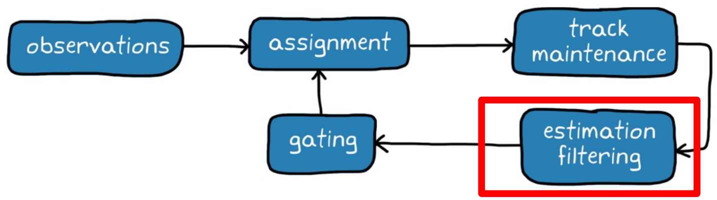
Is a single un-assigned observation a new object or not?

Create a tentative track— maintained but not treated as a real object

- confirmed - when detected M times in the last N updates,
- removed – the same logic as removing a confirmed track.

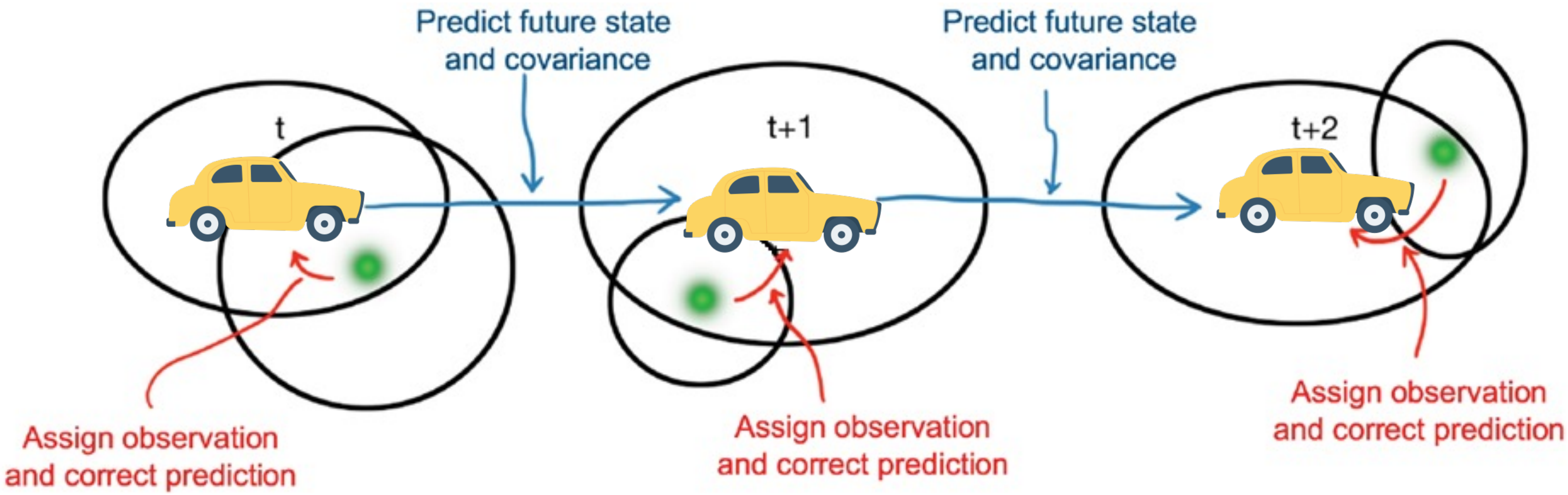


Part 3: Multi-object tracking



A set of estimation filters are running - one filter for each tracked object

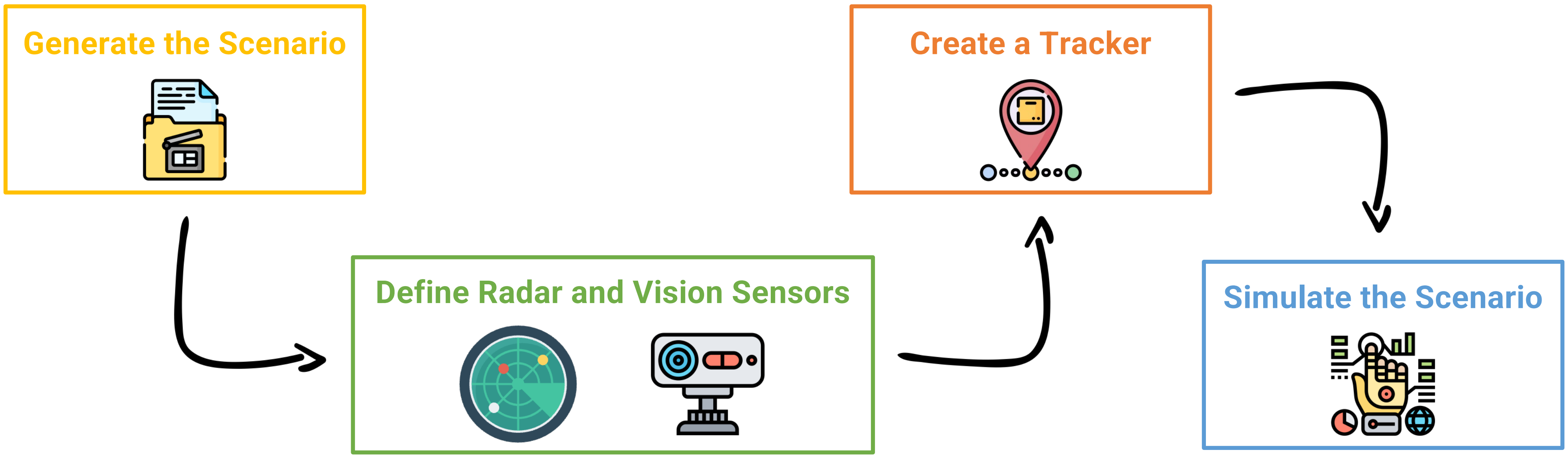
- identical to single-object tracking,
- different types of filters, e.g., interacting multiple model filter or the single model Kalman filter.



Part 4: Sensor fusion using radar and vision data

This example is part of the *Sensor Fusion and Tracking Toolbox (Matlab code but easily to transfer to e.g., Python)*

- shows how to generate a scenario, simulate sensor detections, and use sensor fusion to track simulated vehicles
- main benefit - ability to create rare and potentially dangerous events and test the vehicle algorithms with them



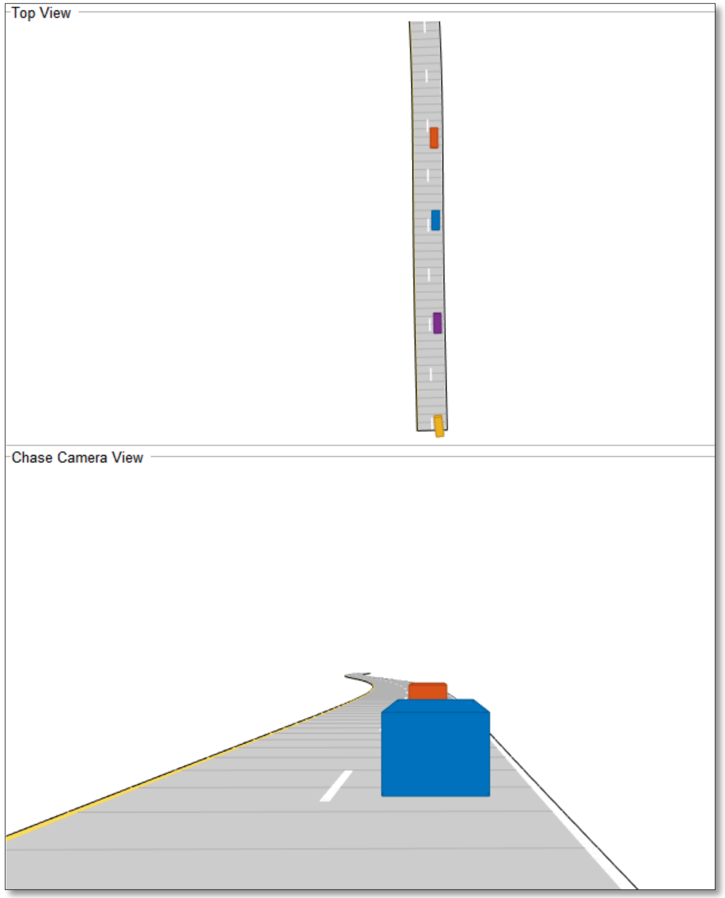
https://www.mathworks.com/help/driving/ug/sensor-fusion-using-synthetic-radar-and-vision-data.html

Part 4: Sensor fusion using radar and vision data

Generate the Scenario

Scenario generation comprises of:

- generating a road network,
- defining vehicles that move on the roads.

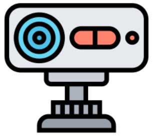


Scenario in this example:

- highway road with two lanes is defined,
- ego vehicle and three cars around it:
  - one overtakes the ego vehicle and passes it on the left,
  - one drives right in front of the ego vehicle,
  - one drives right behind the ego vehicle.

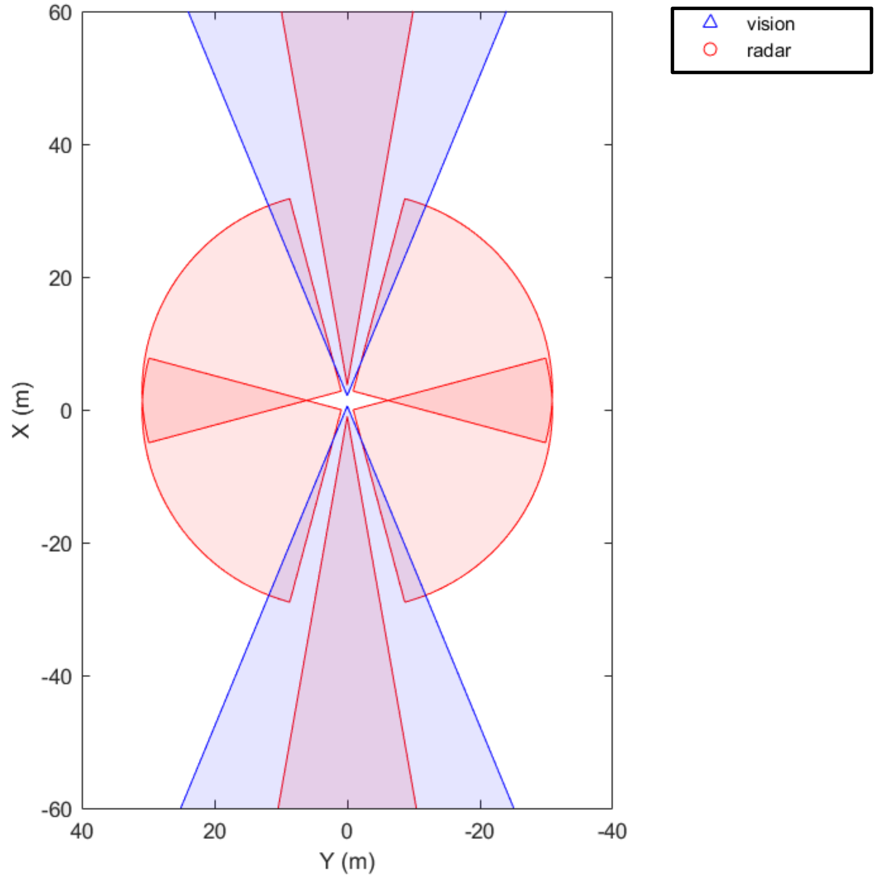
Part 4: Sensor fusion using radar and vision data

Define Radar and Vision Sensors



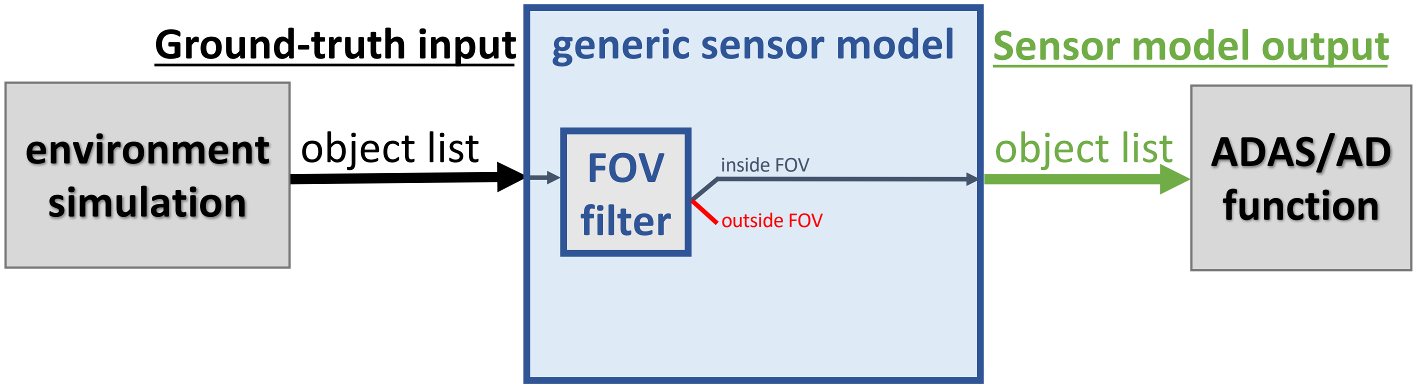
Sensors defined on the ego vehicle:

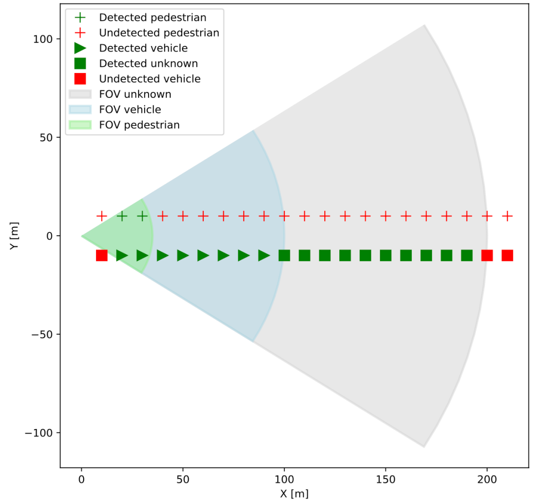
- 6 radar sensors:
  - 2 long-range radar sensors covering 20 degrees (in front and back),
  - 4 short-range radar sensors covering 90 degrees (two per side),
- 2 vision sensors:
  - Front-facing camera located at front windshield,
  - Rear-facing camera located at rear windshield,
- 360 degrees field of view is covered,
- sensors have some overlap and some coverage gap.



Part 4: Sensor fusion using radar and vision data

Basic functionality of a sensor model: FOV (field of view) filter

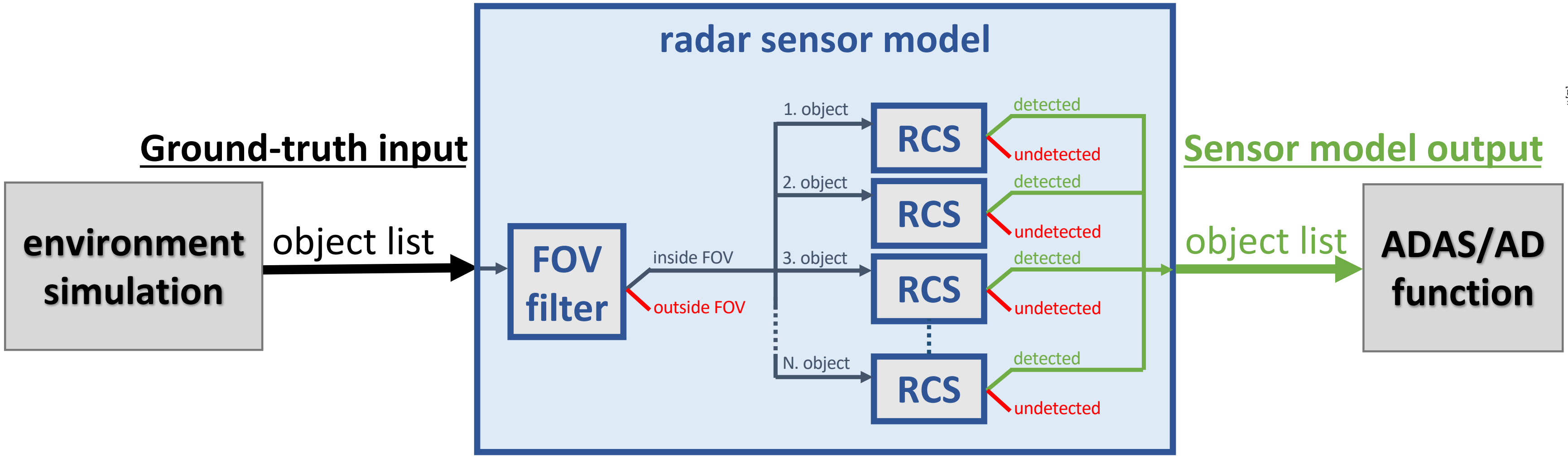




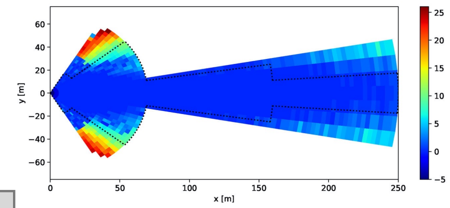
- **Boundary conditions:** maximum range r_{sensor} and opening angle φ_{sensor} of sensor's FOV
- **Input:** x, y-positions of objects in sensor's coordinate system
- **Task:** perform coordinate transformation and evaluate if $r_{target} < r_{sensor}$ and $\varphi_{target} < \varphi_{sensor}$
- **Output:** objects inside FOV

Part 4: Sensor fusion using radar and vision data

RCS (radar cross section) based radar model



$RCS_{min}[dBsm]$
Continental ARS 408-21



$RCS_{target}[dBsm]$
object types

pedestrian	1.74 $dBsm$
bike	3.46 $dBsm$
normal vehicle	8.98 $dBsm$
big vehicle	19.97 $dBsm$

- **Boundary conditions:** $RCS_{min}[dBsm]$ detection thresholds of radar sensor
  e.g. linear increase: $RCS_{min}(r = 0m) = 0.1dBsm; RCS_{min}(r = 250m) = 20dBsm$
- **Input:** x, y-positions and RCS-values of objects
- **Task:** perform coordinate transformation and evaluate if $RCS_{target} > RCS_{min}(r_{target})$
- **Output:** objects detected by radar

Part 4: Sensor fusion using radar and vision data

Underlying equations

- **Coordinate transformation**

$$r_{target} = \sqrt{x_{target}^2 + y_{target}^2}$$
$$\varphi_{target} = \arctan(\frac{y_{target}}{x_{target}})$$

- **Detection threshold of radar at target location:**

$$RCS_{min}(r_{target}) = RCS_{min}(r_0) + \frac{RCS_{min}(r_1) - RCS_{min}(r_0)}{r_1} * r_{target}$$

- **Object detection:**

$RCS_{target} > RCS_{min}(r_{target})$ → object detected

Code implementation

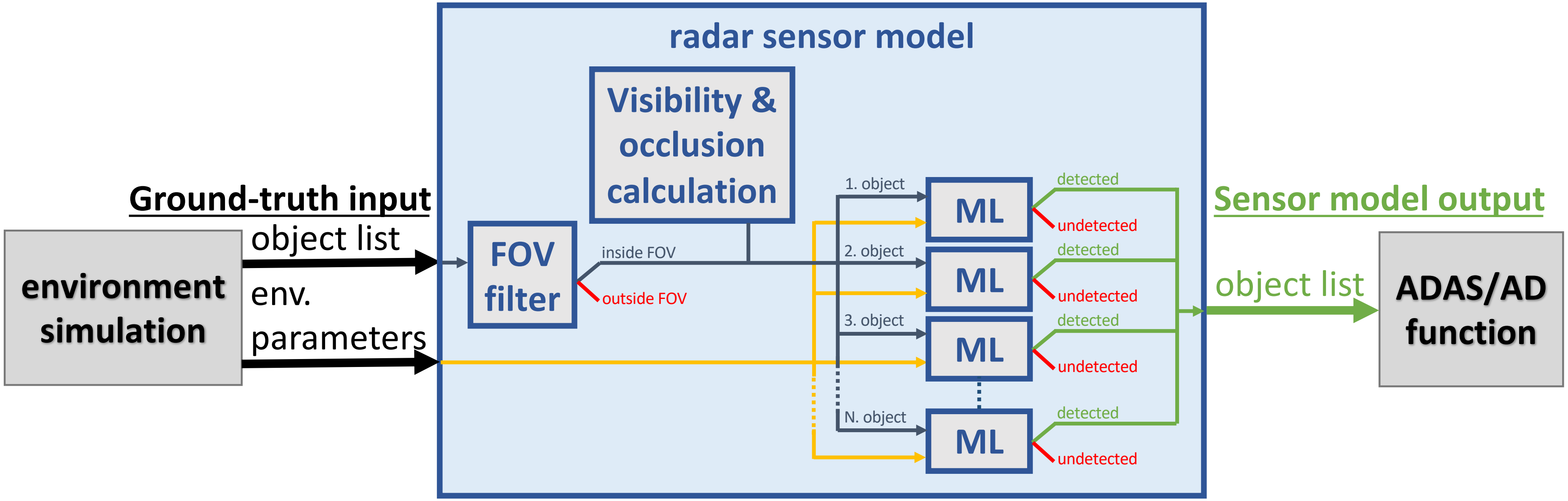
```
object_list = [(50,2,10), (120,-10,5), …]
for (x,y,RCS) in object_list:
   r = sqrt(x**2+y**2); phi = arctan(y/x)
   object_list_transformed.add((r,phi,RCS))

RCS_min_r0 = 0.1
r1 = 250; RCS_min_r1 = 20

for (r,phi,RCS) in object_list_transformed:
   RCS_min = RCS_min_r0 + r*(RCS_min_r1- RCS_min_r0)/r1
      if RCS_min < RCS:
         objects_detected.add((r,phi,RCS))
```

Part 4: Sensor fusion using radar and vision data

ML (machine learning) based detection



- **Pre-trained ML-algorithm** decides based on multiple inputs (x, y-positions, object type, occlusion, visibility, weather conditions, solar irradiance, RCS_{target}, R_{target}, etc.) whether object is detected or not
- **Visibility** can be included as e.g. visible (not occluded) part of object in percentage
- **Occlusion** can be included as e.g. number of objects that are in direct line-of-sight between sensor and target

Part 4: Sensor fusion using radar and vision data

Create a Tracker

- to track the vehicles that are close to the ego vehicle,
- Note:
  1. initialize a constant velocity motion model
  2. initialize the Kalman filter that works with position and velocity

It is responsible for the following:

A. Assigning detections to tracks.

B. Initializing new tracks based on unassigned detections. All tracks are initialized as 'Tentative', accounting for the possibility that they resulted from a false detection.

C. Confirming tracks if they have more than *M* assigned detections in *N* frames.

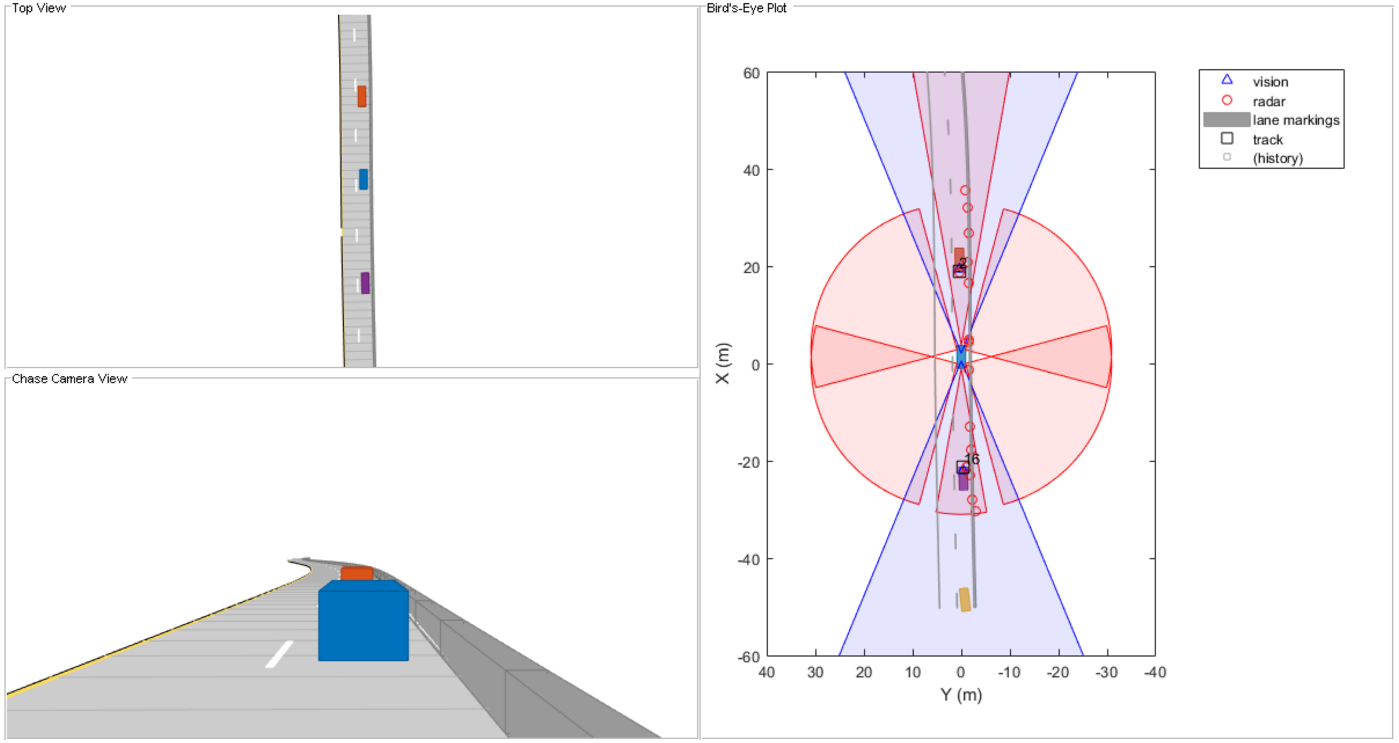
D. Updating existing tracks based on assigned detections.

E. Coasting (predicting) existing unassigned tracks.

F. Deleting tracks if they have remained unassigned (coasted) for too long

Part 4: Sensor fusion using radar and vision data

Simulate the Scenario



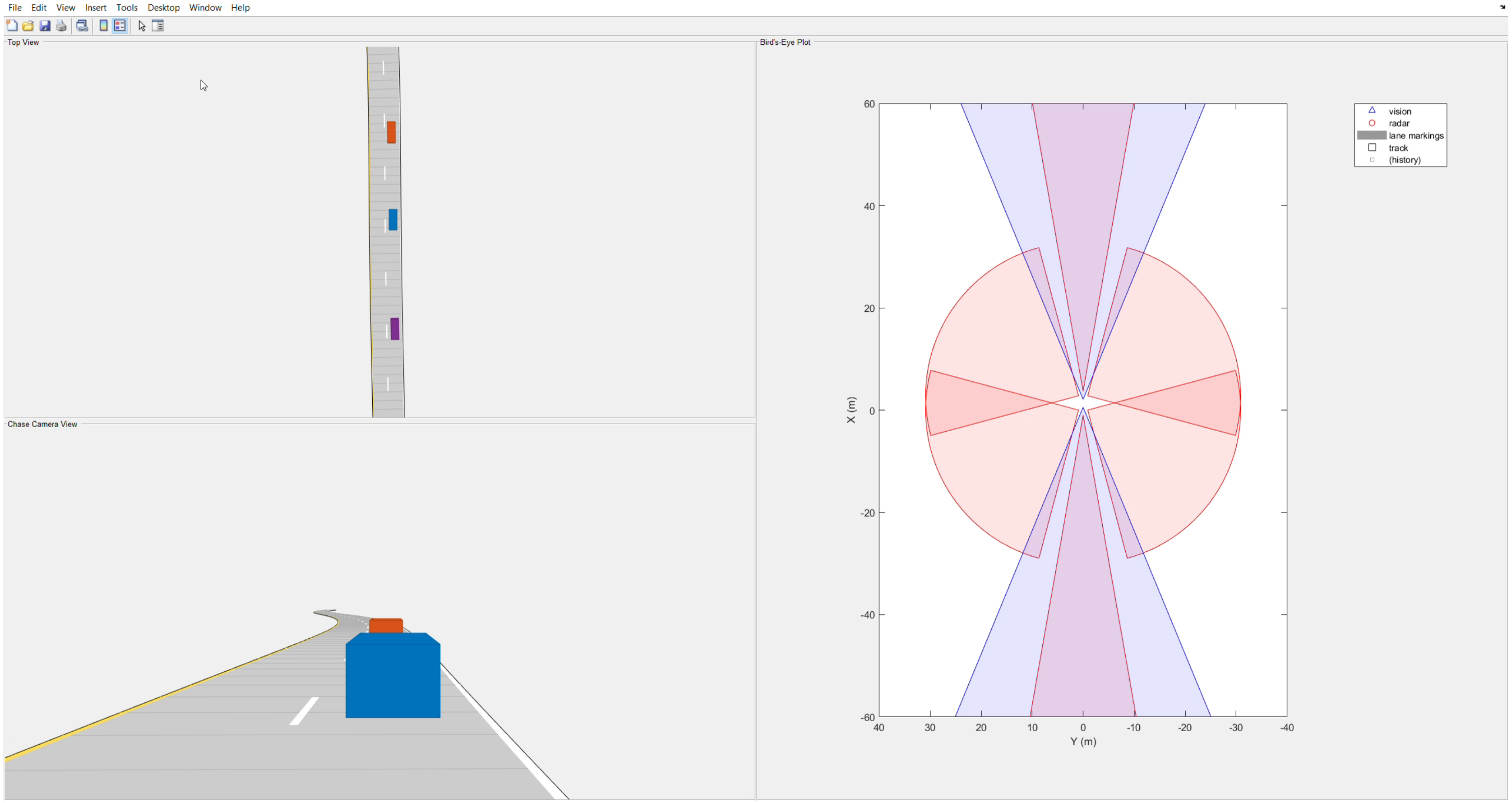
Simulation loop:

- moves the vehicles,
- calls the sensor simulation,
- performs the tracking.

Sampling times:

- scenario generation every 10 ms,
- sensors detect every 100 ms.

Part 4: Sensor fusion using radar and vision data



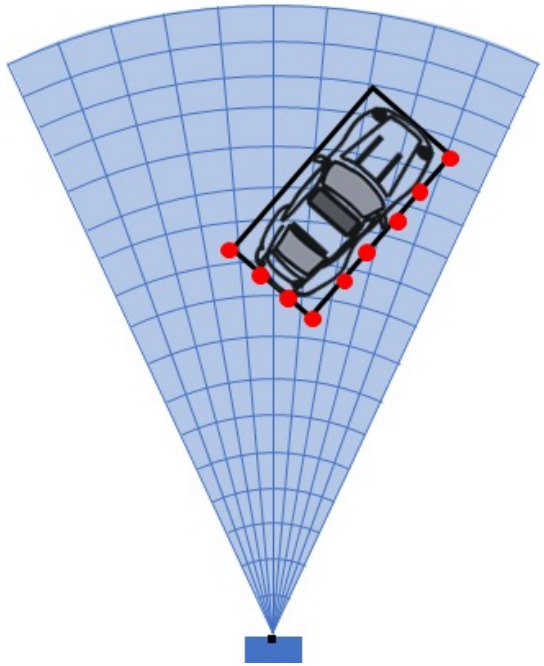
Part 4: Sensor fusion using radar and vision data

- How to track objects that return **multiple detections** in a single sensor scan?
- How to track objects with **high-resolution radar sensors**?

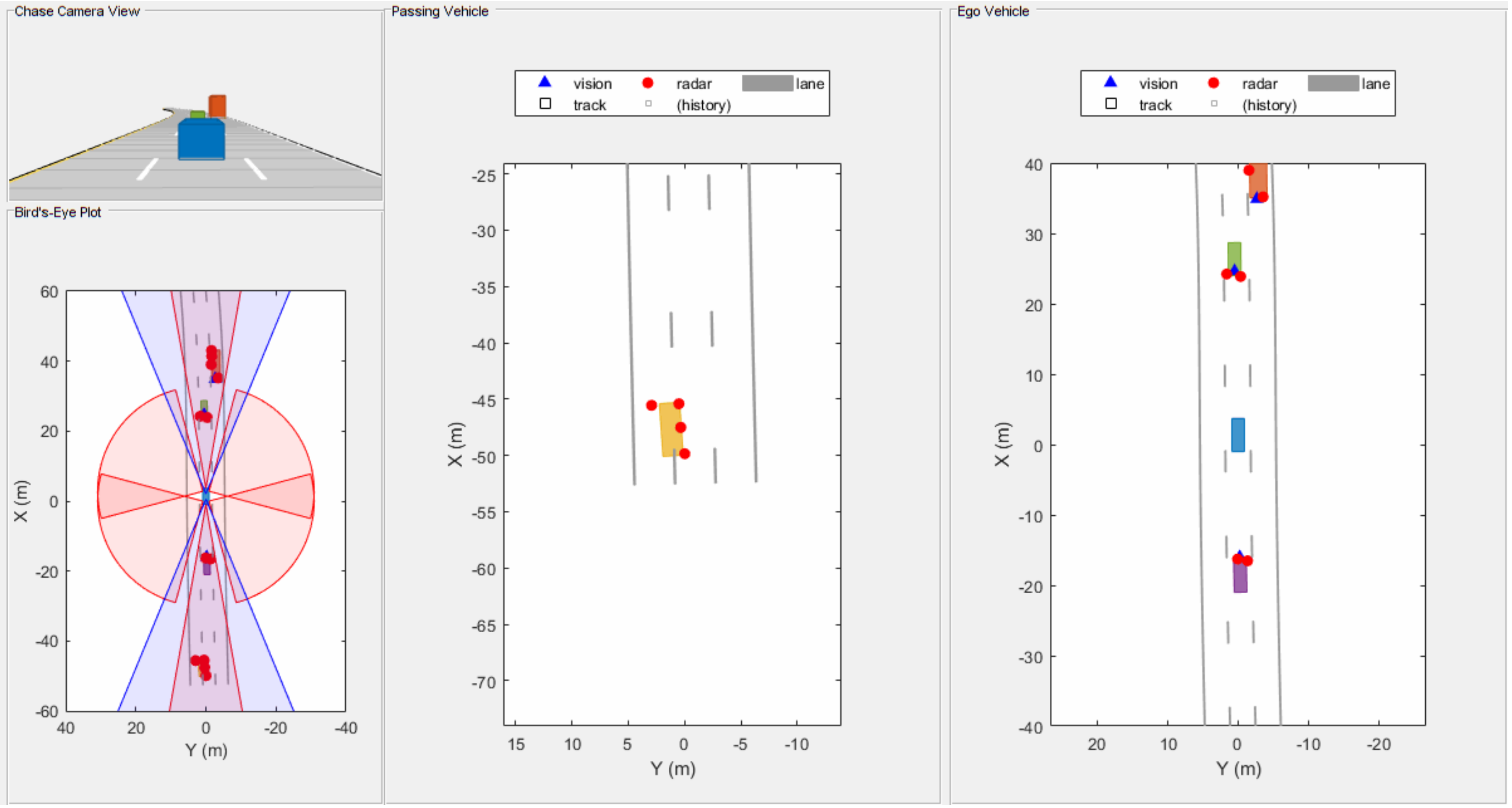
- Extended objects present new challenges to conventional trackers
- Standard trackers assume a single detection per object per sensor.

- **Extended object trackers**
  - Can handle multiple detections per object.
  - Estimate position and velocity, but also the dimensions and orientation of the object.

- **Prominent algorithms:**
  - Gamma Gaussian inverse Wishart probability hypothesis density (phd) tracker
  - Gaussian-mixture phd tracker
  - …



Part 4: Sensor fusion using radar and vision data



- Gaussian mixture phd tracker (here: MATLAB implementation)
- Can handle multiple detections per object per sensor (here: 6 radars, 2 cameras)
- It estimates the size and orientation of the object (along with pose and velocity)

Concluding remarks

- **Multi-object tracking** and **multi-sensor data fusion** - core of the autonomous systems perception

- Tracking is essential for guidance, navigation, and control of autonomous systems.

- **Typical tracking system**
  - estimates targets (number of targets and their states),
  - evaluates the situational environment in an area of interest by taking detections,
  - tracks the targets over time.

Takeaways

- **Single target tracking (STT)**
  - assumes only one target,
  - does not require data assignment or association,
  - the detection is directly fed to an estimator / filter.

- **Multiple target tracking (MTT), multi-object tracking (MOT)**
  - multiple detections from multiple targets,
  - use of one or more sensors,
  - one or more tracks are used to estimate the states of the targets.

- **Extended object tracking**
  - high-resolution radar/lidar sensors,
  - can handle multiple detections per object.
  - estimate position and velocity, but also the dimensions and orientation of the object.

Thank you for your attention!