# CS231A

# **Computer Vision:**
# From 3D Reconstruction to Recognition

Optimal Estimation Cont’

# Recap

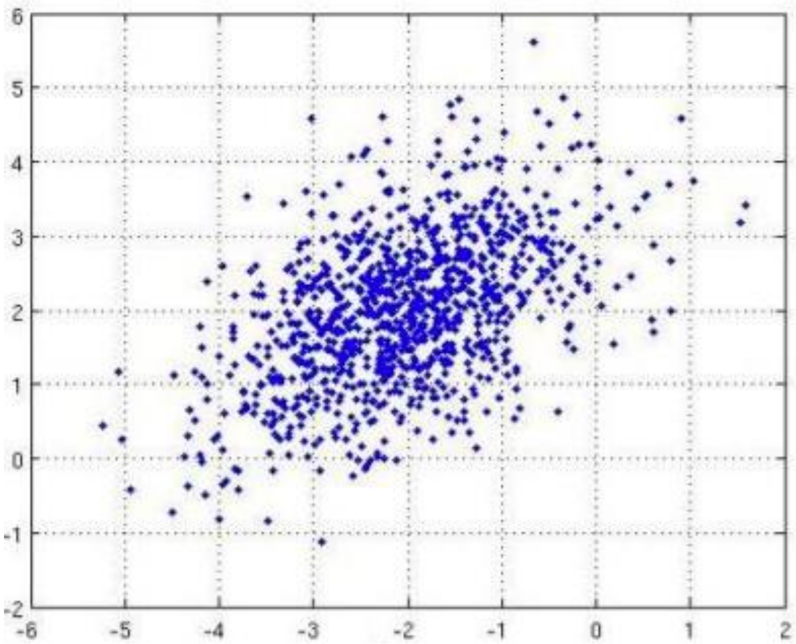
- Recursive Filter
- Kalman Filter
- Extended Kalman Filter

# Nonparametric filters

- No fixed functional form of the posterior – can capture multimodality
- Instead: finite numbers of values

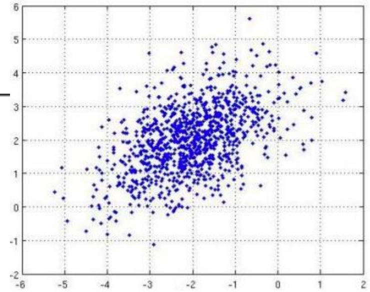
- Histogram filter: State = finitely many regions
- Particle filter: Distribution represented by samples

# Particle Filter




$$p(x_t|z_{t:1}, u_{t:1}, \boldsymbol{x}_0) \rightarrow X_t = \{x_t^0, \ldots, x_t^N\}$$

# The Particle filter algorithm



**Algorithm Particle_filter**$(\mathcal{X}_{t-1}, u_t, z_t)$**:**

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ to $M$ do

  sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$ **Process Model**

  $w_t^{[m]} = p(z_t \mid x_t^{[m]})$ **Measurement Model**

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ to $M$ do

  draw $i$ with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

**Importance Sampling**

Before resampling → $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx$

After resampling → $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$

# Particle Filter - Process Model

$$p(x_{t-1}|z_{t-1:1}, u_{t-1:1}, \boldsymbol{x}_0) \rightarrow X_{t-1} = \{x_{t-1}^0, \ldots, x_{t-1}^N\}$$

$$x_{t-1}^n \rightarrow p(x_t|x_{t-1}^n, u_t, \boldsymbol{x}_0) \rightarrow \hat{x}_t^n$$

$$\hat{X}_t = \{\hat{x}_t^0, \ldots, \hat{x}_t^N\}$$

# Particle Filter – Measurement Model

$$w_t^{[i]} = \frac{target}{proposal} \propto p(z_t \mid x_t, m)$$

- Draw sample $i$ with probability $w_t^{[i]}$. Repeat M times.
- Informally: "Replace unlikely samples by more likely ones"
- Survival of the fittest
- "Trick" to avoid that many samples cover unlikely states
- Needed as we have a limited number of samples

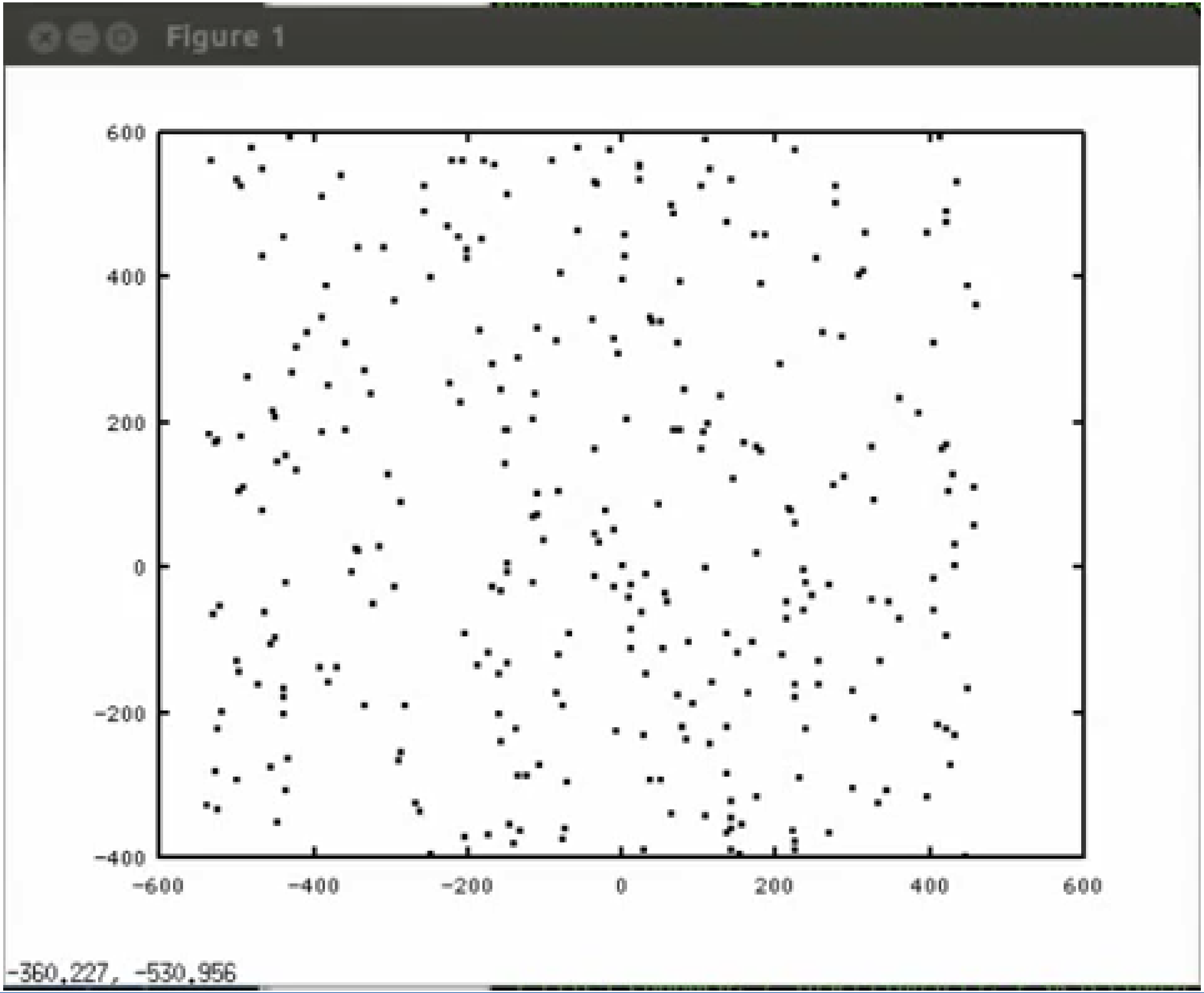


Figure 1
600
400
200
0
-200
-400
-600
-400
-200
0
200
400
600
-360.227, -530.956

# When to Use Each?

Bayes Filter

General Framework
No implementation!

Kalman Filter

Linear Models
Gaussian Distributions

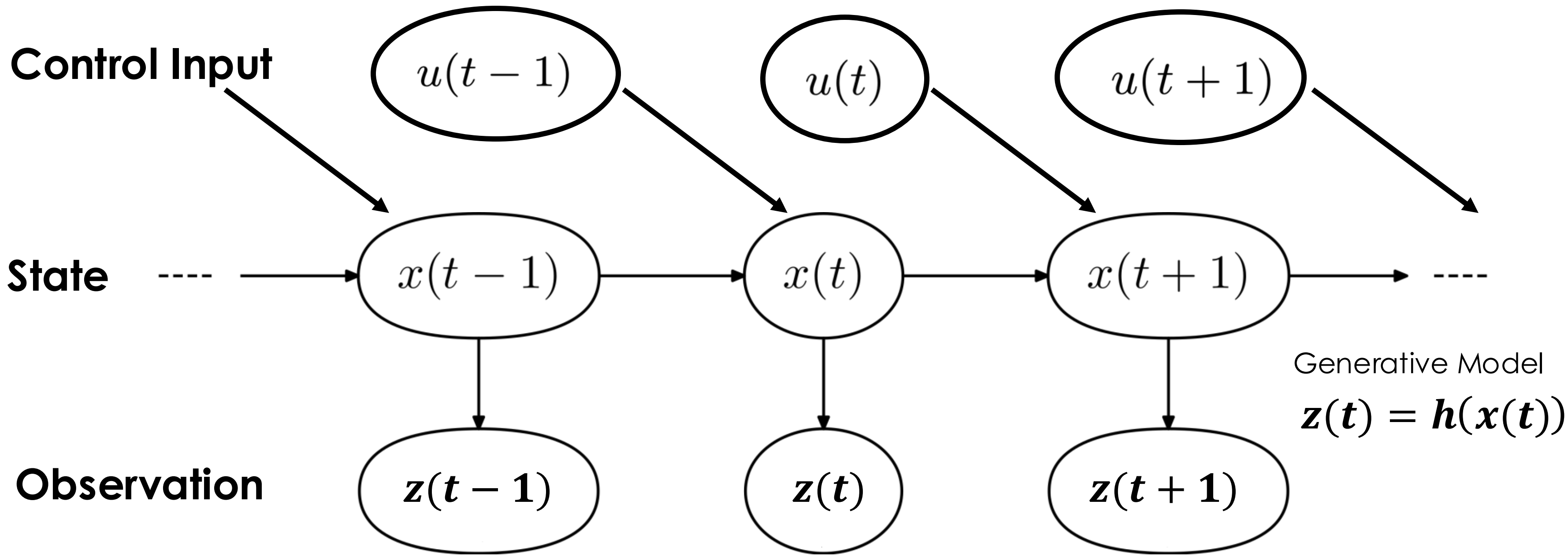
Extended Kalman Filter

Non-Linear Models (linearizable)
Gaussian Distributions

Particle Filter

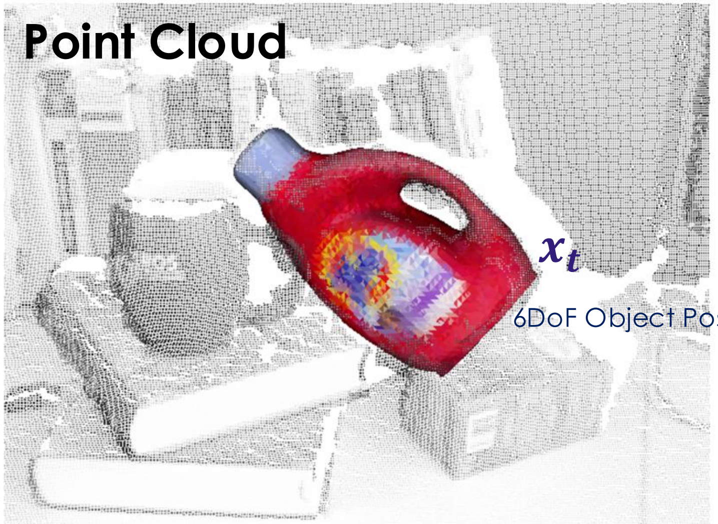
Any Model
Any Distribution
Low Dimensional State Space

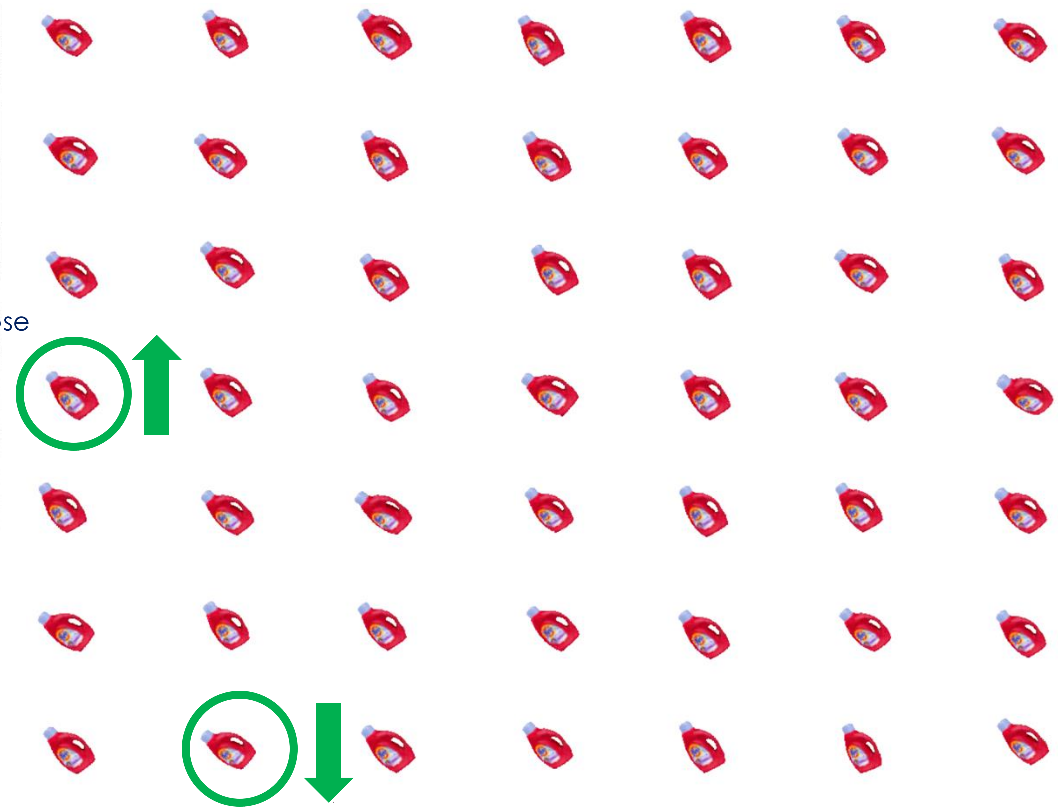
# Graphical Model of System to Estimate




**Algorithm Bayes_filter**($bel(x_{t-1}), u_t, z_t$)**:**
    for all $x_t$ do
        $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx$
        $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$
    endfor
    return $bel(x_t)$

# Example Observation model for 3D object



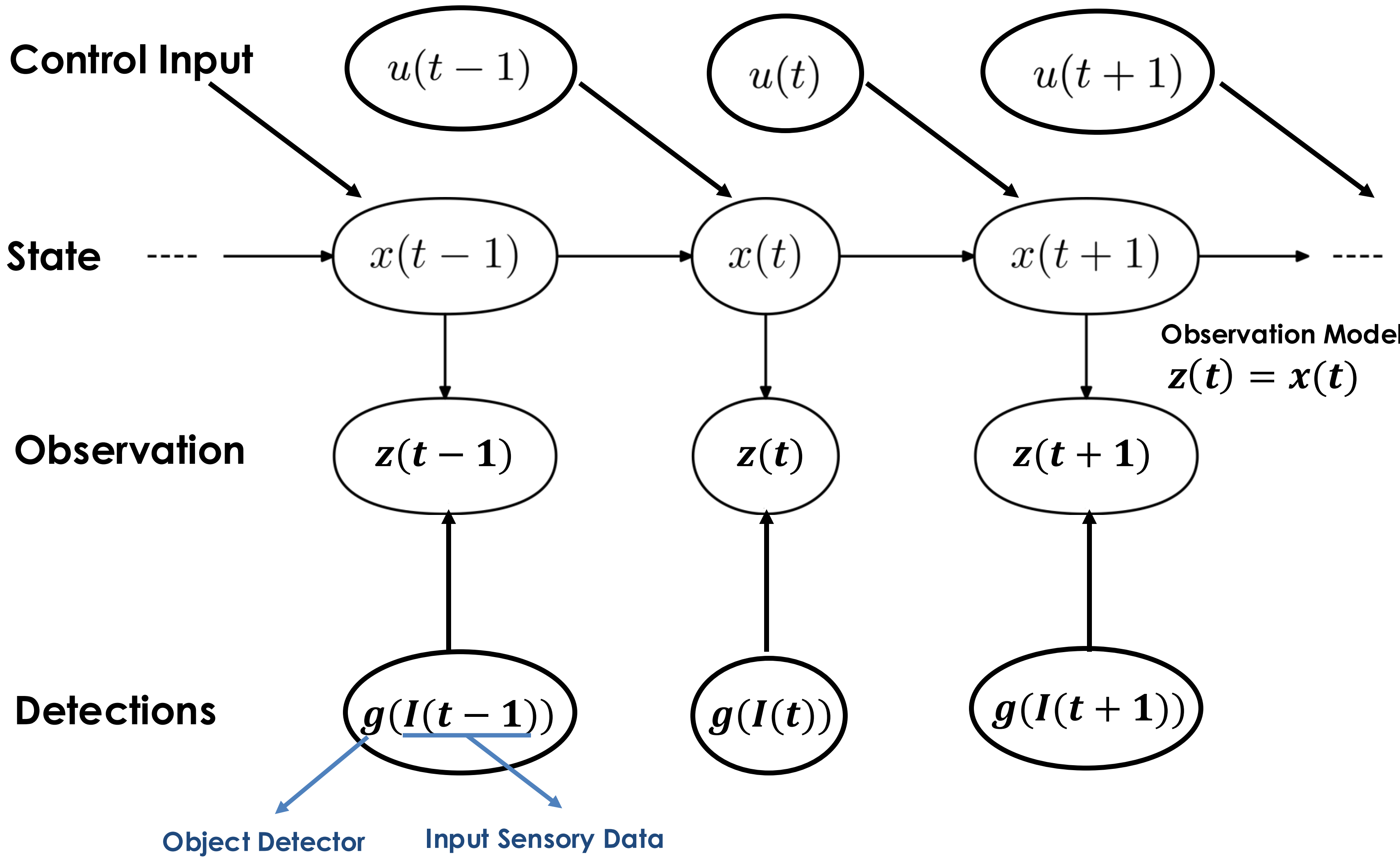
**Rendered Particles**

**Algorithm Particle_filter**$(\mathcal{X}_{t-1}, u_t, z_t)$:
$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
for $m = 1$ to $M$ do
sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
$w_t^{[m]} = p(z_t \mid x_t^{[m]})$
$\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
endfor
for $m = 1$ to $M$ do
draw $i$ with probability $\propto w_t^{[i]}$
add $x_t^{[i]}$ to $\mathcal{X}_t$
endfor
return $\mathcal{X}_t$

Importance Sampling

Changhyun Choi and Henrik I. Christensen. Rgb-d object tracking: A particle filter approach on gpu. In IROS, pages 1084–1091, 2013

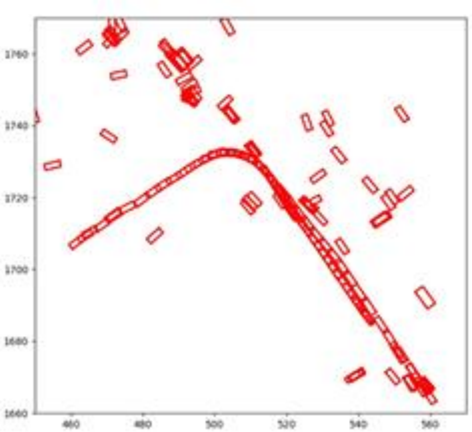
# Tracking by Detection



# Problem Statement: Input

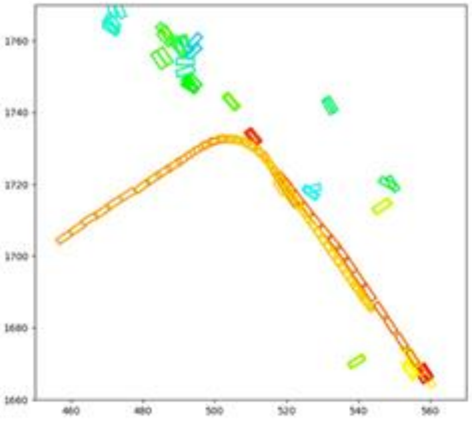
**Probabilistic 3d multi-object tracking for autonomous driving**. H Chiu, A Prioletti, J Li, J Bohg
arXiv preprint arXiv:2001.05673

- Object detections at each frame in a sequence
- Each detection bounding box is represented by:
  - center position (x, y, z), rotation angle along the z-axis (a), and the scale (l, w, h)
  - category label (car, pedestrian, ...), confidence score (c)



# Problem Statement: Output

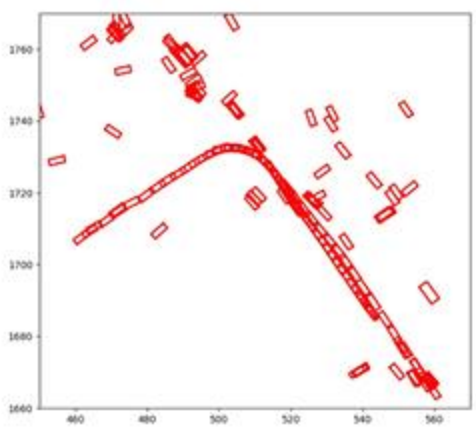
- Tracking object bounding boxes at each frame in a sequence
- Each tracking bounding box is represented by:
  - center position (x, y, z), rotation angle along the z-axis (a), and the scale (l, w, h)
  - category label (car, pedestrian, ...), confidence score (c)
  - tracking id: one unique tracking id for each object instance across frames



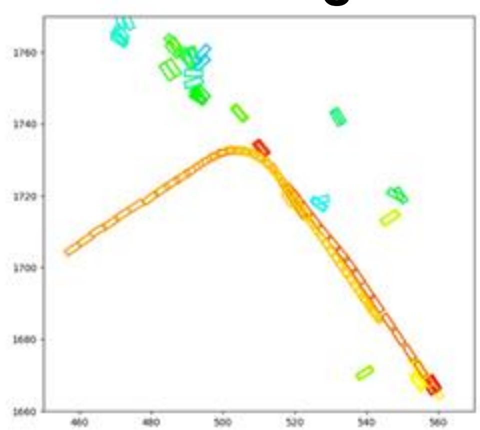
# Why Tracking?

- Filter out the out-liners from the detection results
- Continue estimating object states even if occluded
- Forecast the future based on past trajectories and motion patterns
- Make autonomous driving decisions

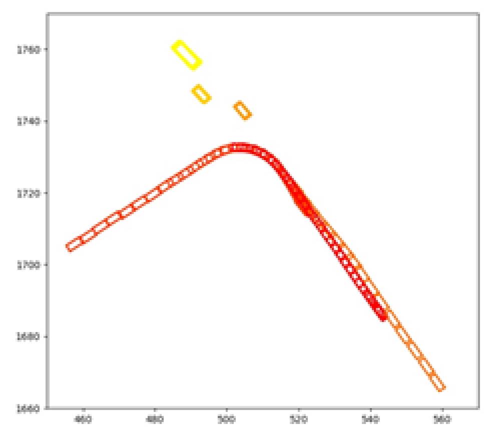
**Detection**

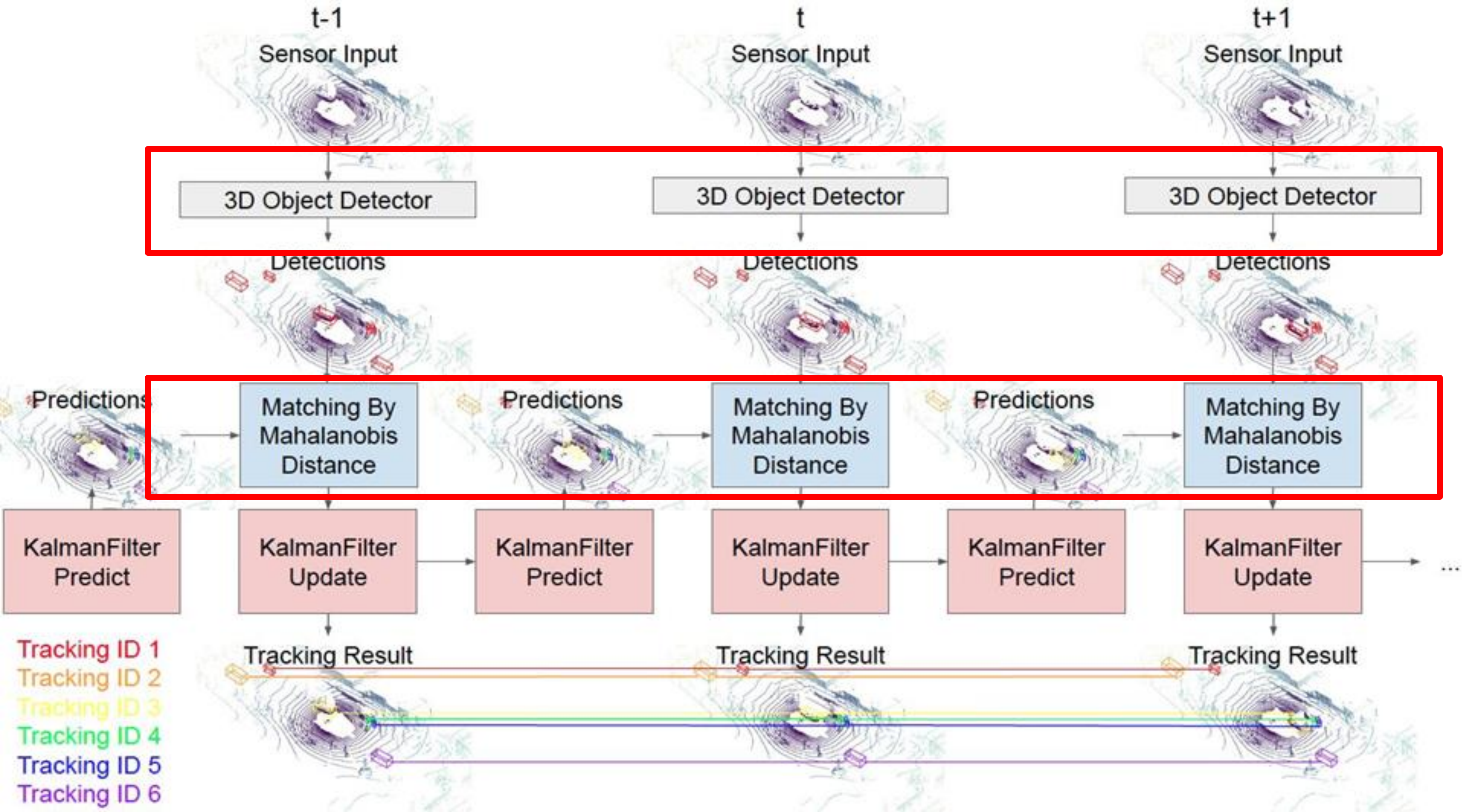


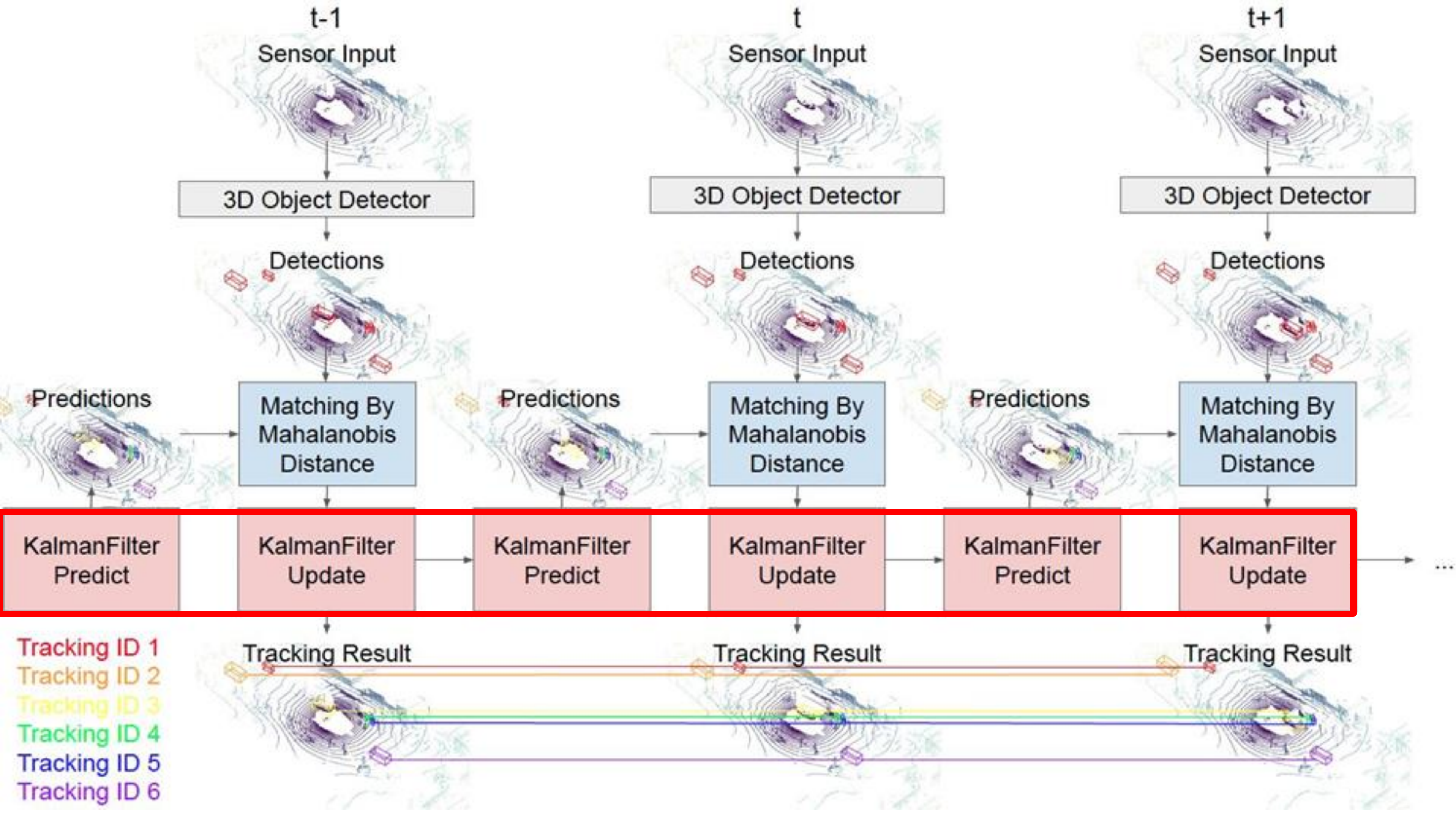
**Tracking**



**Ground-truth**




t-1
t
t+1
Sensor Input
Sensor Input
Sensor Input
3D Object Detector
3D Object Detector
3D Object Detector
Detections
Detections
Detections
Predictions
Matching By Mahalanobis Distance
Predictions
Matching By Mahalanobis Distance
Predictions
Matching By Mahalanobis Distance
KalmanFilter Predict
KalmanFilter Update
KalmanFilter Predict
KalmanFilter Update
KalmanFilter Predict
KalmanFilter Update
...
Tracking ID 1
Tracking ID 2
Tracking ID 3
Tracking ID 4
Tracking ID 5
Tracking ID 6
Tracking Result
Tracking Result
Tracking Result


t-1
Sensor Input
3D Object Detector
Detections
Predictions
Matching By Mahalanobis Distance
KalmanFilter Predict
KalmanFilter Update
Tracking Result
t
Sensor Input
3D Object Detector
Detections
Predictions
Matching By Mahalanobis Distance
KalmanFilter Predict
KalmanFilter Update
Tracking Result
t+1
Sensor Input
3D Object Detector
Detections
Predictions
Matching By Mahalanobis Distance
KalmanFilter Predict
KalmanFilter Update
Tracking Result
...
Tracking ID 1
Tracking ID 2
Tracking ID 3
Tracking ID 4
Tracking ID 5
Tracking ID 6

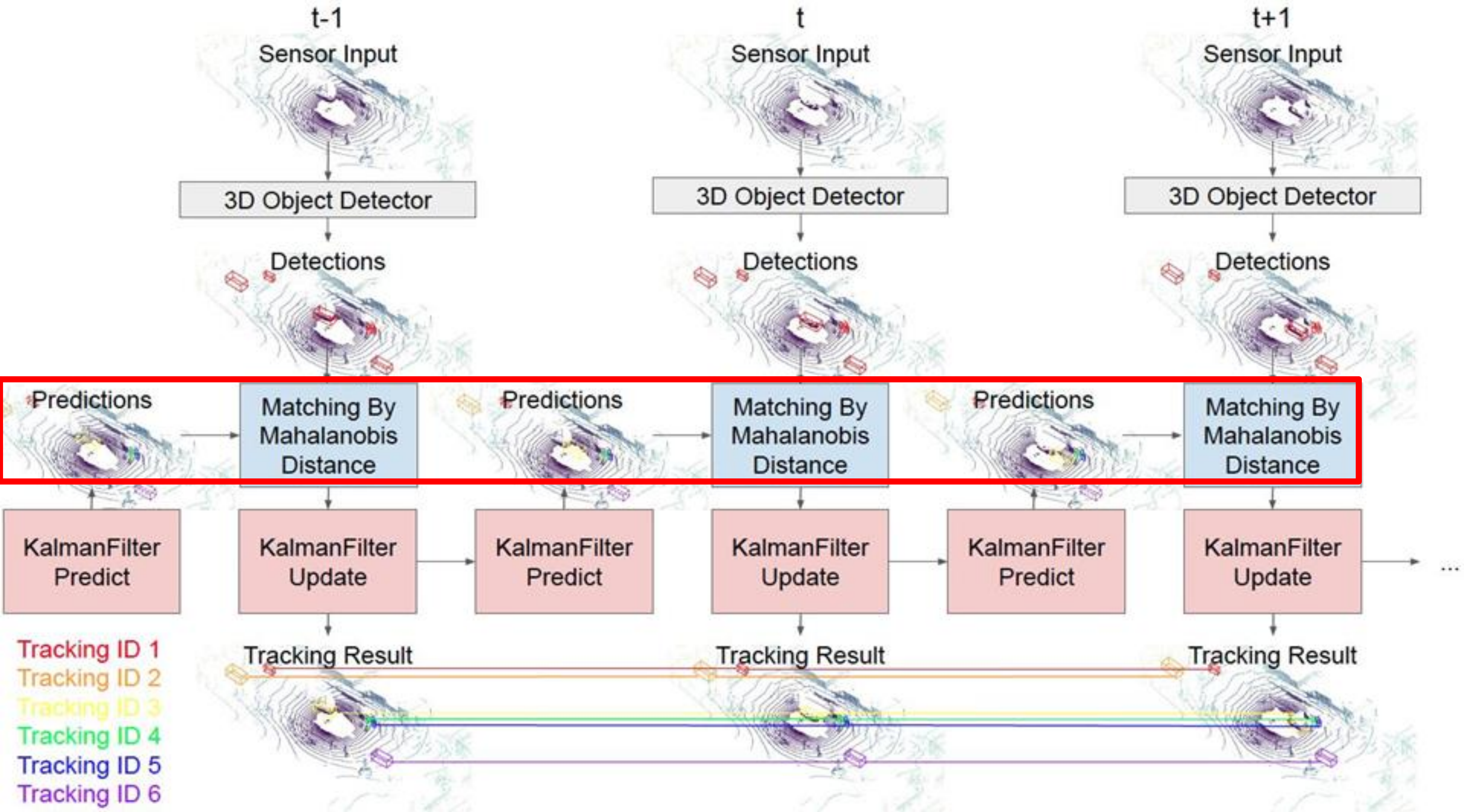
# Kalman Filter for Tracking

Define the object **state** using a vector of random variables including the position, the rotation, the scale, linear velocity, and the angular velocity.

$$\mathbf{s}_t = (x, y, z, a, l, w, h, d_x, d_y, d_z, d_a)^T$$

Define the **Process Model** for prediction based on the constant velocity motion:

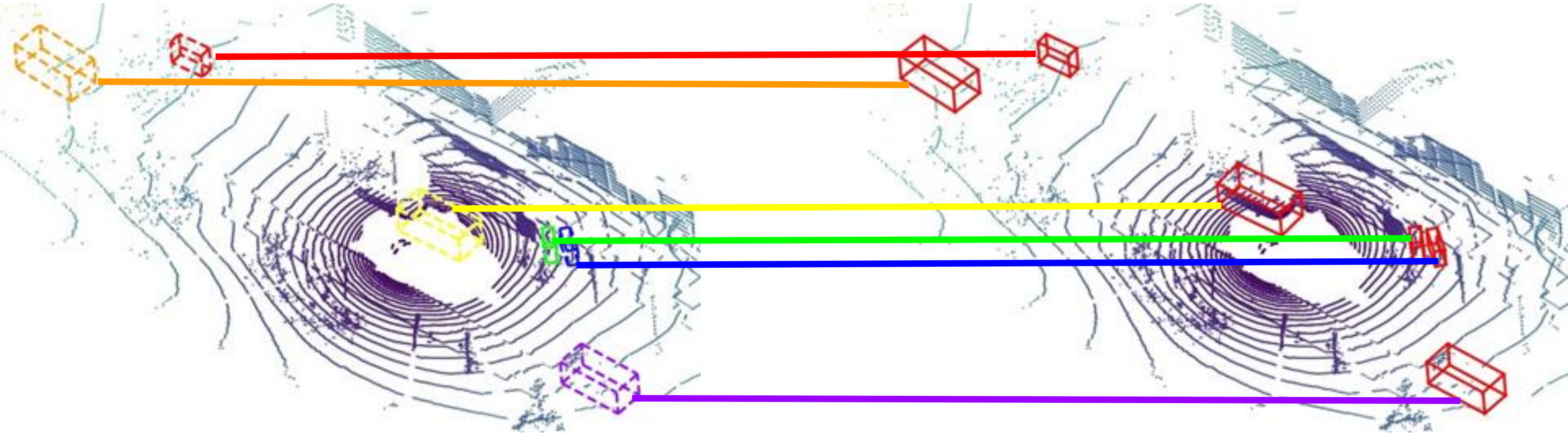
$$\begin{aligned}
\hat{x}_{t+1} &= x_t + d_{x_t} + q_{x_t}, & \hat{d}_{x_{t+1}} &= d_{x_t} + q_{d_{x_t}} & \hat{l}_{t+1} &= l_t \\
\hat{y}_{t+1} &= y_t + d_{y_t} + q_{y_t}, & \hat{d}_{y_{t+1}} &= d_{y_t} + q_{d_{y_t}} & \hat{w}_{t+1} &= w_t \\
\hat{z}_{t+1} &= z_t + d_{z_t} + q_{z_t}, & \hat{d}_{z_{t+1}} &= d_{z_t} + q_{d_{z_t}} & \hat{h}_{t+1} &= h_t \\
\hat{a}_{t+1} &= a_t + d_{a_t} + q_{a_t}, & \hat{d}_{a_{t+1}} &= d_{a_t} + q_{d_{a_t}}
\end{aligned}$$


t-1
Sensor Input
3D Object Detector
Detections
t
Sensor Input
3D Object Detector
Detections
t+1
Sensor Input
3D Object Detector
Detections
Predictions
Matching By Mahalanobis Distance
Predictions
Matching By Mahalanobis Distance
Predictions
Matching By Mahalanobis Distance
KalmanFilter Predict
KalmanFilter Update
KalmanFilter Predict
KalmanFilter Update
KalmanFilter Predict
KalmanFilter Update
...
Tracking ID 1
Tracking ID 2
Tracking ID 3
Tracking ID 4
Tracking ID 5
Tracking ID 6
Tracking Result
Tracking Result
Tracking Result

# Data Association

Mahalanobis Distance $m = \sqrt{\boxed{(z_t - C\bar{\mu}_t)^T S_t^{-1} (z_t - C\bar{\mu}_t)}}$

$S$ = Innovation Covariance
$z_t - C\mu_t$ = innovation



**Kalman Filter Predictions**

**Object Detections**

# Kalman Filter

**Algorithm Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**

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

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

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

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

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

return $\mu_t, \Sigma_t$

# Data Association

Mahalanobis Distance $m = \sqrt{(z_t - C\bar{\mu}_t)^T S_t^{-1} (z_t - } C\bar{\mu}_t)$

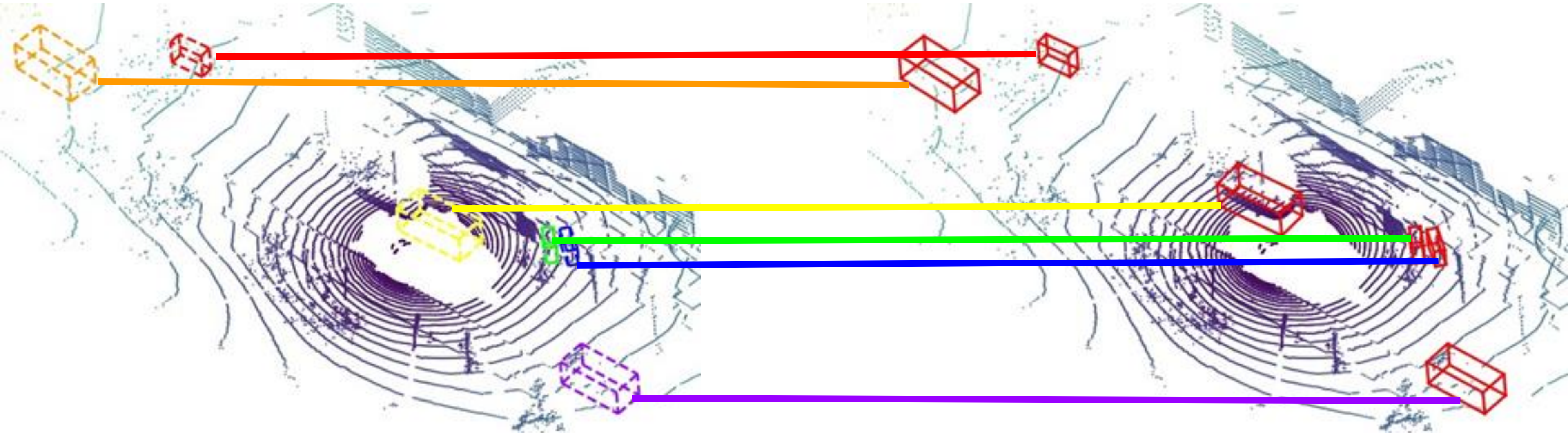
If $\boldsymbol{m} > \mathbf{3} * \boldsymbol{\sigma}$ then reject as outlier. 99.7% of values lie within 3*standard deviatio

Measuring the distance between the observation and the distribution of the predicted state.

Providing distance measurement when there is no overlap between the prediction and detection.

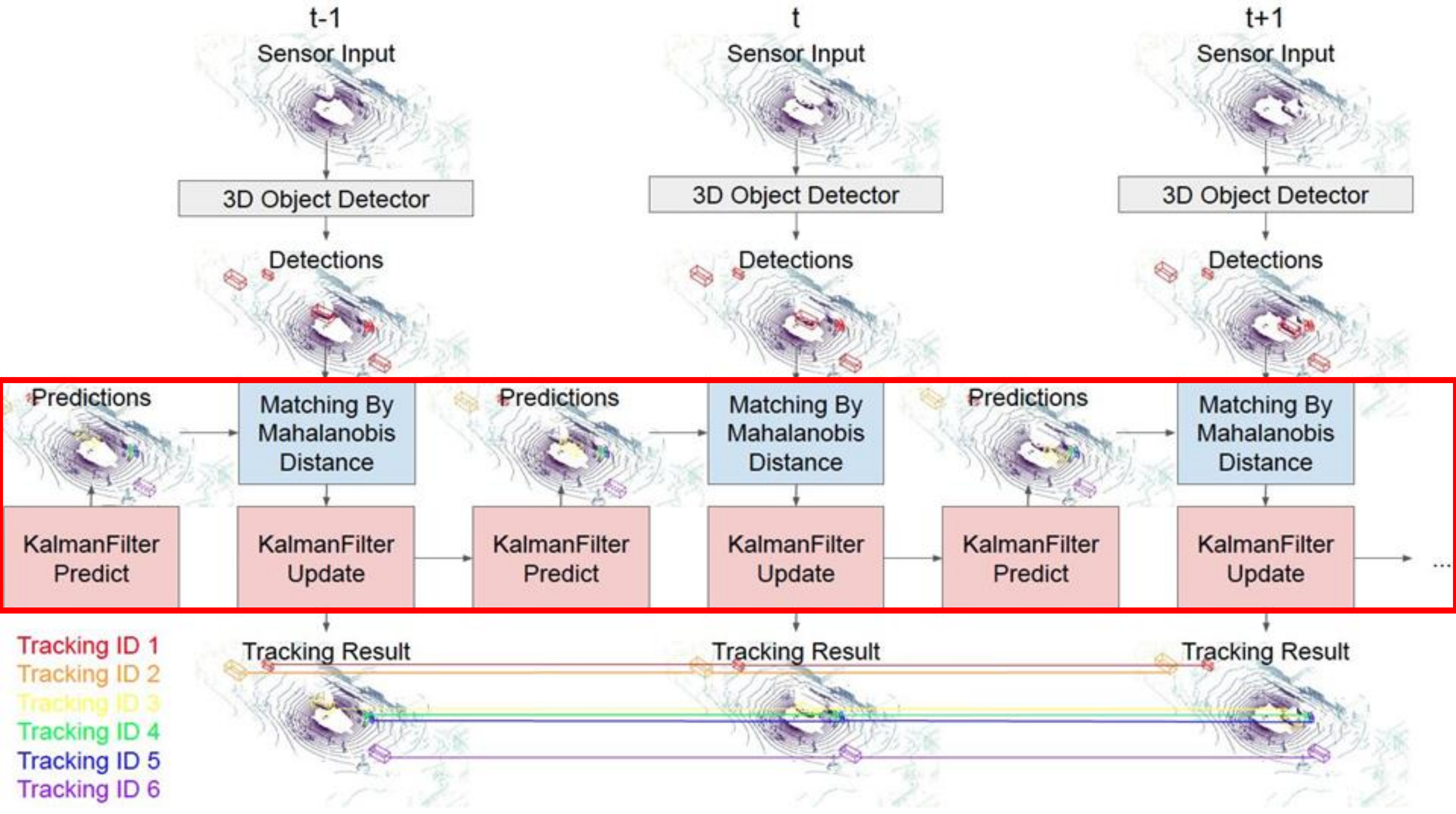
Taking the uncertainty information from the prediction into account.

# Data Association - Greedy



**Kalman Filter Predictions**

**Detections**


t-1
Sensor Input
3D Object Detector
Detections
t
Sensor Input
3D Object Detector
Detections
t+1
Sensor Input
3D Object Detector
Detections
Predictions
Matching By Mahalanobis Distance
KalmanFilter Predict
KalmanFilter Update
Predictions
Matching By Mahalanobis Distance
KalmanFilter Predict
KalmanFilter Update
Predictions
Matching By Mahalanobis Distance
KalmanFilter Predict
KalmanFilter Update
...
Tracking ID 1
Tracking ID 2
Tracking ID 3
Tracking ID 4
Tracking ID 5
Tracking ID 6
Tracking Result
Tracking Result
Tracking Result

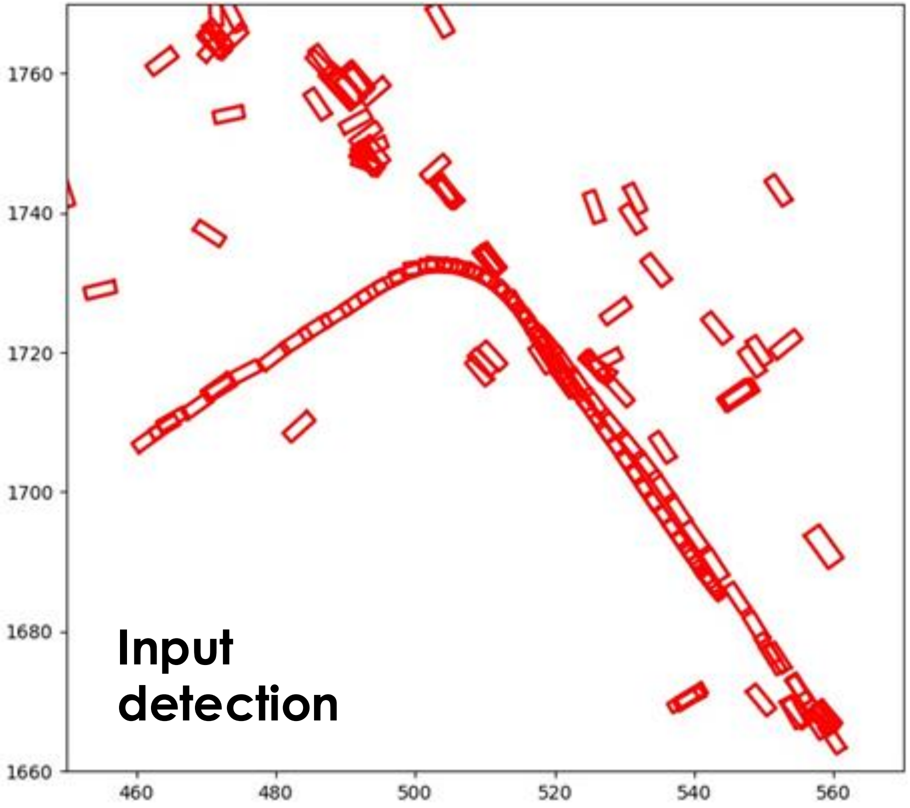
# Kalman Filter

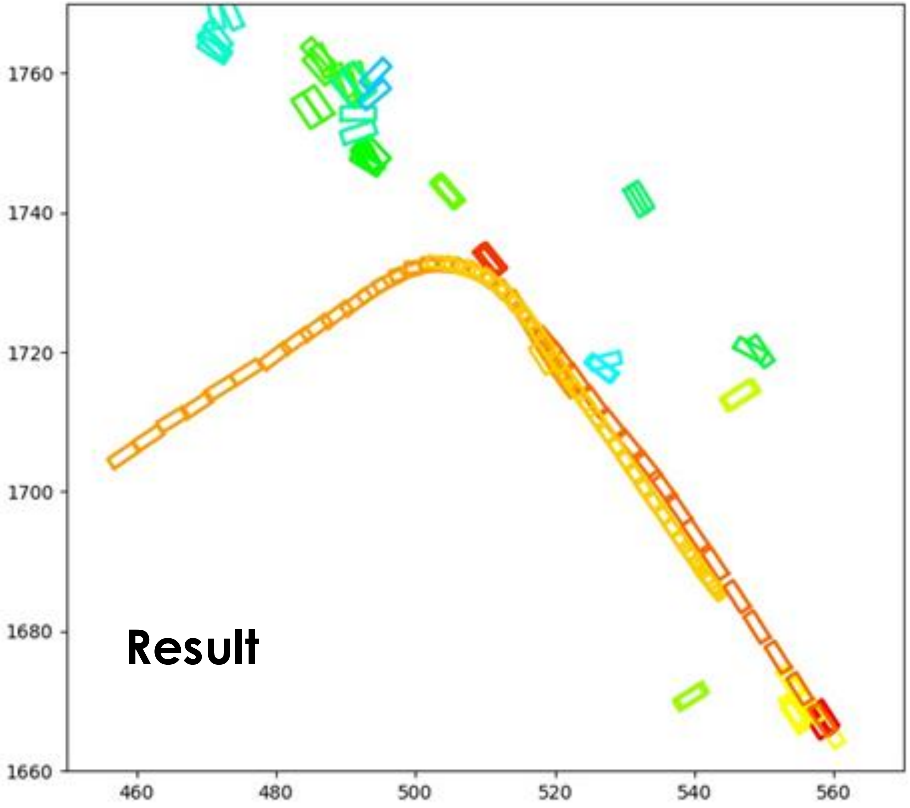
**Algorithm Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**

$$\bar{\mu}_t = A_t\, \mu_{t-1} + B_t\, u_t$$
$$\bar{\Sigma}_t = A_t\, \Sigma_{t-1}\, A_t^T + R_t$$
$$K_t = \bar{\Sigma}_t\, C_t^T (C_t\, \bar{\Sigma}_t\, C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t\, \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t\, C_t)\, \bar{\Sigma}_t$$

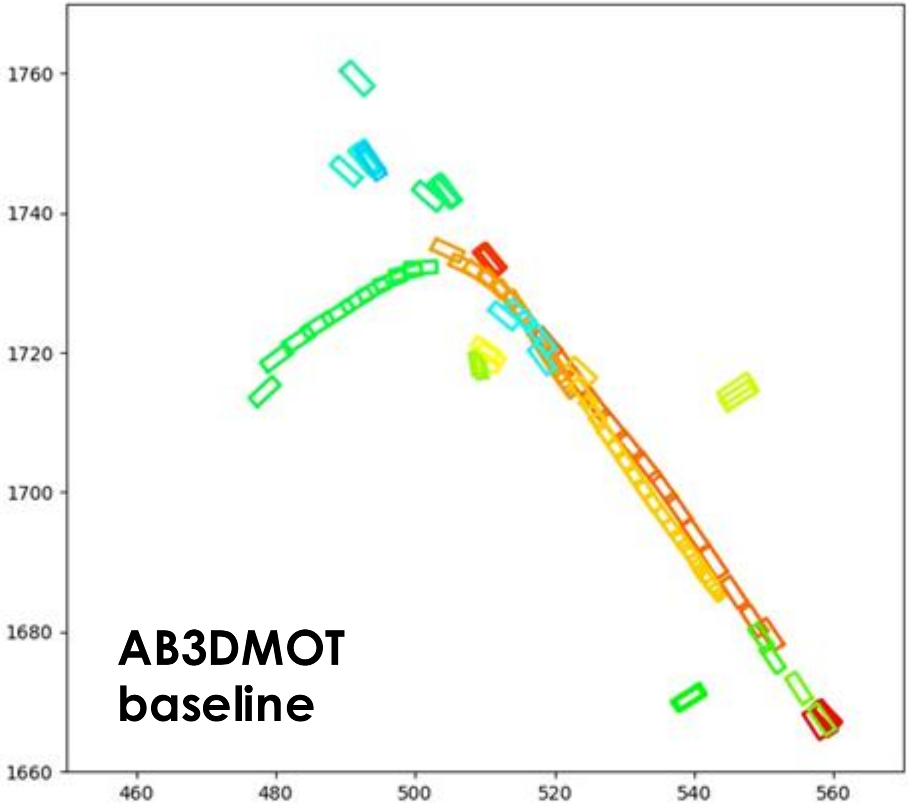
return $\mu_t, \Sigma_t$

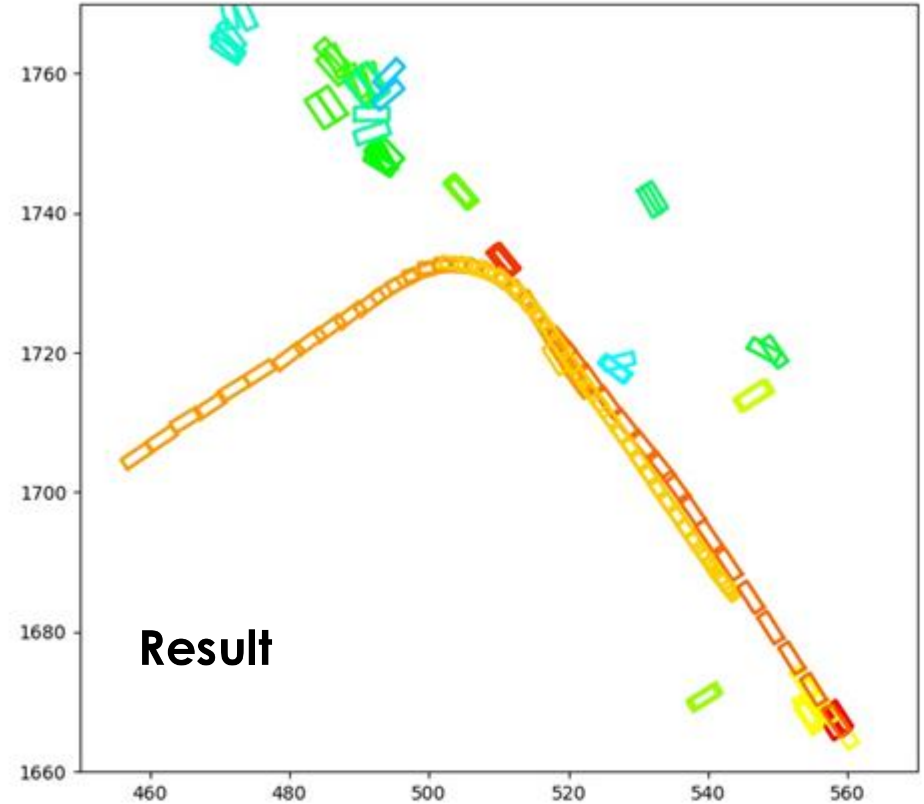
# Qualitative Results



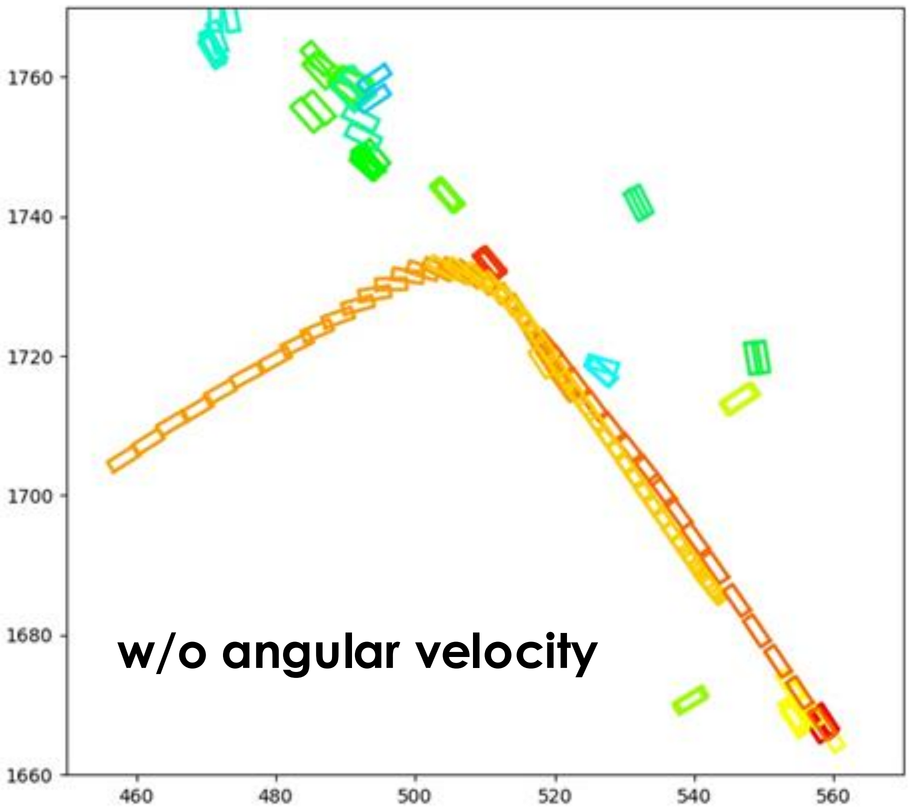



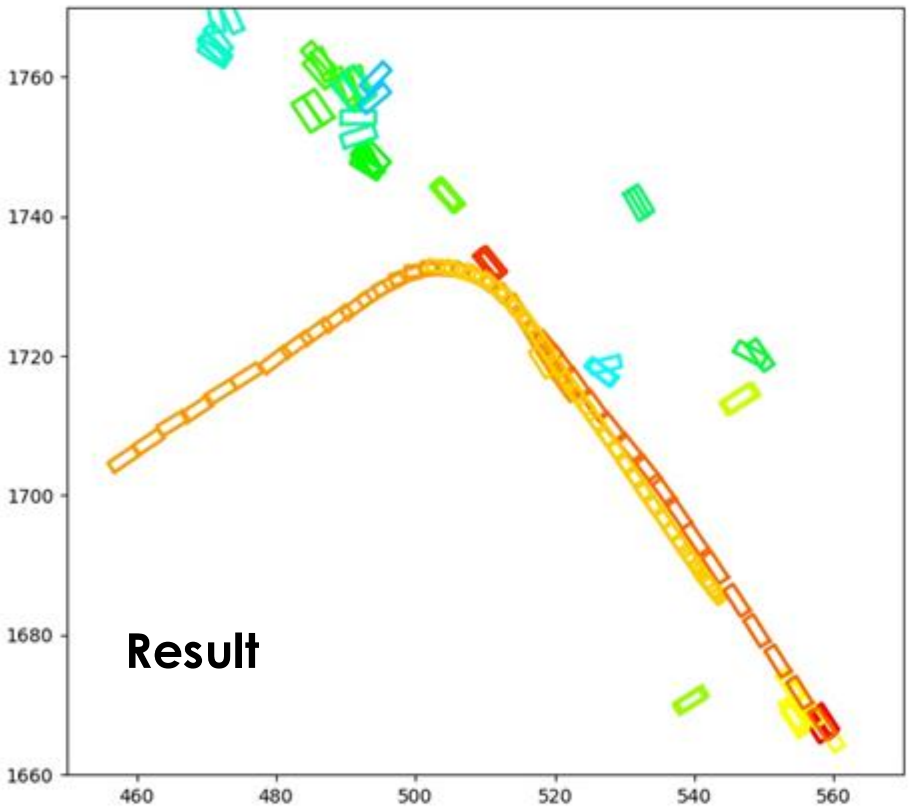
# Qualitative Results



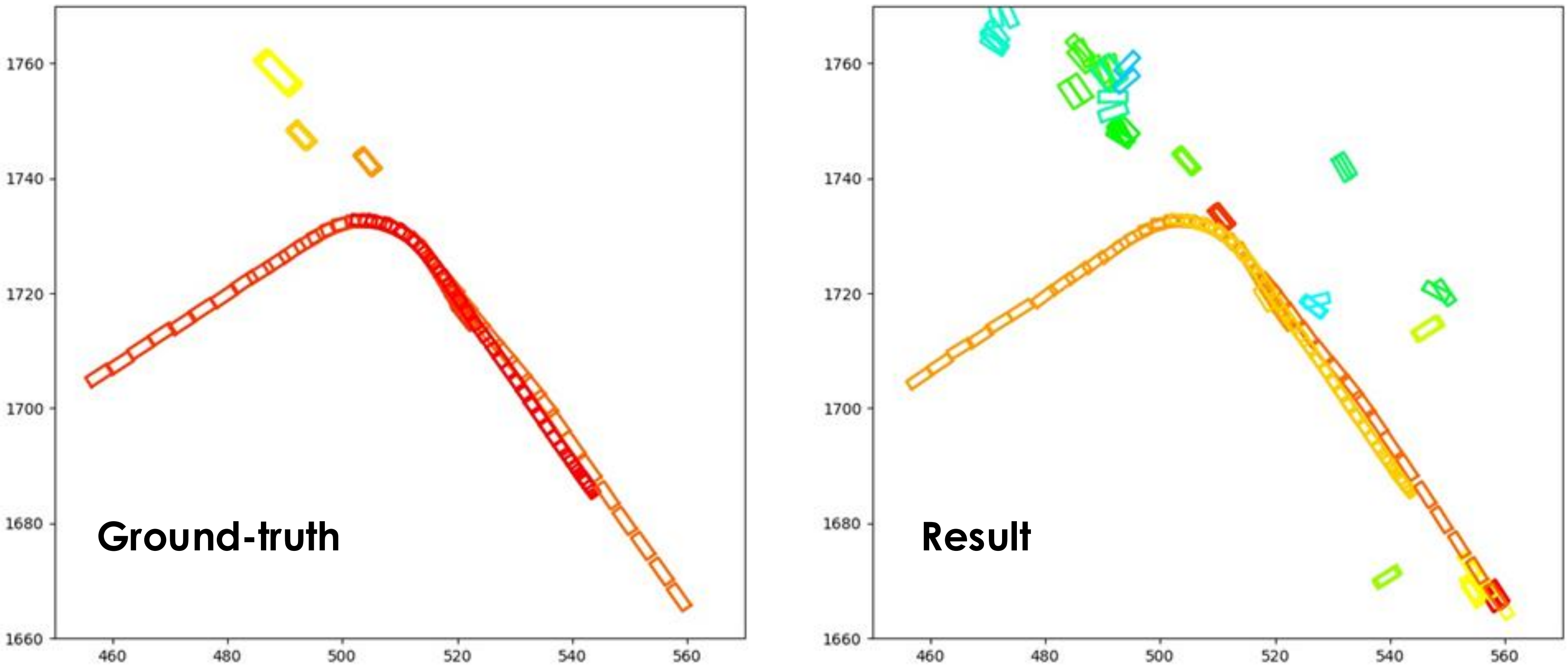



# Qualitative Results



# Qualitative Results



# Priors and Hyperparameters

A lot of hardcoded knowledge!

- State Representation
- Models
    - Forward Model
        - State to next state
        - Action to next state
    - Measurement Model
- Probabilistic Properties
    - Process Noise
    - Measurement Noise

# Differentiable filters

Can we learn models and hyperparameters from data?

Approach: Embed algorithmic structure of Bayesian Filtering into a recurrent neural network.

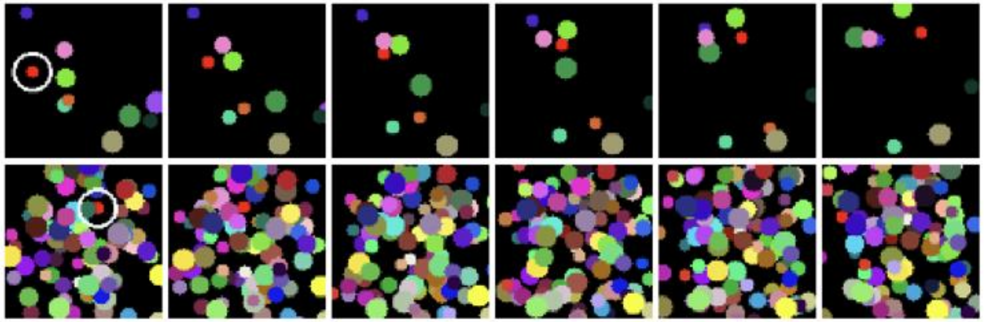
- prevents overfitting through regularization
- Avoids manual tuning and modeling

# BackpropKF : Learning Discriminative Deterministic State Estimators. Haarnoja et al. NeurIPS 2016

- Differentiable version of the Kalman Filter
- Uses Images as observations; learns a sensors that outputs state directly

$$g(I_t) = z_t \approx x_t$$

Track red disk position

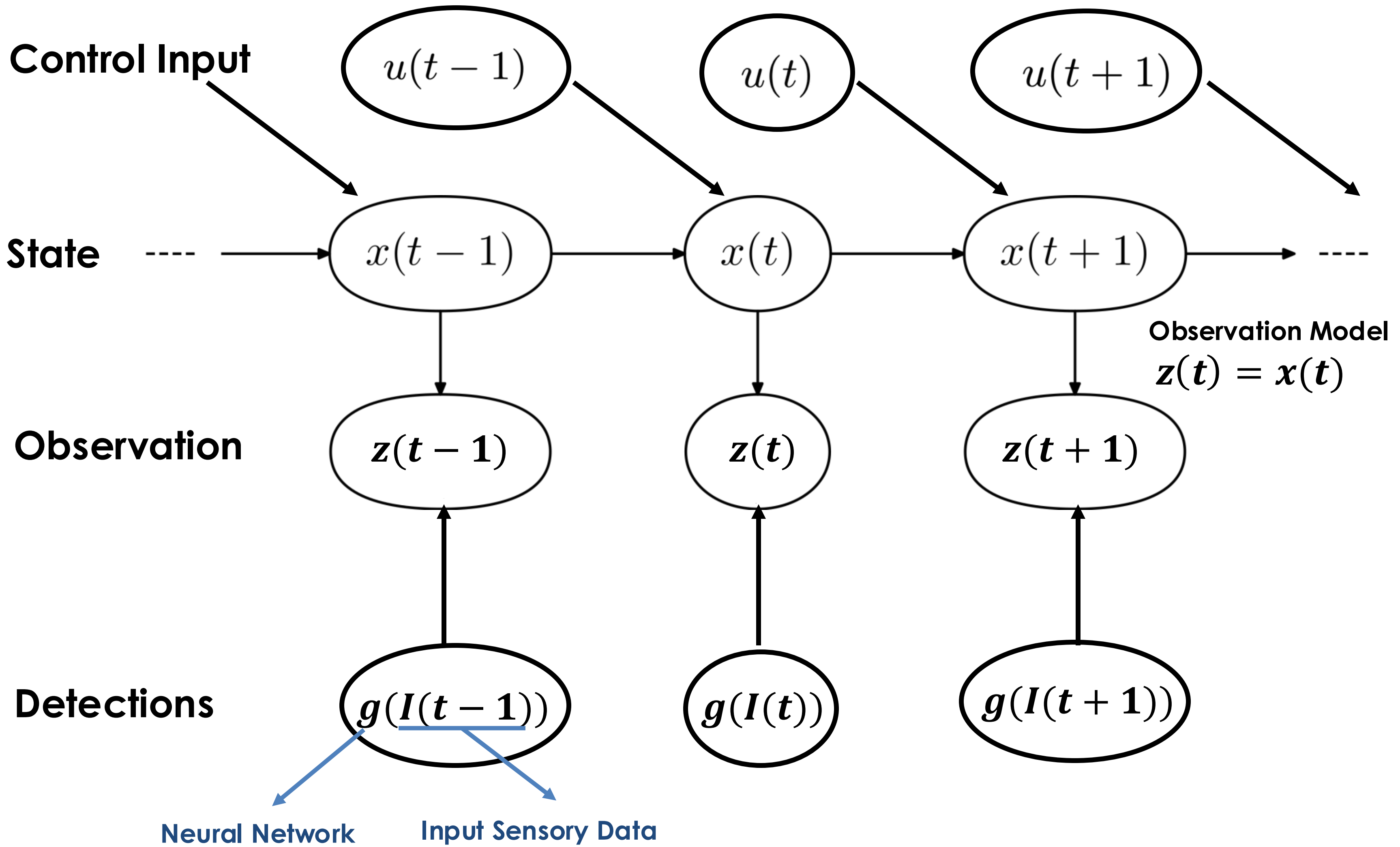


Example Sequence w/ few occlusions

Example Sequence w/ many occlusions

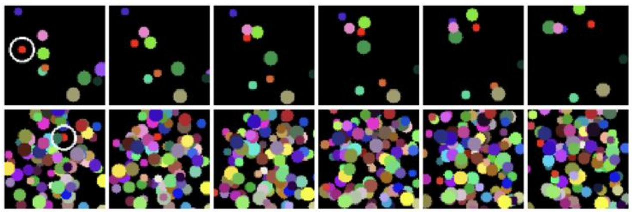
$t$ $t+1$

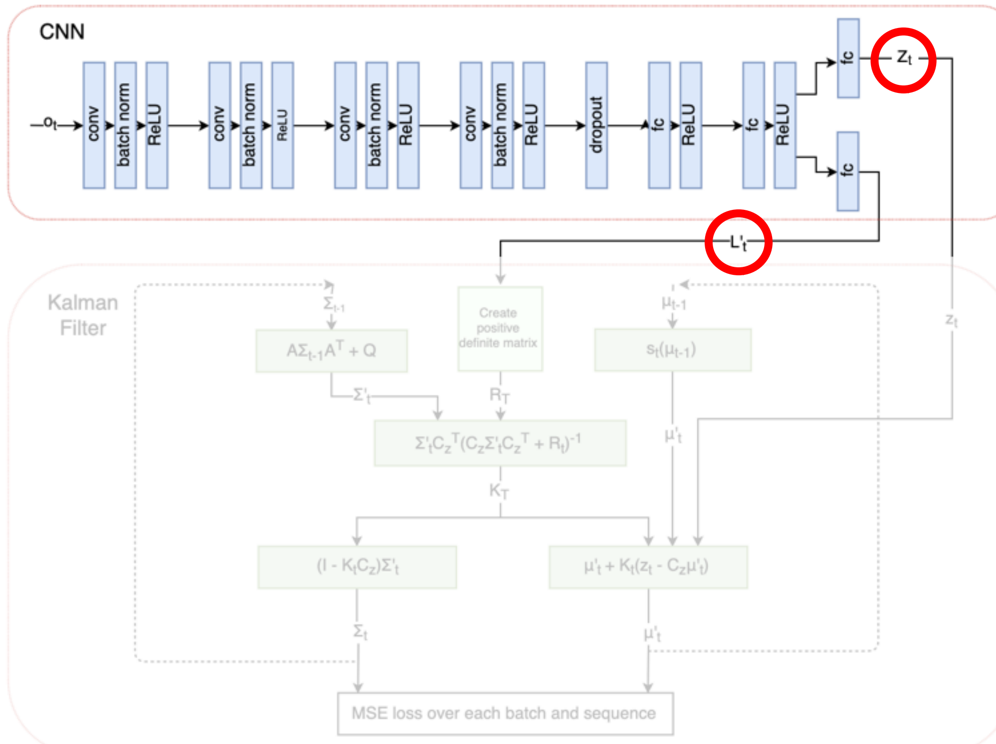
# Tracking by Detection



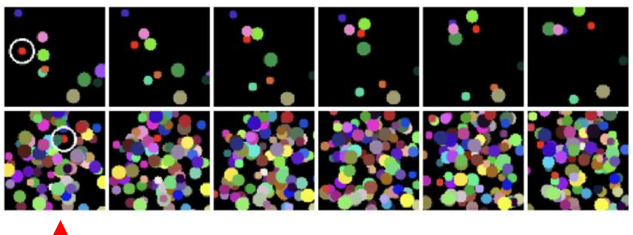
# Differentiable Kalman Filter - Structure

$$g(I_t) = z_t \approx x_t$$



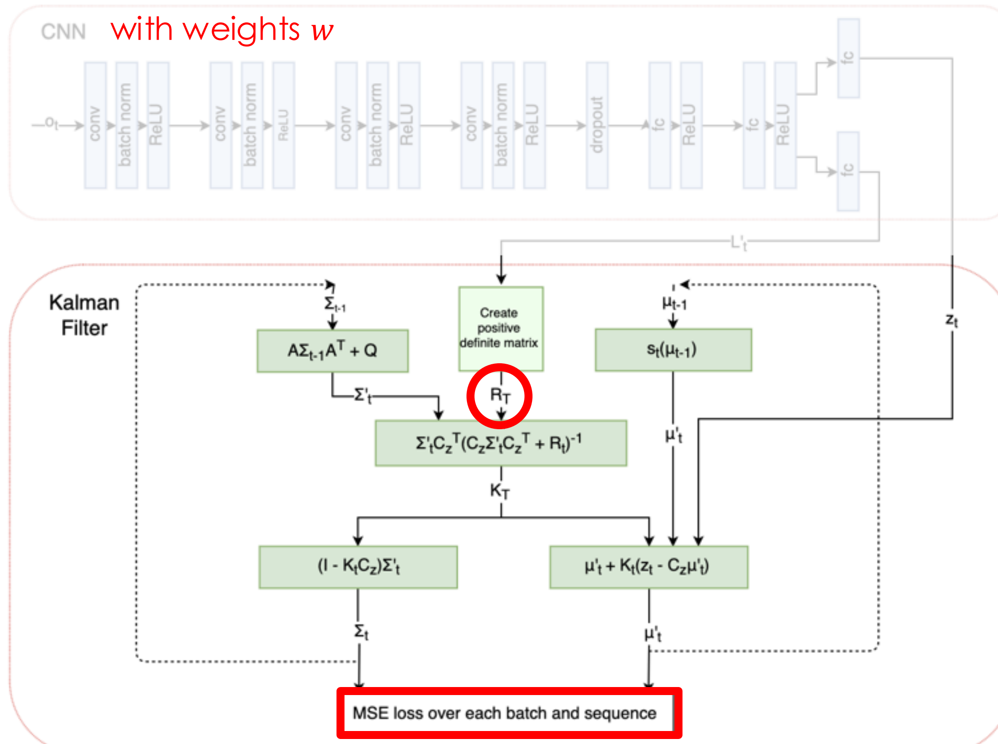


# Differentiable Kalman Filter - Structure



R is high if red disk is occluded

$$L'L^T = R$$




$$\frac{\delta Loss}{\delta w}$$

# Differentiable Kalman Filter – Loss Function

Ground truth state    Network weights

Belief

$$L(\mathbf{l}_{0\ldots T}, \overline{\mu_{0\ldots T}, \boldsymbol{\Sigma}_{0\ldots T}}, \mathbf{w}) =$$

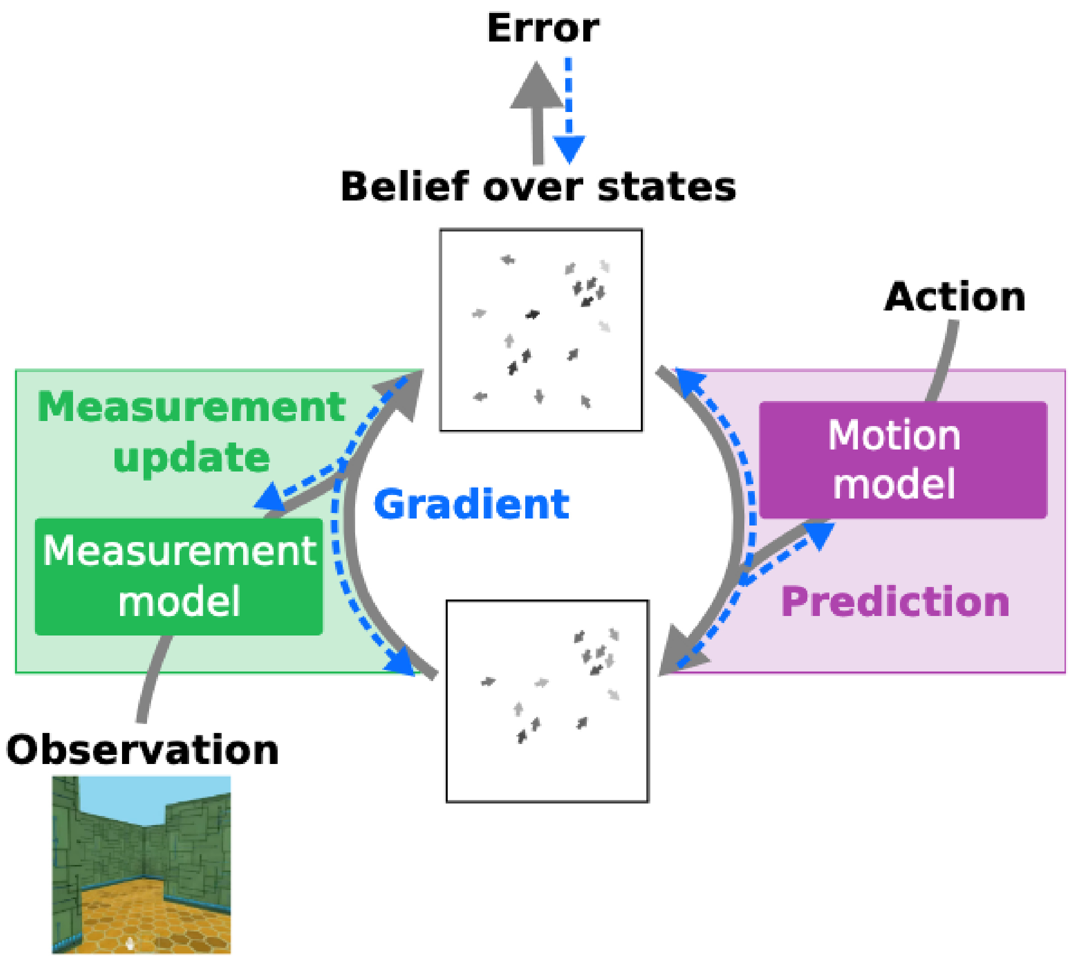
$$\lambda_1 \sum_{t=0}^{T} \frac{1}{2} \underbrace{((\mathbf{l}_t - \mu_t)^T \Sigma_t^{-1} (l_t - \mu_t) + \log(|\Sigma_t|))}_{\text{Negative log likelihood of ground truth given current belief}} + \lambda_2 \sum_{t=0}^{T} \| \underbrace{(l_t - \mu_t)}_{\text{Mean-Squared Error}} \|_2 + \lambda_3 \underbrace{\| \mathbf{w} \|_2}_{\text{Regularization}}$$

# Differentiable Kalman Filter – Experiments and Baselines

Table 1: Benchmark Results

| **State Estimation Model** | **# Parameters** | **RMS test error** $\pm\sigma$ |
|---|---|---|
| feedforward model | 7394 | $0.2322 \pm 0.1316$ |
| piecewise KF | 7397 | $0.1160 \pm 0.0330$ |
| LSTM model (64 units) | 33506 | $0.1407 \pm 0.1154$ |
| LSTM model (128 units) | 92450 | $0.1423 \pm 0.1352$ |
| **BKF (ours)** | **7493** | $\mathbf{0.0537} \pm \mathbf{0.1235}$ |

# Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors. Jonschkowski et al. RSS 2018.



# CS231A

## Computer Vision: From 3D Reconstruction to Recognition

Next lecture:

Neural Radiance Fields for Novel View Synthesis