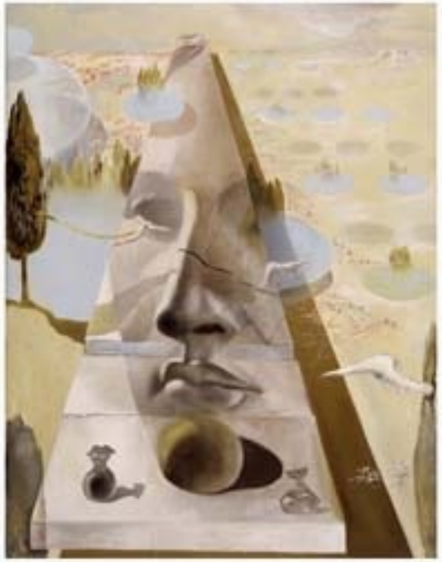
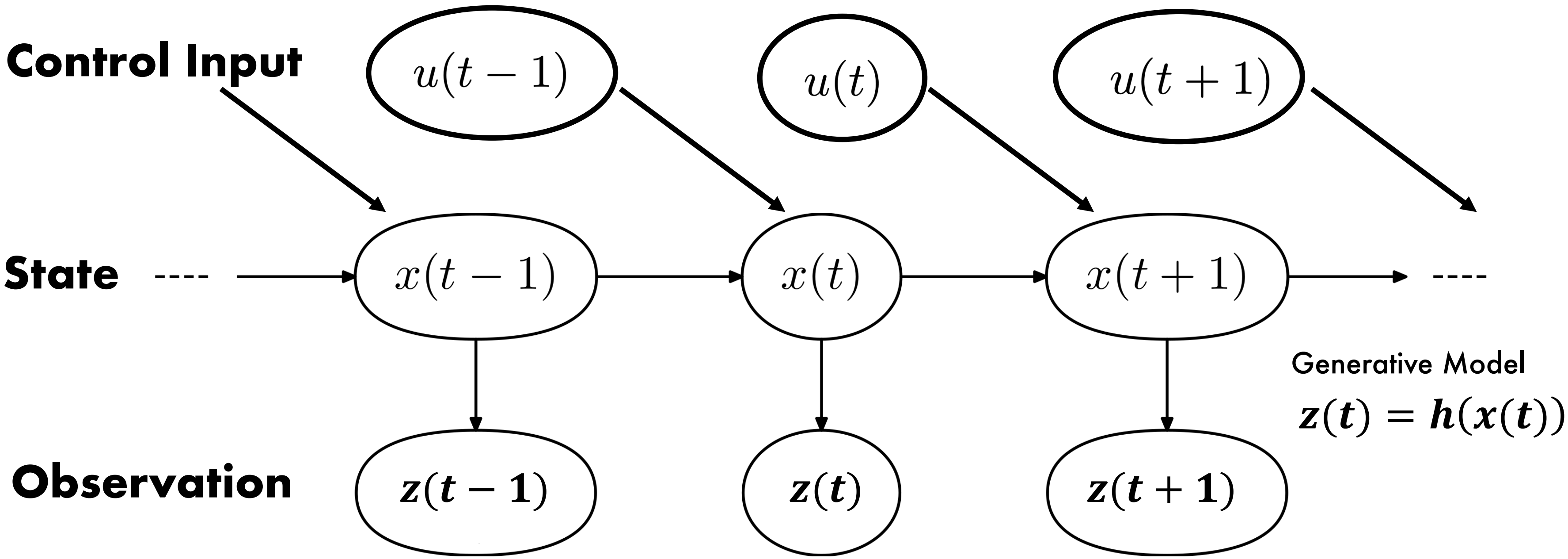
CS231A

Computer Vision:
From 3D Reconstruction to Recognition



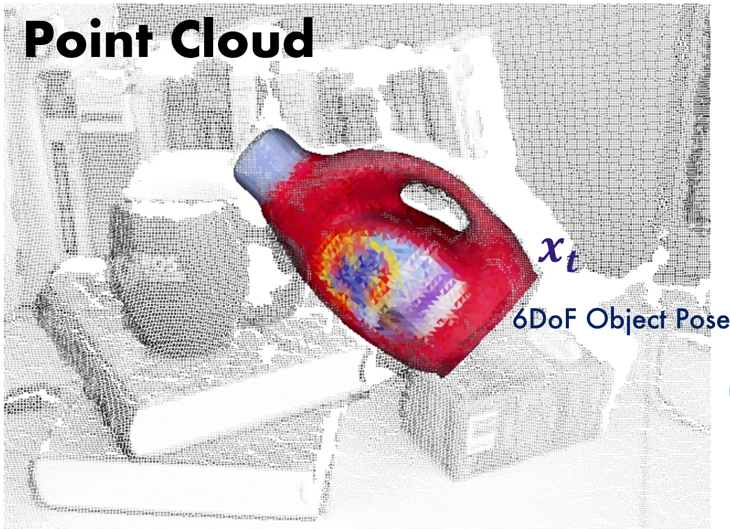
Optimal Estimation Cont'

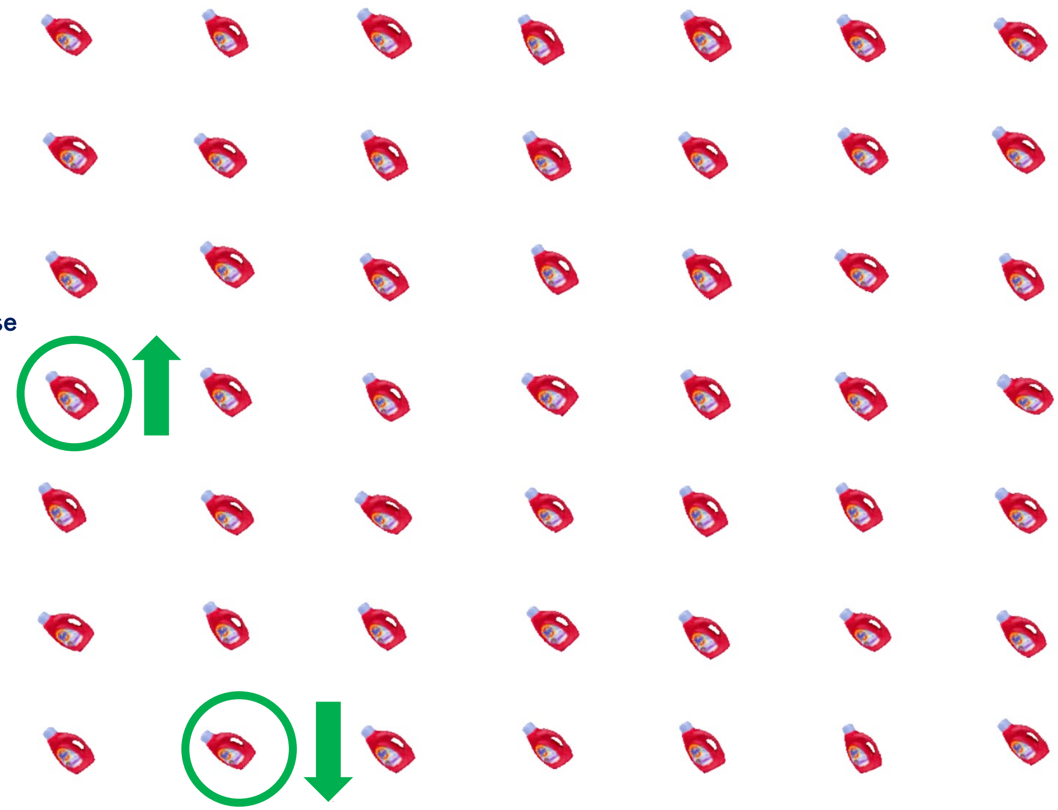
Graphical Model of System to Estimate



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 $\mathcal{X}_t$

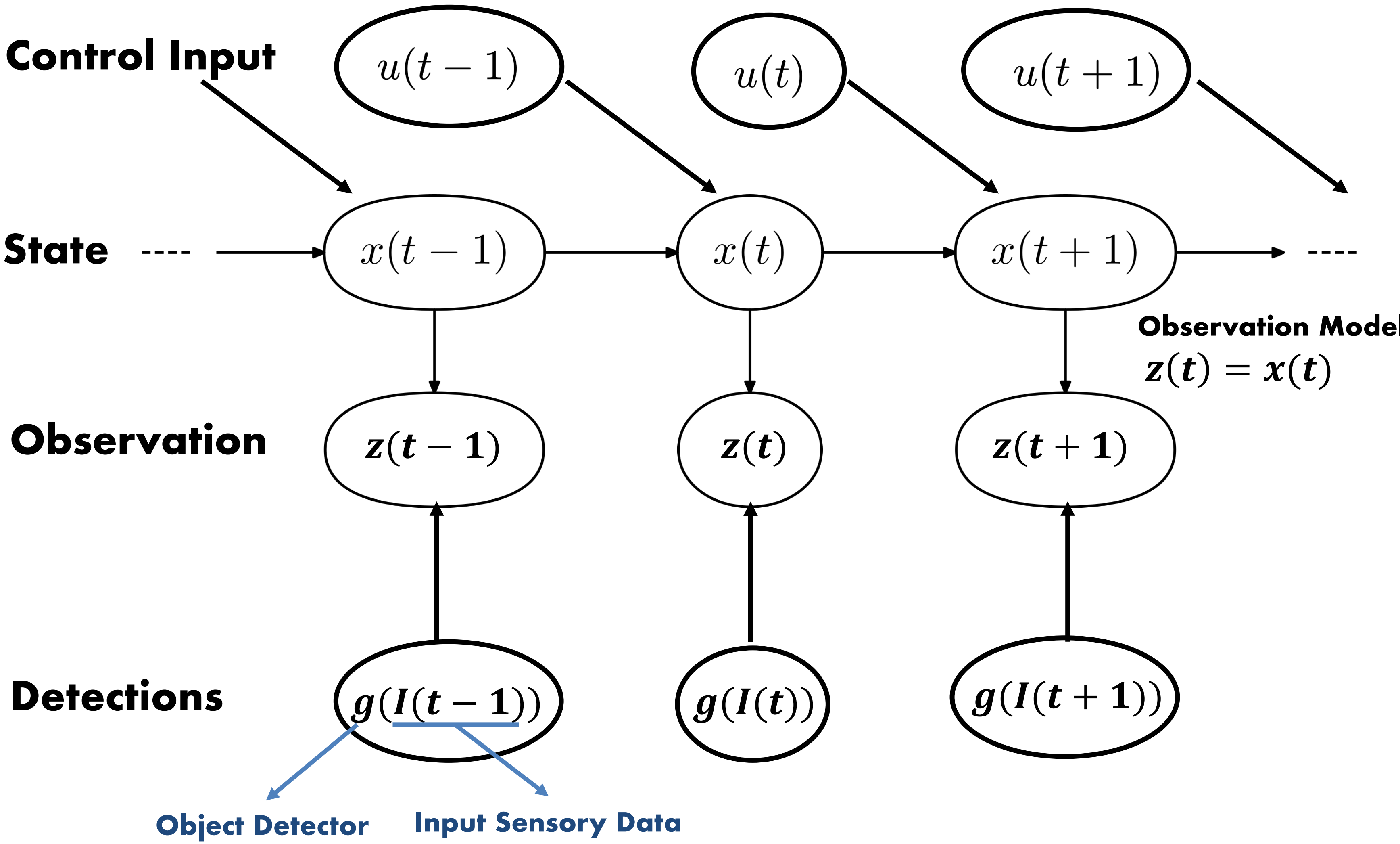
endfor

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

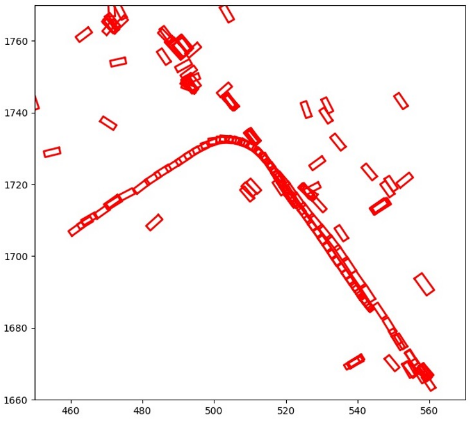
Tracking by Detection



Problem Statement: Input

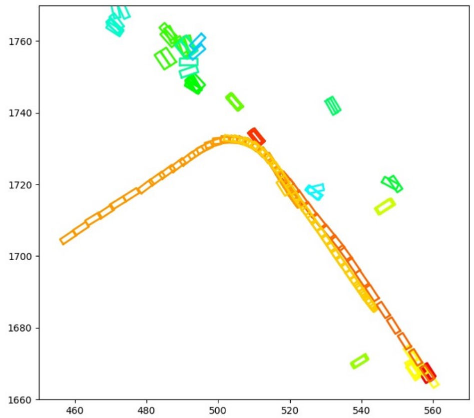
Probabilistic 3d multi-object tracking for autonomous driving. H Chiu, A Prioletti, J Li, J Bohg
arXiv preprint arXiv:2001.05673

- 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)



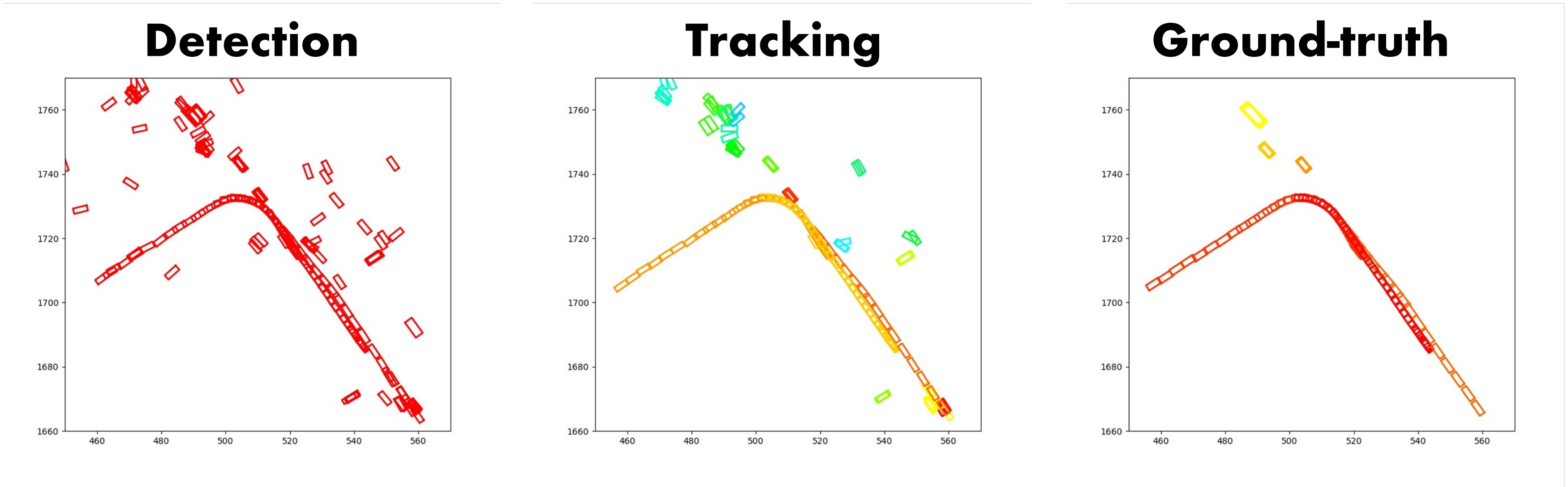
Problem Statement: Output

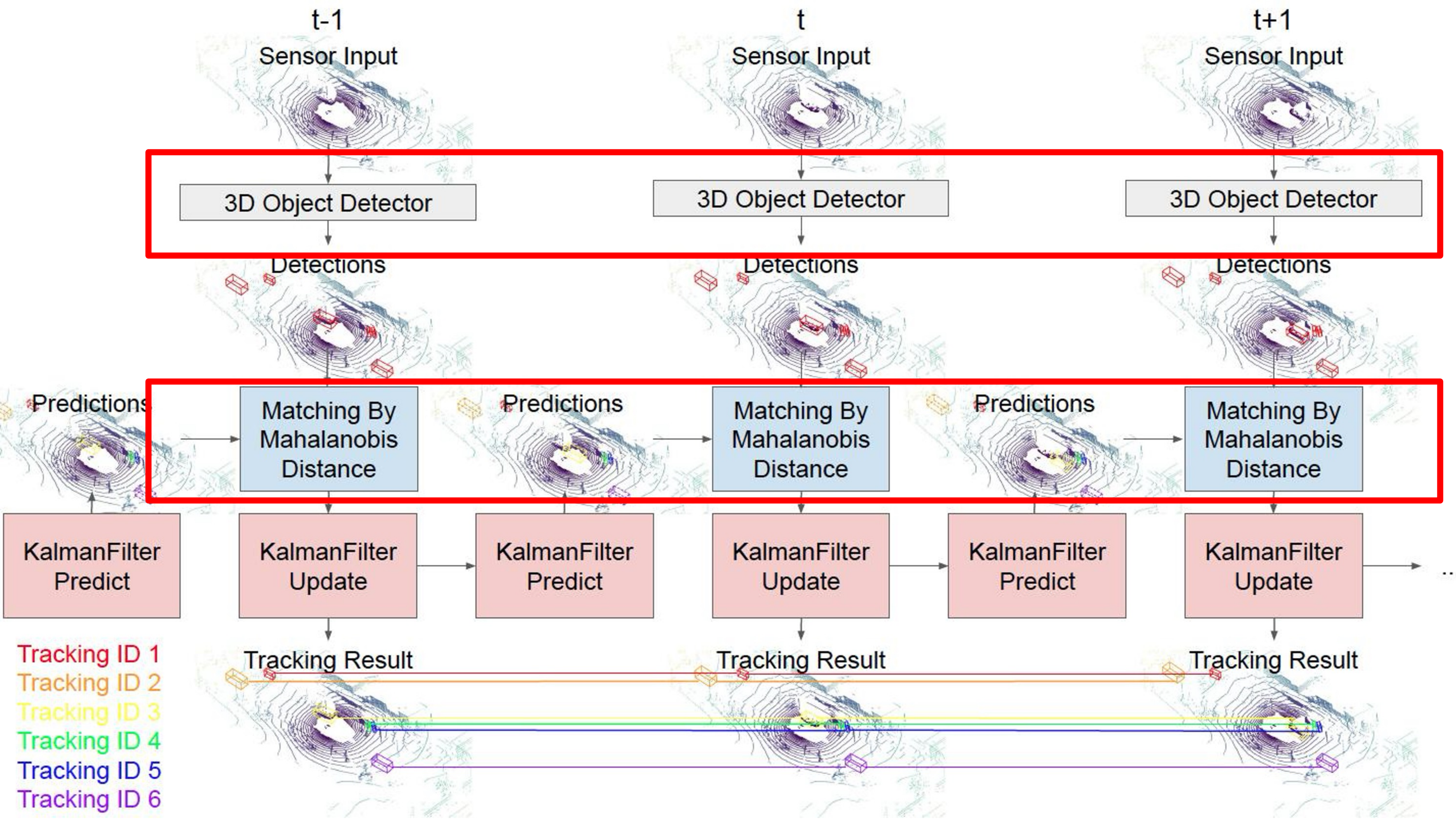
- Tracking object bounding boxes at each frame in a sequence
- Each tracking bounding box is represented by:
  - center position (x, y, z), rotation angle along the z-axis (a), and the scale (l, w, h)
  - category label (car, pedestrian, ...), confidence score (c)
  - tracking id: one unique tracking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



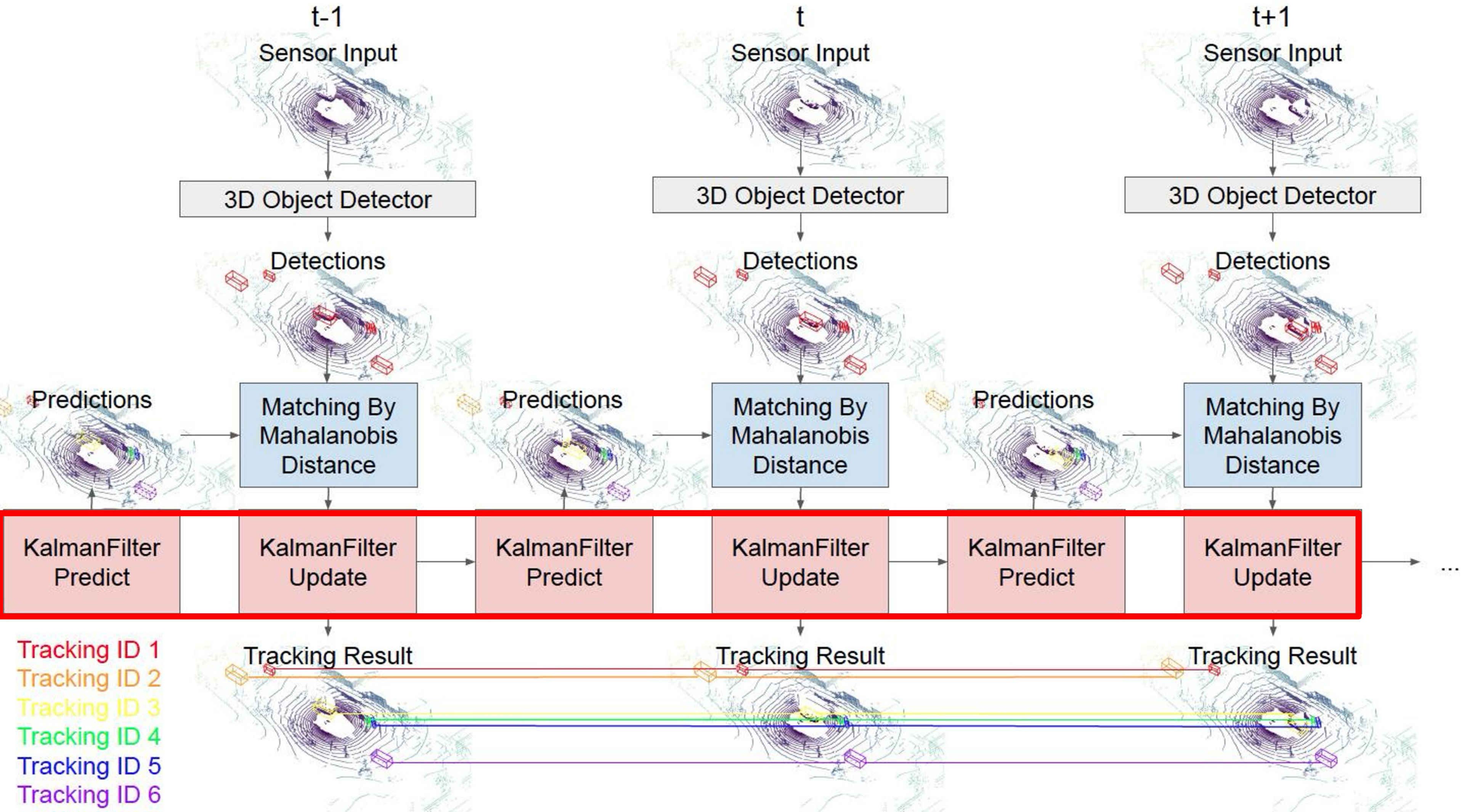
Why Tracking?

- 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





t-1
Sensor Input
t
Sensor Input
t+1
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

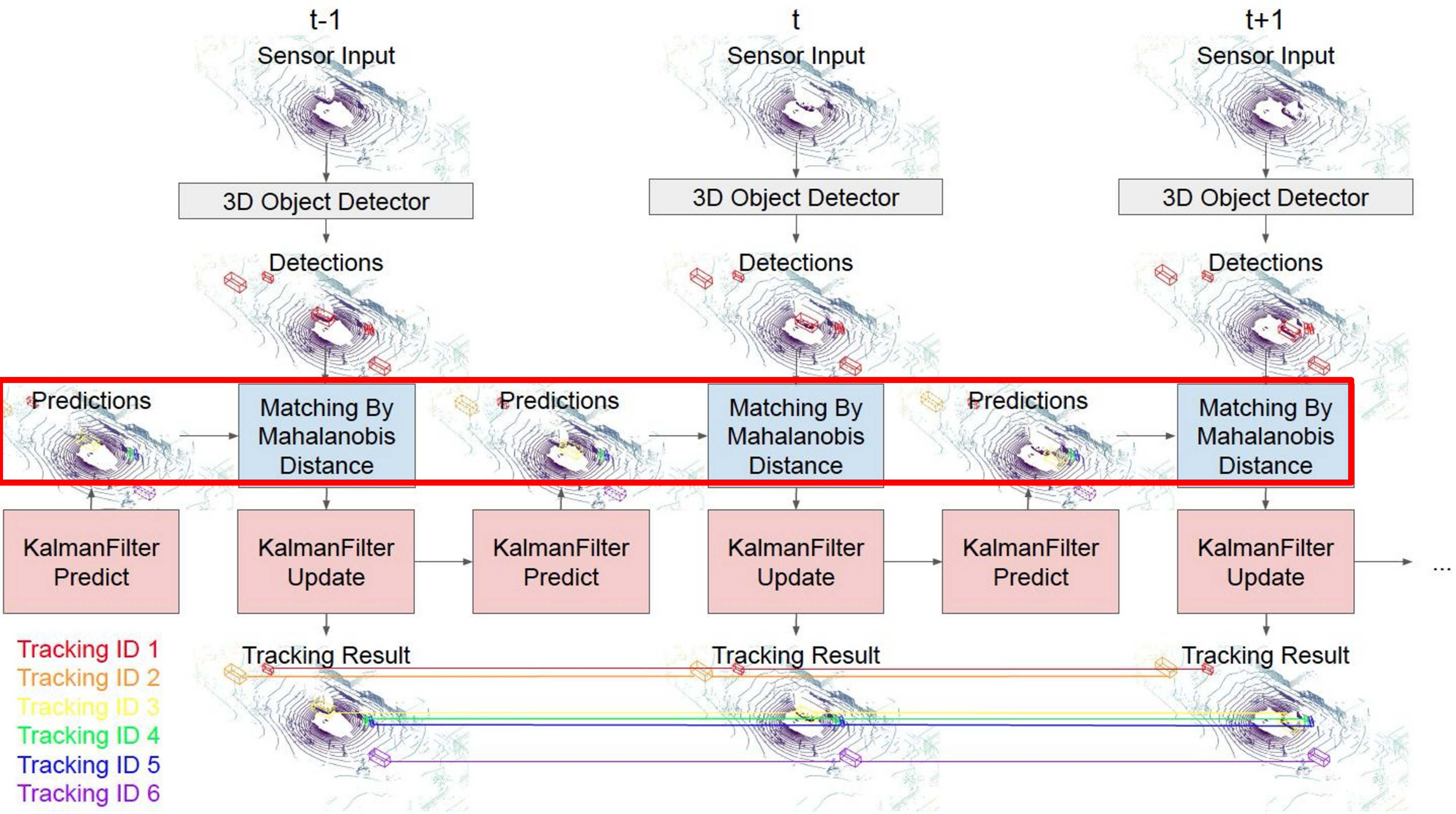
Kalman Filter for Tracking

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:

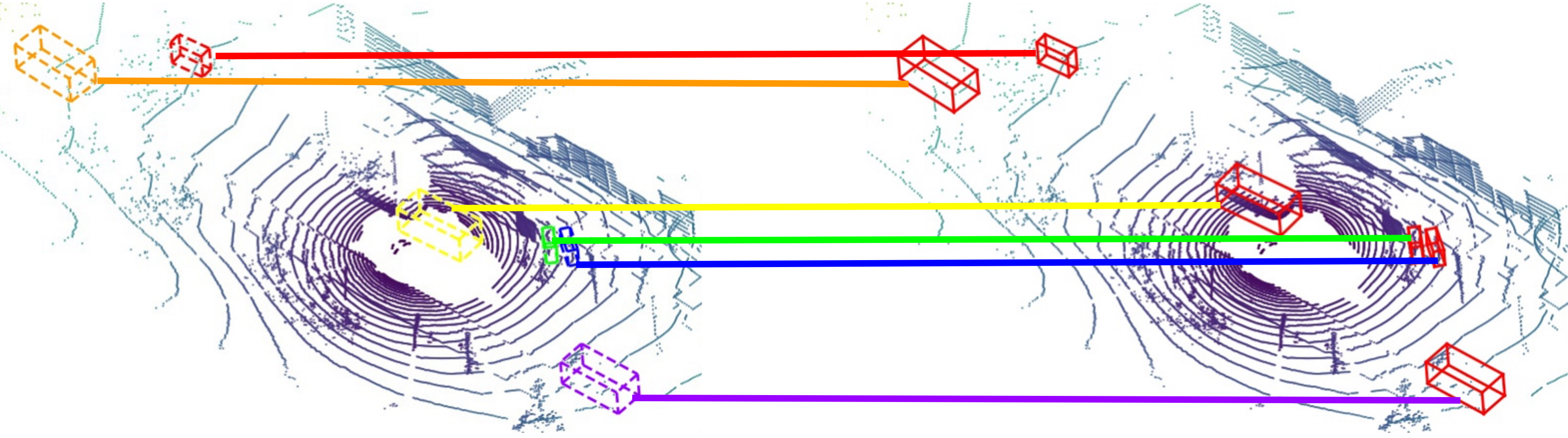
$$\begin{aligned}
\hat{x}_{t+1} &= x_t + d_{x_t} + q_{x_t}, & \hat{d}_{x_{t+1}} &= d_{x_t} + q_{d_{x_t}} & \hat{l}_{t+1} &= l_t \\
\hat{y}_{t+1} &= y_t + d_{y_t} + q_{y_t}, & \hat{d}_{y_{t+1}} &= d_{y_t} + q_{d_{y_t}} & \hat{w}_{t+1} &= w_t \\
\hat{z}_{t+1} &= z_t + d_{z_t} + q_{z_t}, & \hat{d}_{z_{t+1}}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


t-1
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

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 μ_t, Σ_t

Data Association

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

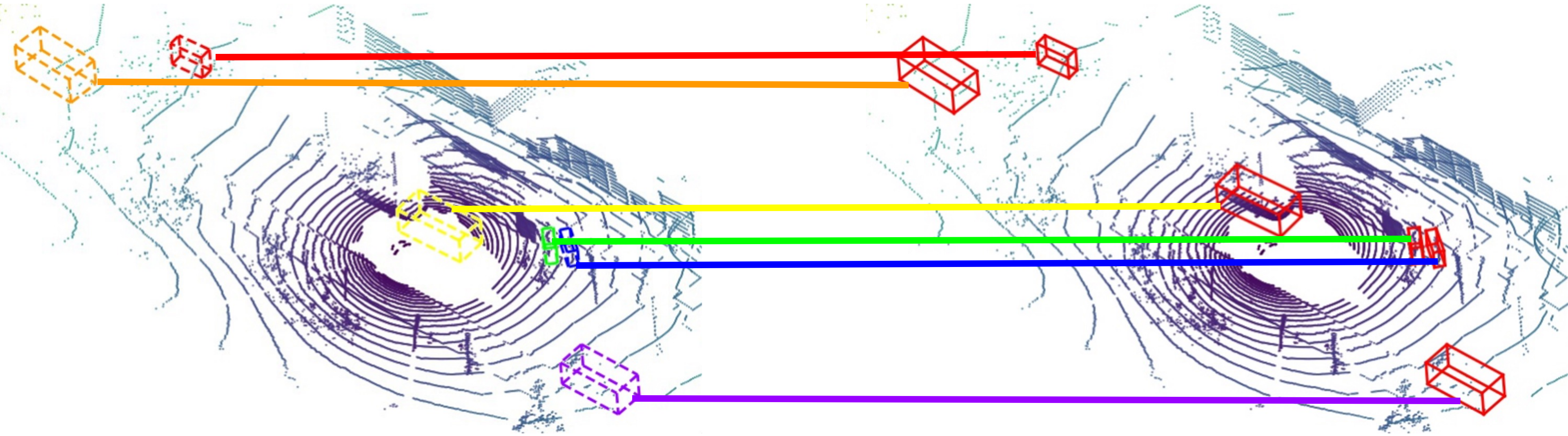
If $m > 3 * \sigma$ then reject as outlier. 99.7% of values lie within 3*standard deviation.

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.

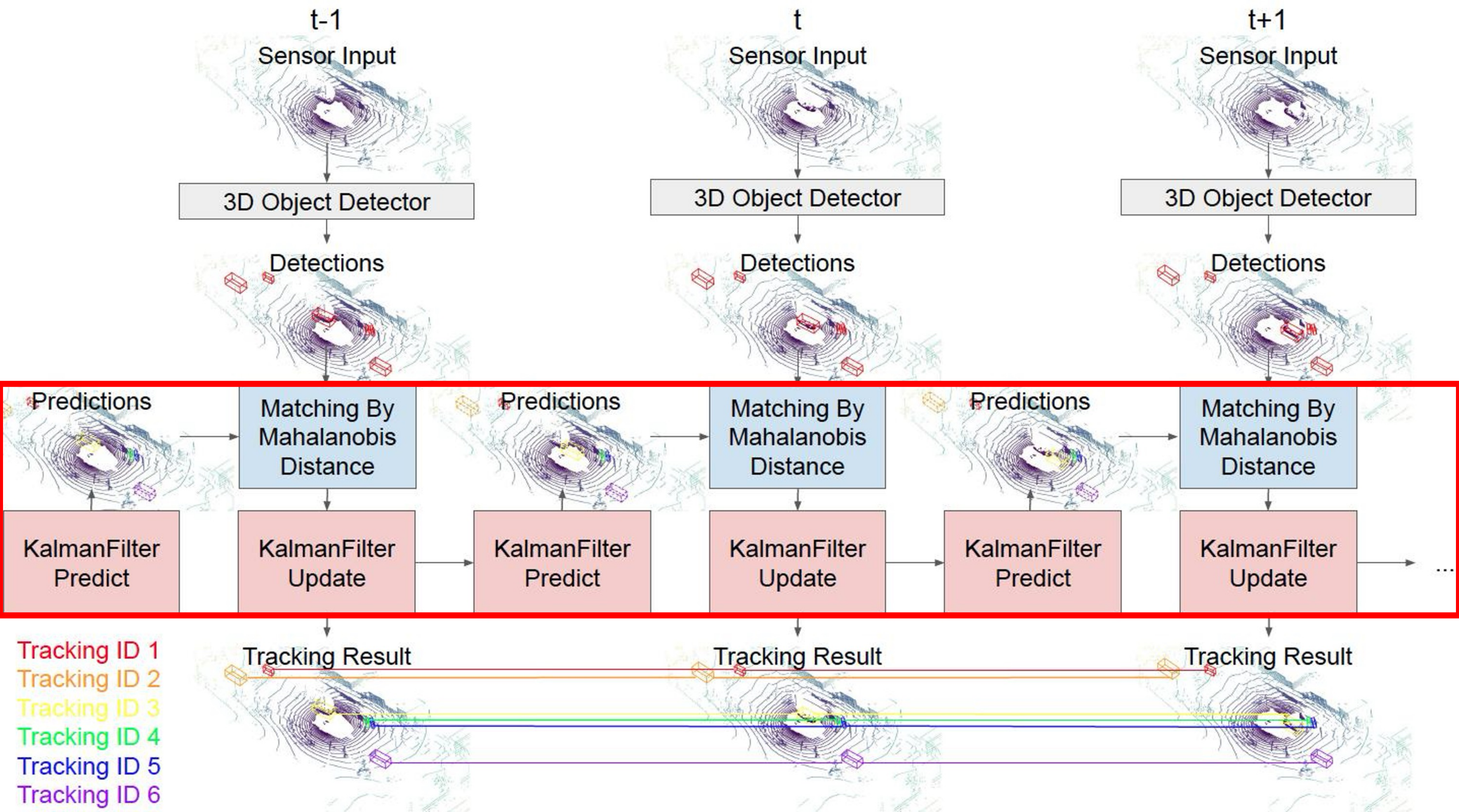
Taking the uncertainty information from the prediction into account.

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

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

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

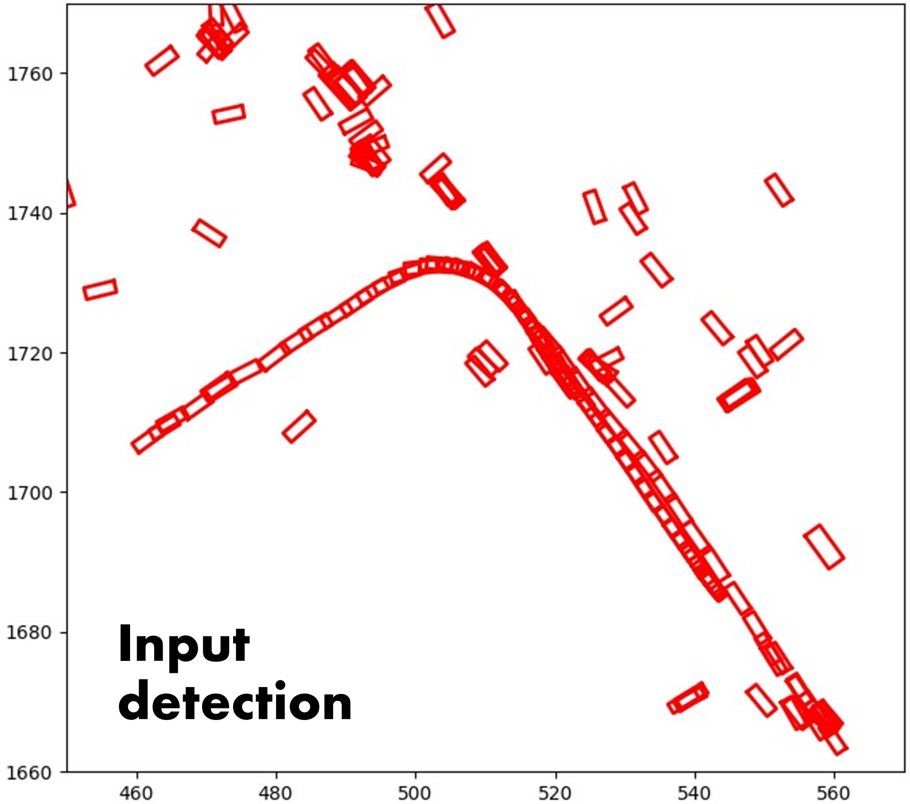
$K_t = \bar{\Sigma}_t \, C_t^T (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1}$

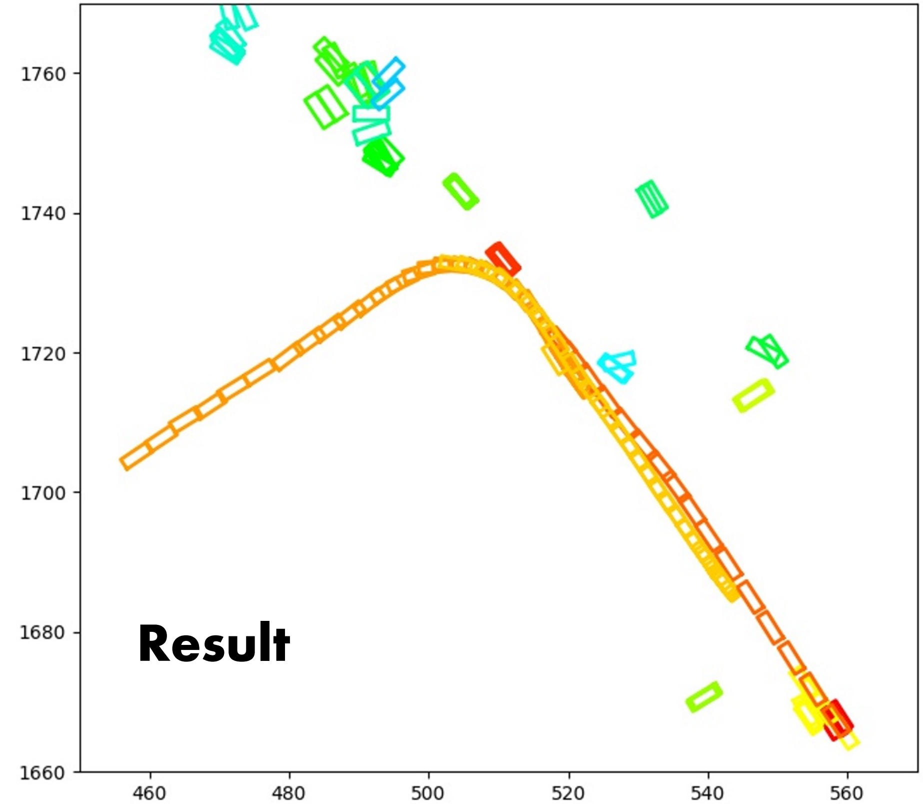
$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \, \bar{\mu}_t)$

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

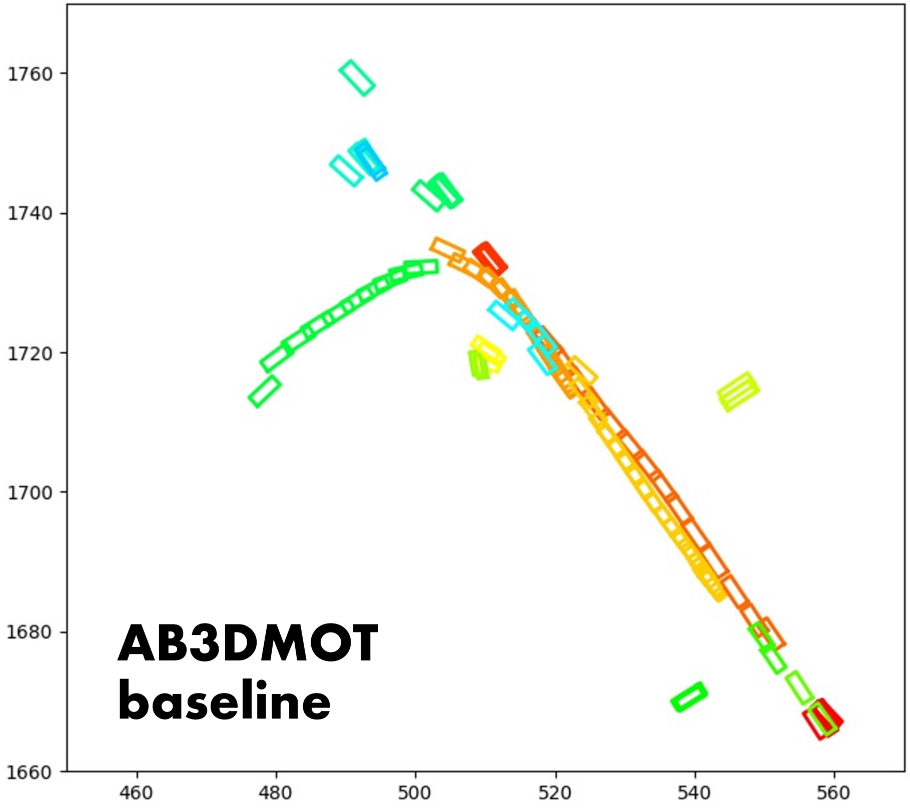
return μ_t, Σ_t

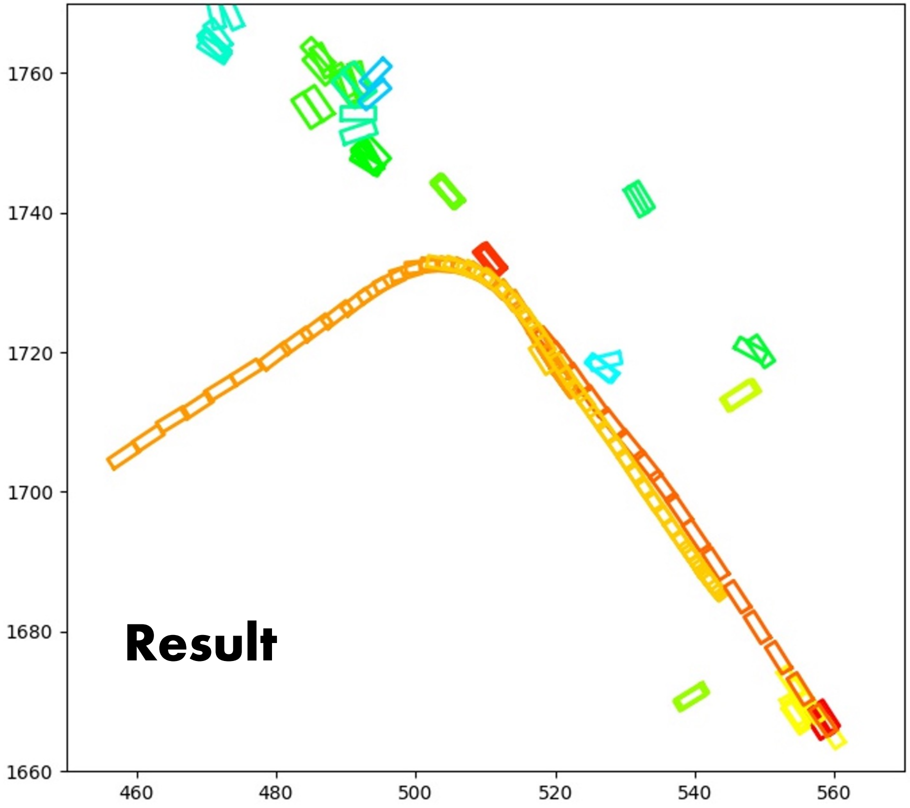
Qualitative Results



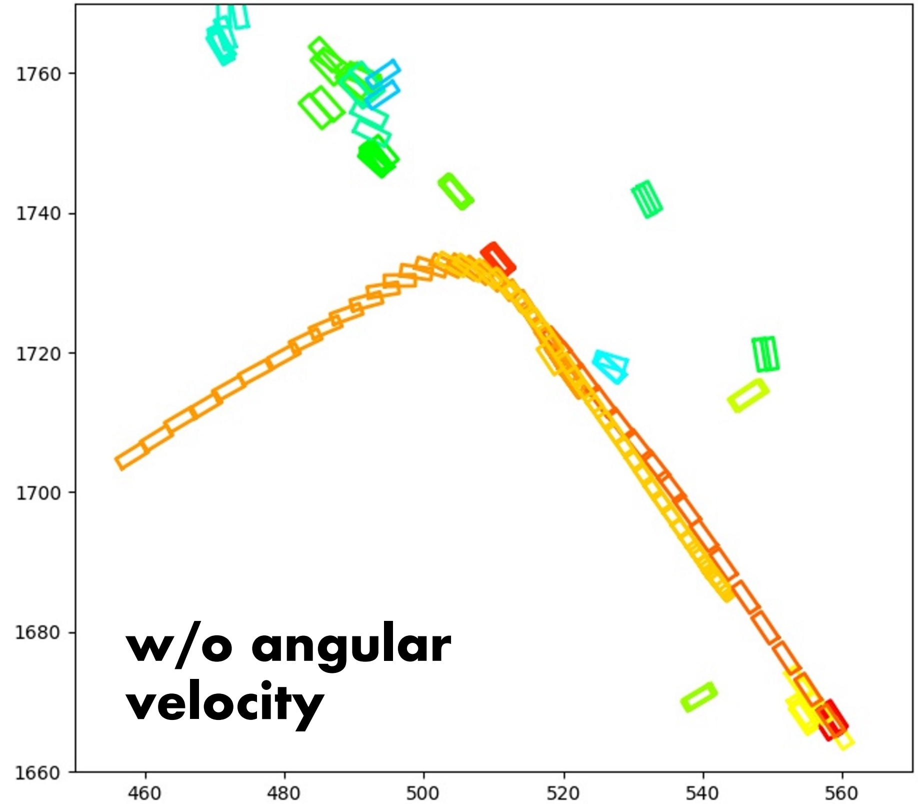


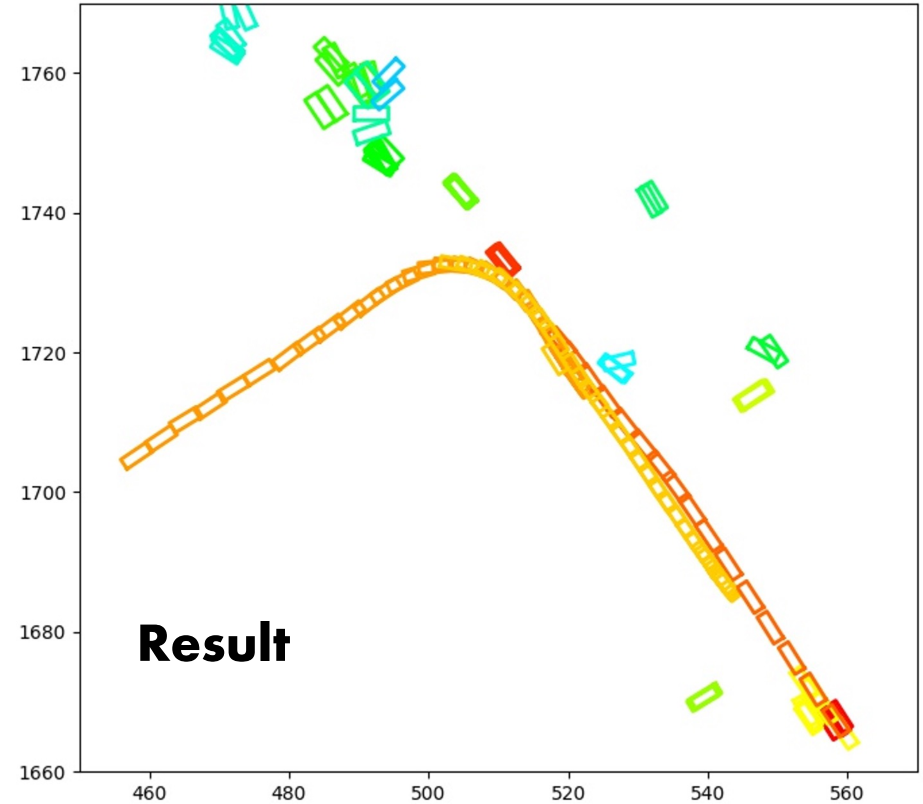
Qualitative Results



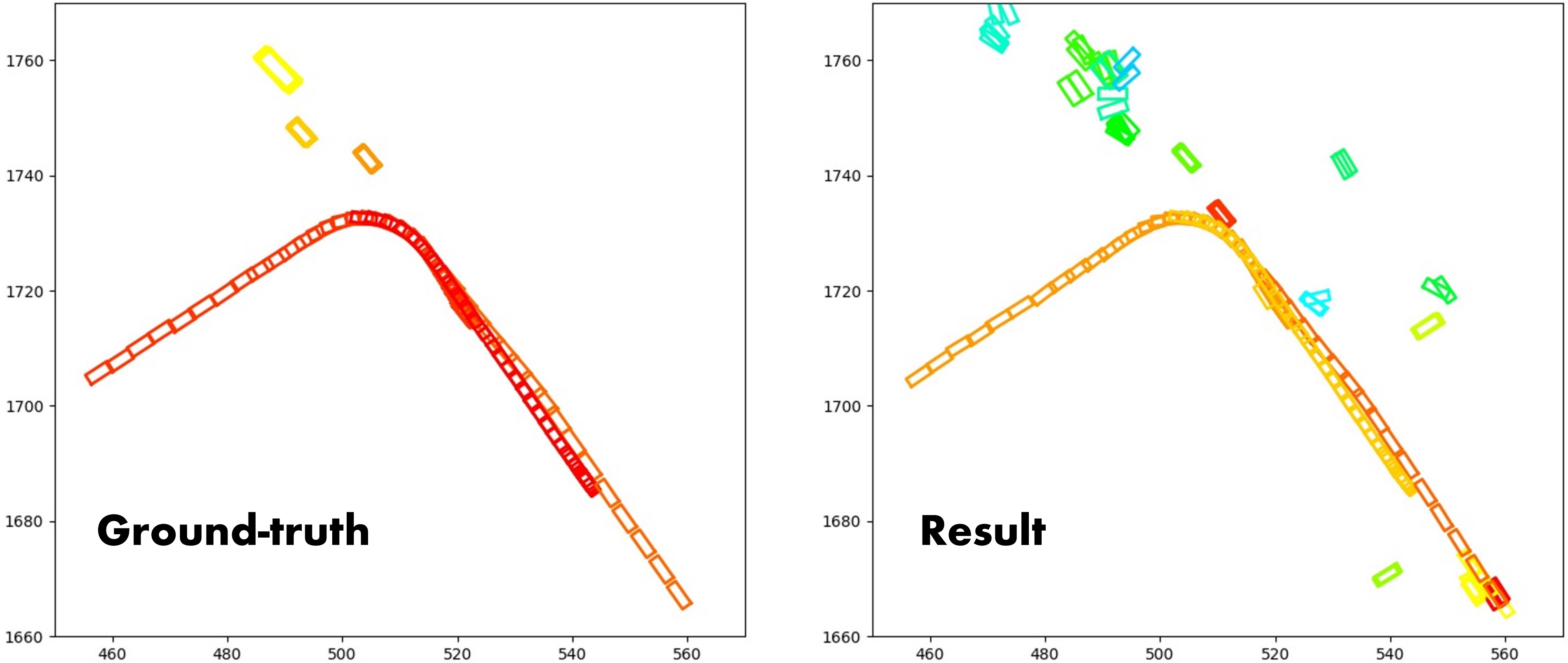


Qualitative Results





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.

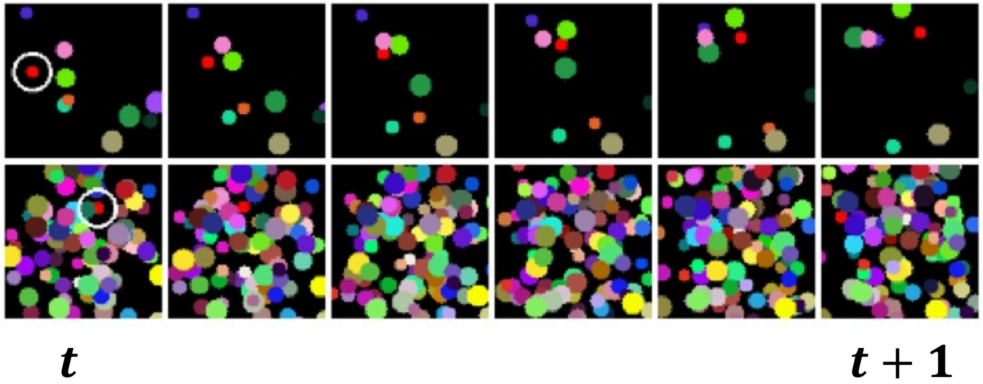
- prevents overfitting through regularization
- Avoids manual tuning and modeling

Estimators. Haarnoja et al. NeurIPS 2016

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

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

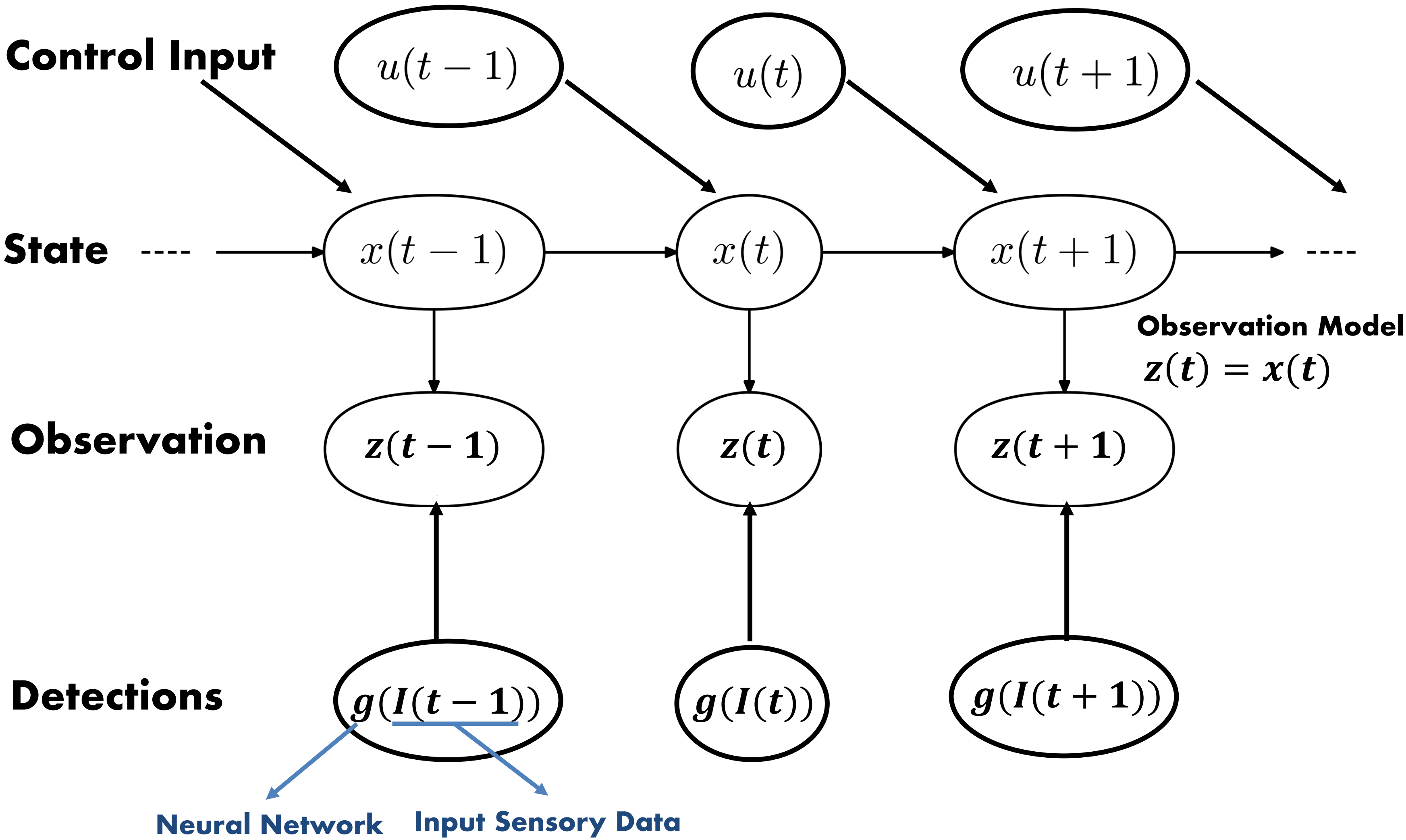
Track red disk position



Example Sequence w/ few occlusions

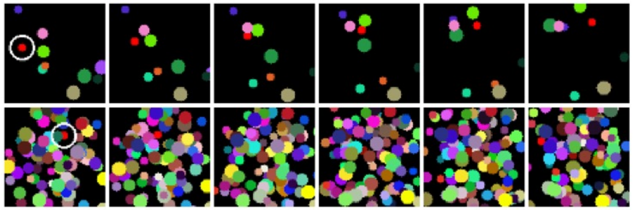
Example Sequence w/ many occlusions

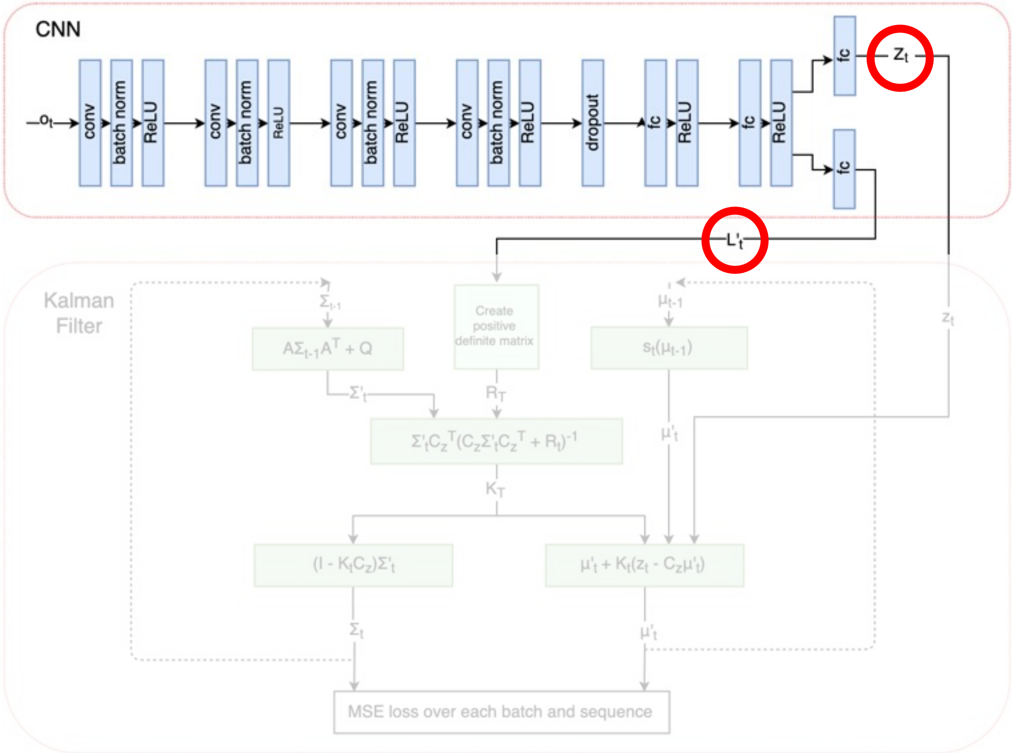
Tracking by Detection



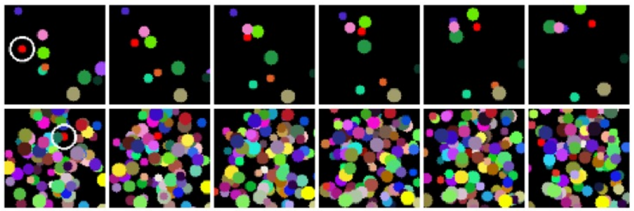
Differentiable Kalman Filter - Structure

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



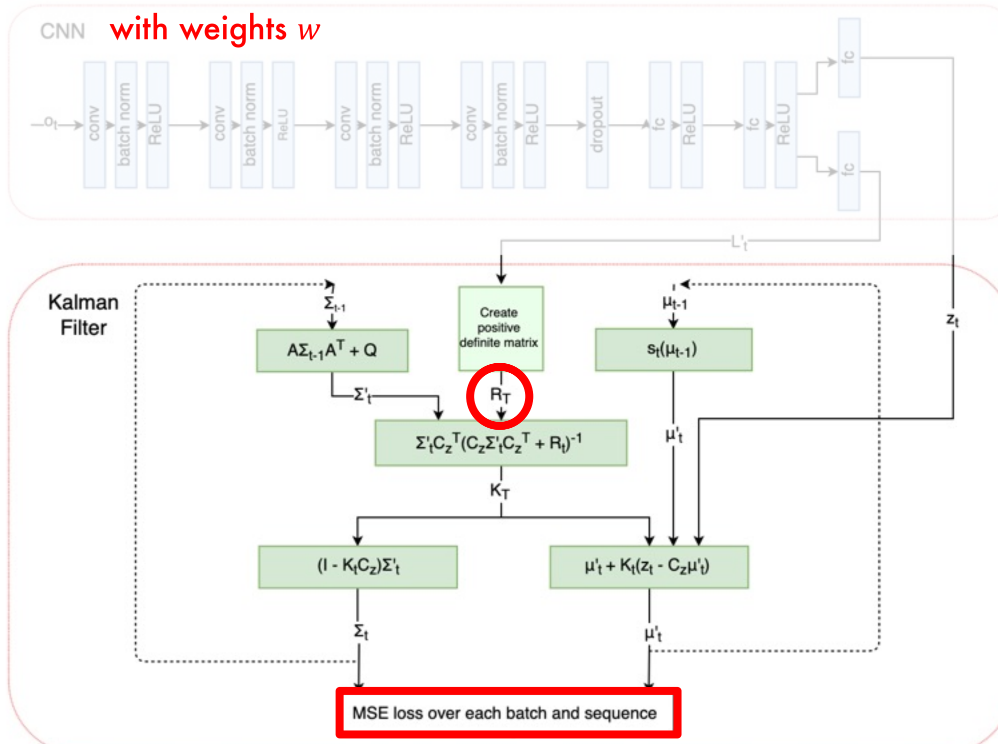


Differentiable Kalman Filter - Structure



R is high if red disk is occluded

$$L' L^T = R$$



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

Differentiable Kalman Filter – Loss Function

$$L(\overset{\text{Ground truth state}}{\mathbf{l}_{0\ldots T}}, \overbrace{\mu_{0\ldots T}, \boldsymbol{\Sigma}_{0\ldots T}}^{\text{Belief}}, \overset{\text{Network weights}}{\mathbf{w}}) =$$

$$\lambda_1 \sum_{t=0}^{T} \frac{1}{2} \underbrace{((\mathbf{l}_t - \mu_t)^T \Sigma_t^{-1} (l_t - \mu_t) + \log(|\Sigma_t|))}_{\text{Negative log likelihood of ground truth given current belief}} + \lambda_2 \sum_{t=0}^{T} \parallel \underbrace{(l_t - \mu_t)}_{\text{Mean-Squared Error}} \parallel_2 + \lambda_3 \underbrace{\parallel \mathbf{w} \parallel_2}_{\text{Regularization}}$$

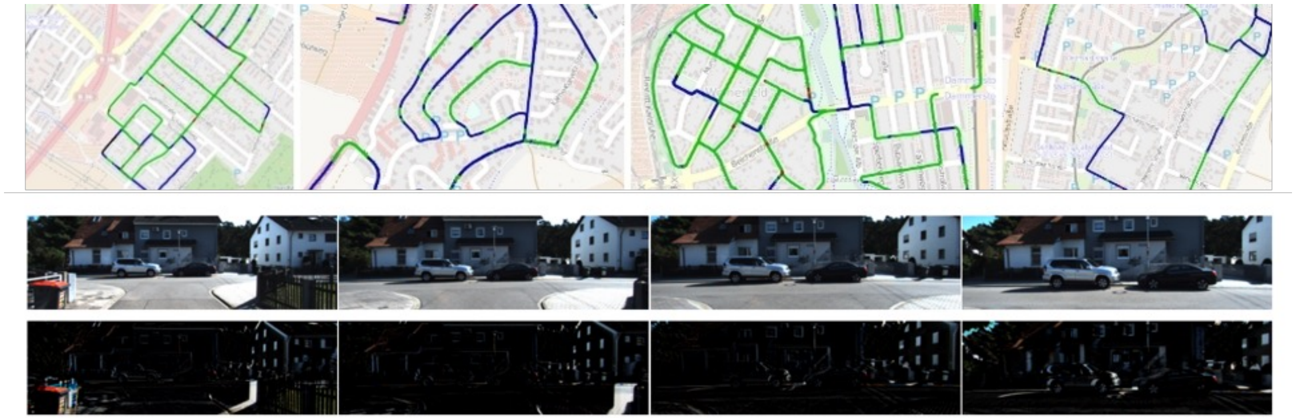
Differentiable Kalman Filter – Experiments and Baselines

Table 1: Benchmark Results

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

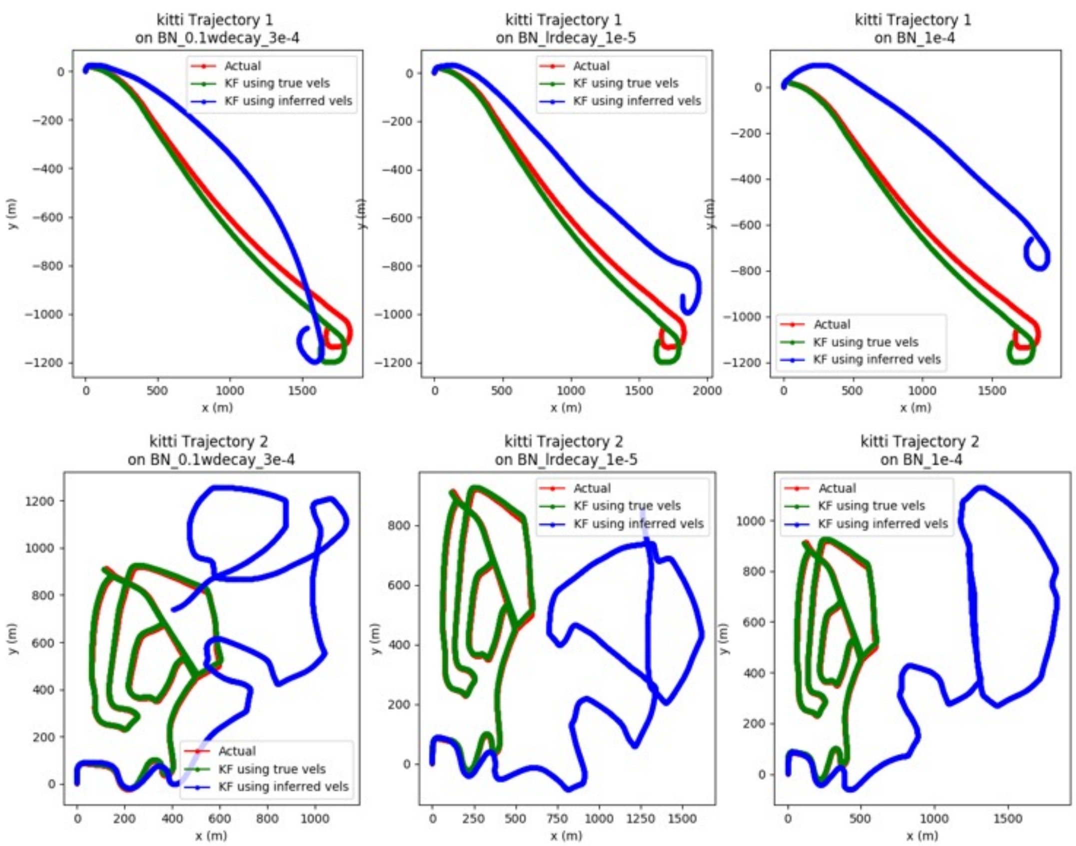
Differentiable Kalman Filter – Experiments and Baselines

- **Kitti – Visual Odometry Datatset**
- **22 stereo sequences with LIDAR**
  - **11 sequences with ground truth (GPS/IMU data)**
  - **11 sequences without ground truth (for evaluation)**



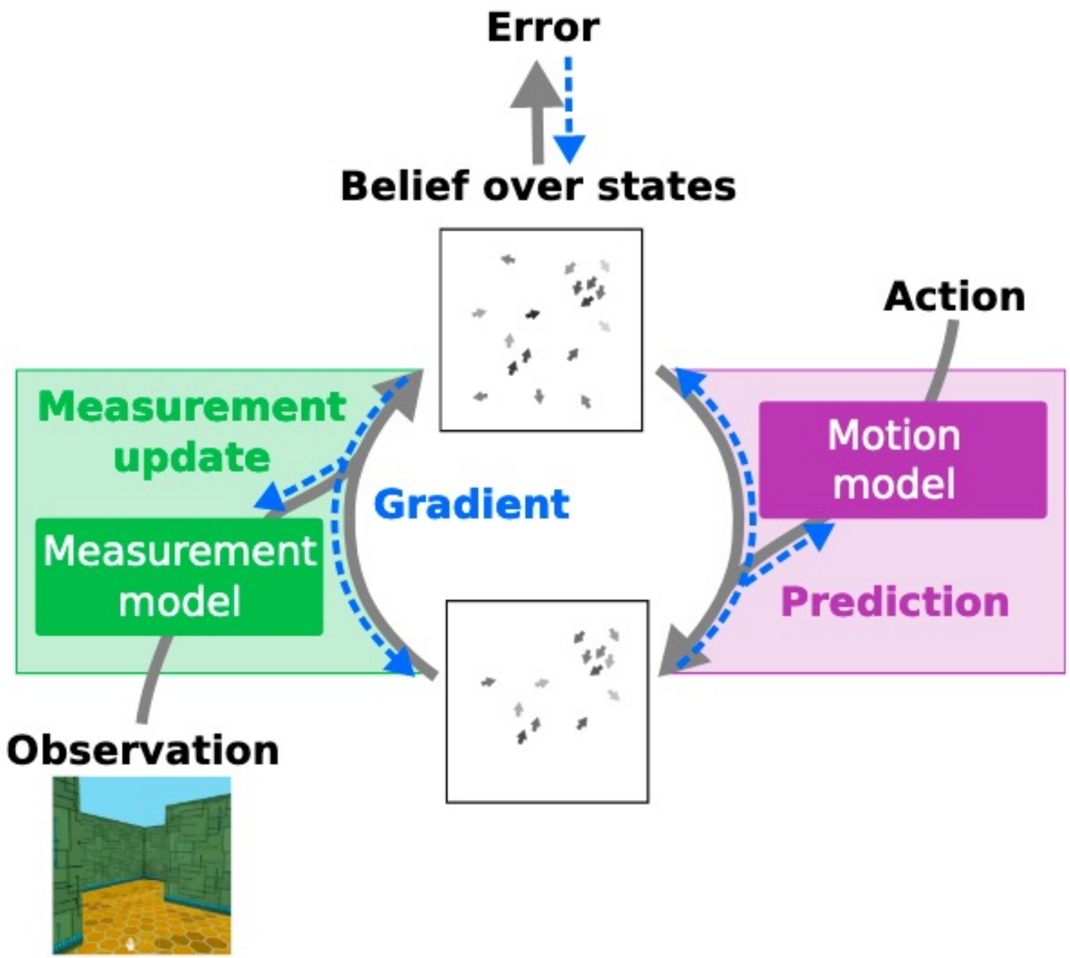
Differentiable Kalman Filter – Experiments and Baselines

Results reproduced by Claire Chen

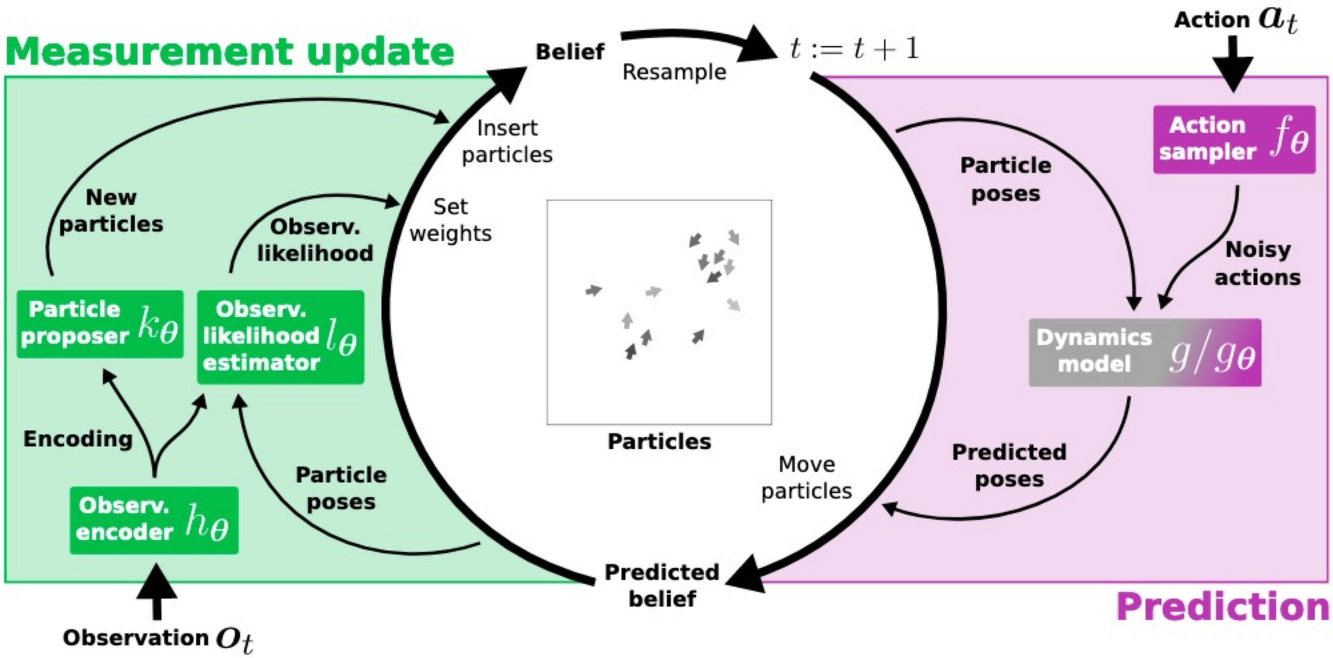


Blue – Result of BackpropKF
Red – Ground truth
Green – w/ Ground truth velocities

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

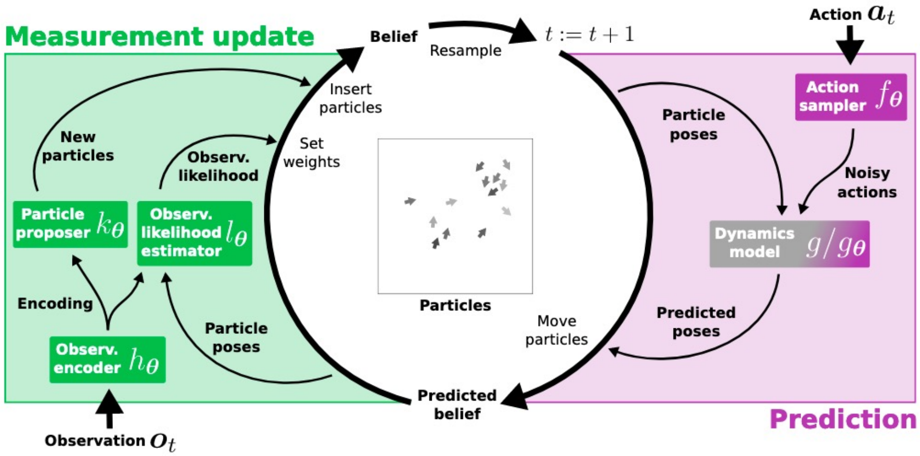


Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors. Jonschkowski et al. RSS 2018.



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

- Prediction Step



w_t weights

a_t actions

s_t states

o_t observations

$$\mathrm{bel}(s_t) = (S_t, w_t)$$

$$\hat{a}_t^{[i]} = a_t + f_\theta(a_t, \epsilon^{[i]} \sim \mathcal{N}),$$ Action sampler

$$s_t^{[i]} = s_{t-1}^{[i]} + g(s_{t-1}^{[i]}, \hat{a}_t^{[i]}),$$ Dynamics model

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

- Measurement Update

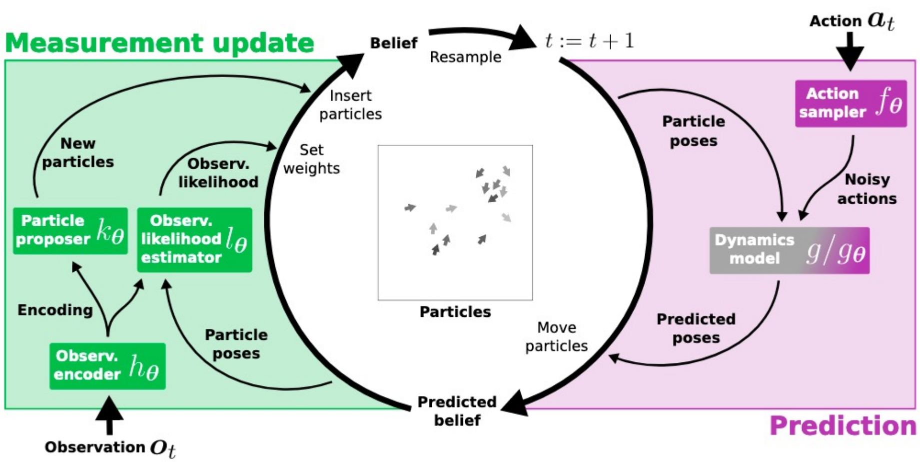
w_t weights

a_t actions

s_t states

o_t observations

e_t encoding



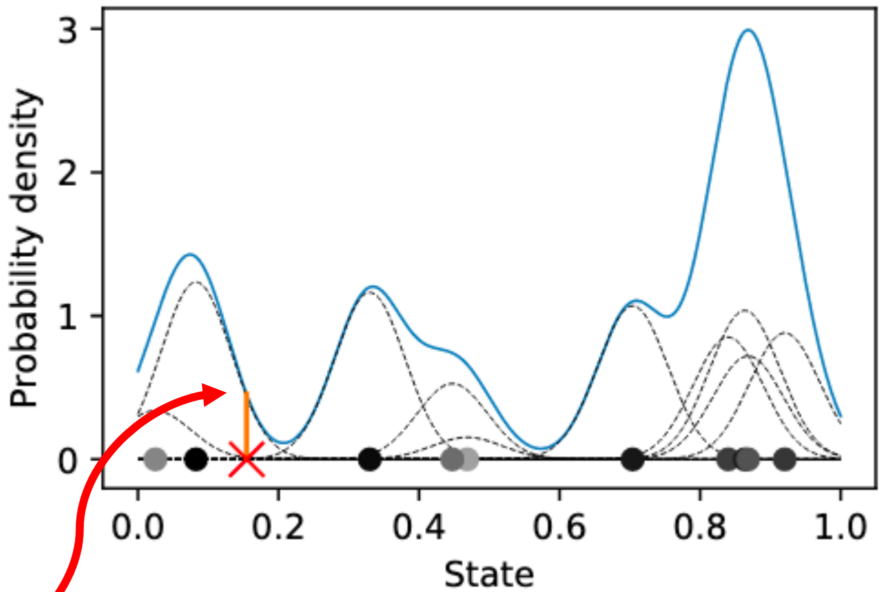
$$e_t = h_\theta(o_t),$$ Observation Encoder

$$s_t^{[i]} = k_\theta(e_t, \delta^{[i]} \sim B),$$ Particle Proposer

$$w_t^{[i]} = l_\theta(e_t, s_t^{[i]}),$$ Observation likelihood estimator

Differentiable Particle Filter – Loss Function

- Supervised learning given data $\boldsymbol{o}_{1:T}, \boldsymbol{a}_{1:T}, \boldsymbol{s}^*_{1:T}$

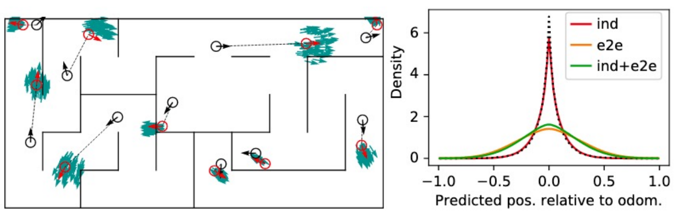


$\mathrm{bel}(\boldsymbol{s}^*_t)$

$$\boldsymbol{\theta}^* = \mathrm{argmin}_{\boldsymbol{\theta}} - \log \mathrm{E}_t[\mathrm{bel}(\boldsymbol{s}^*_t; \boldsymbol{\theta})].$$

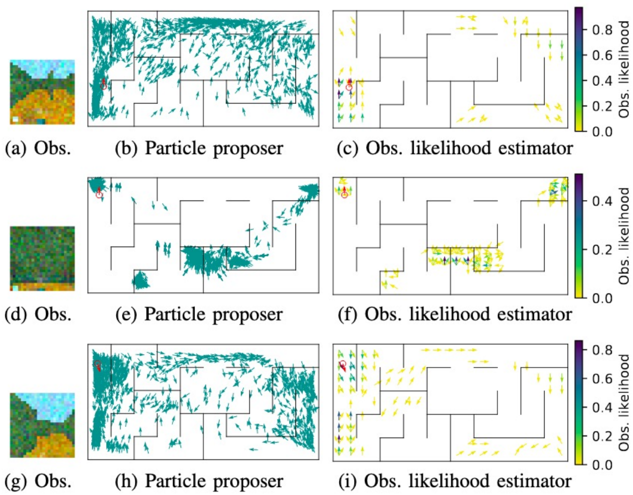
Maximizing the belief at the true state

Differentiable Particle Filter – Experiments and Baselines



(a) Predictions with learned motion model (b) Comparison of learned noise

Fig. 5: **Learned motion model**. (a) shows predictions (cyan) of the state (red) from the previous state (black). (b) compares prediction uncertainty in x to true odometry noise (dotted line).



(a) Obs. (b) Particle proposer (c) Obs. likelihood estimator
(d) Obs. (e) Particle proposer (f) Obs. likelihood estimator
(g) Obs. (h) Particle proposer (i) Obs. likelihood estimator

Fig. 6: **Learned measurement model**. Observations, corresponding model output, and true state (red). To remove clutter, the observation likelihood only shows above average states.

Differentiable Particle Filter – Experiments and Baselines

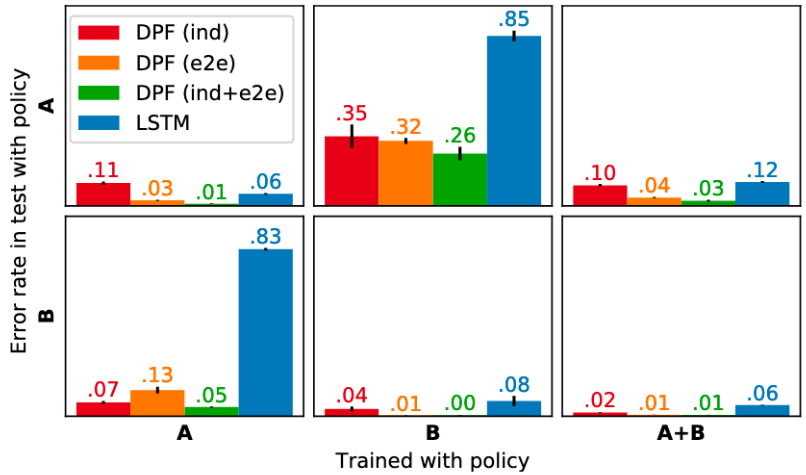
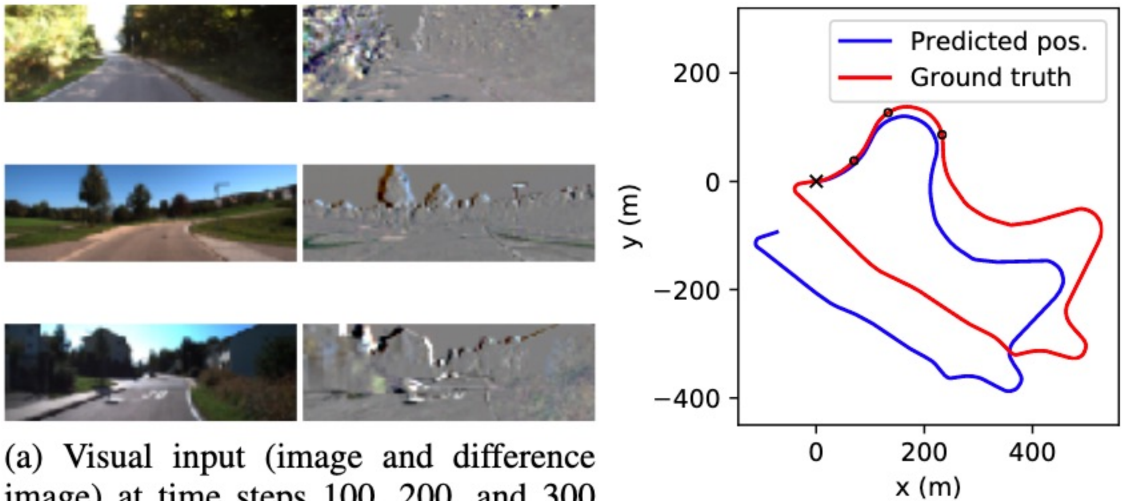


Fig. 9: **Generalization between policies** in maze 2. A: heuristic exploration policy, B: shortest path policy. Methods were trained using 1000 trajectories from A, B, or an equal mix of A and B, and then tested with policy A or B.

Differentiable Particle Filter – Experiments and Baselines



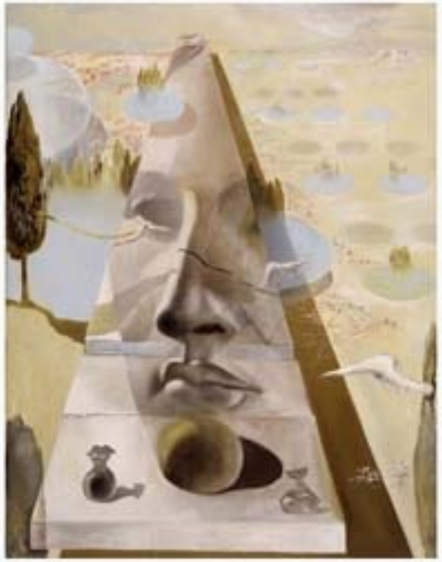
(a) Visual input (image and difference image) at time steps 100, 200, and 300 (indicated in (b) by black circles)

(b) Trajectory 9; starts at (0,0)

Fig. 10: **Visual odometry with DPFs.** Example test trajectory

CS231A

Computer Vision:
From 3D Reconstruction to Recognition



Next lecture:

Neural Radiance Fields for Novel View Synthesis