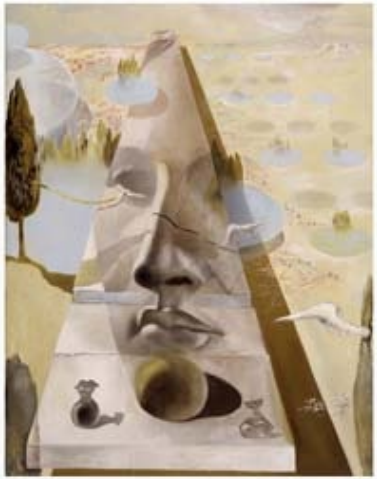
CS231A

Computer Vision:

From 3D Reconstruction to Recognition



Optimal Estimation Cont’

Recap

- Recursive Filter
- Kalman Filter → linear
- Extended Kalman Filter → non-linear

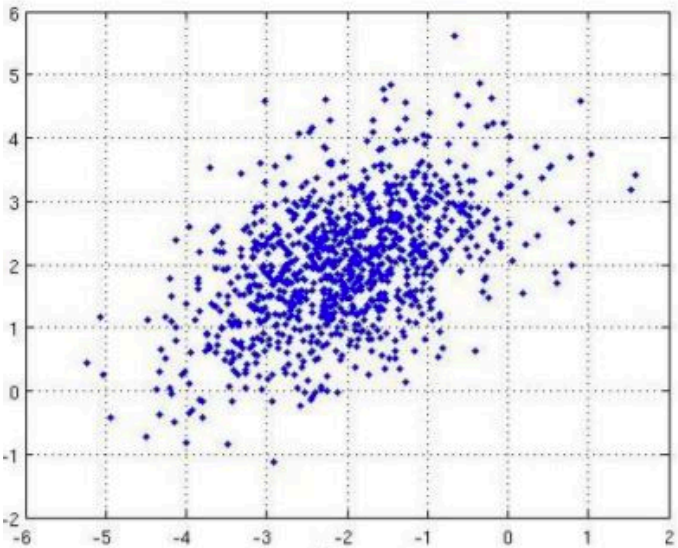
} assuming Gaussian distribution

Nonparametric filters

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

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

Particle Filter



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

N particles, each representing a hypothesis of what the actual state might be

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

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

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

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 $\mathcal{X}_t$

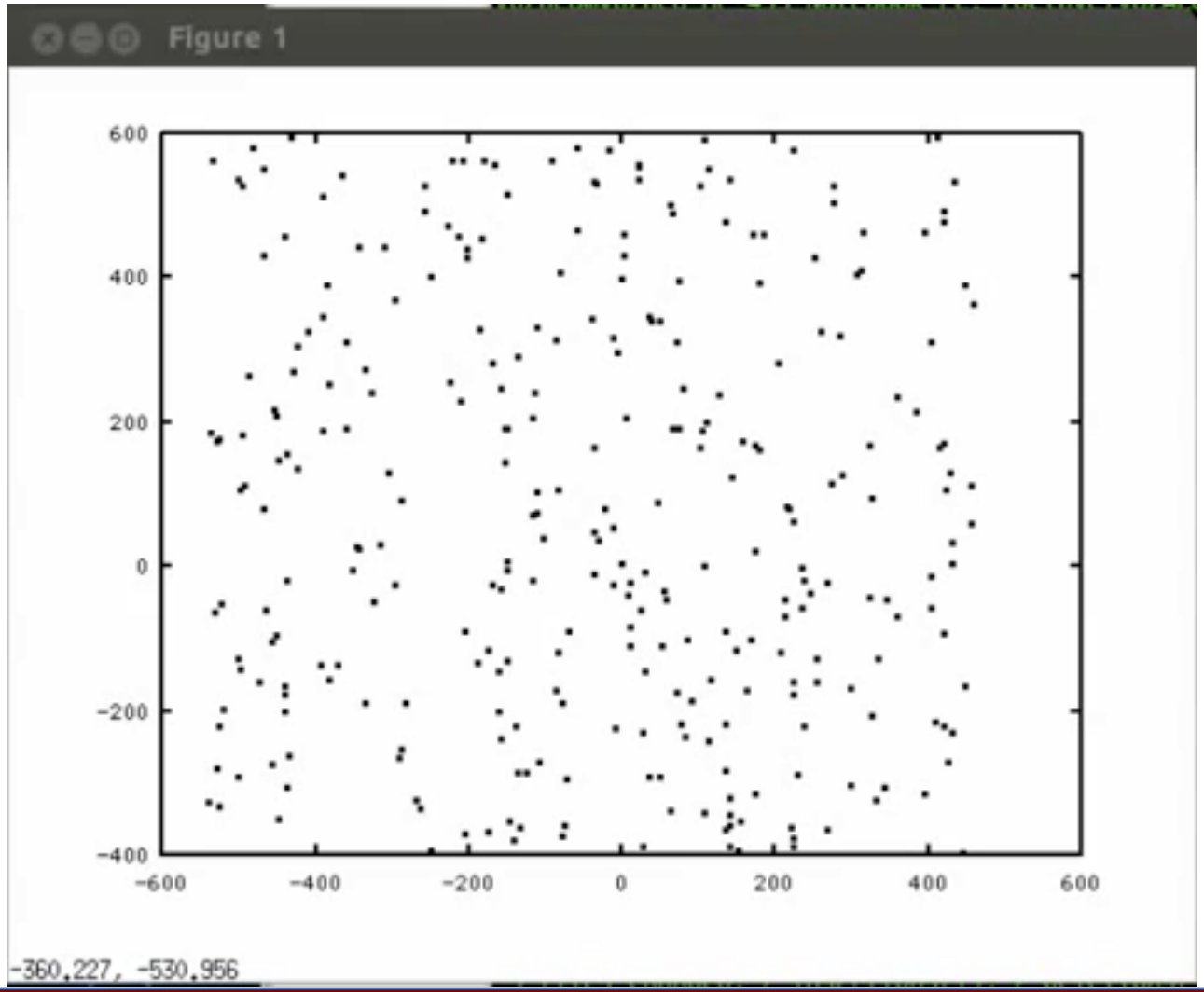


Figure 1
600
400
200
0
-200
-400
-600
-400
-200
0
200
400
600
-360.227, -530.956

When to Use Each?

Bayes Filter

General Framework
No implementation!

Kalman Filter

Linear Models
Gaussian Distributions

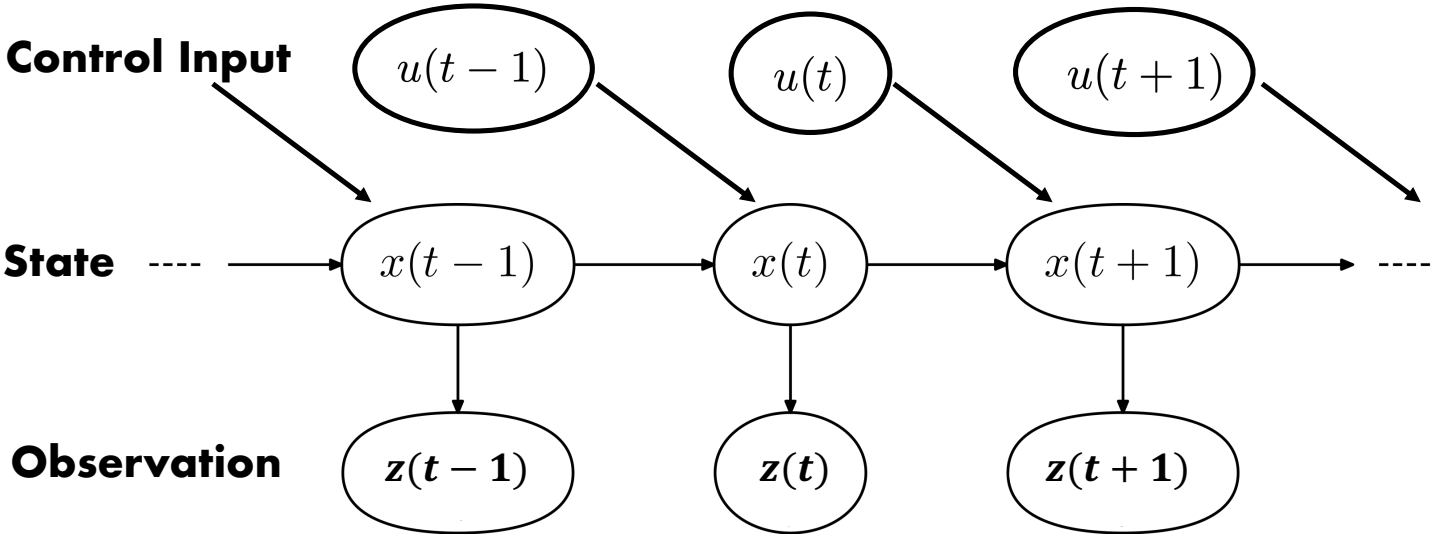
Extended Kalman Filter

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

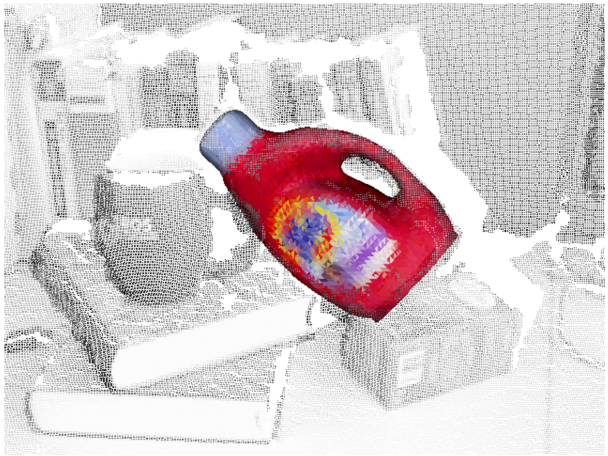
$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx$

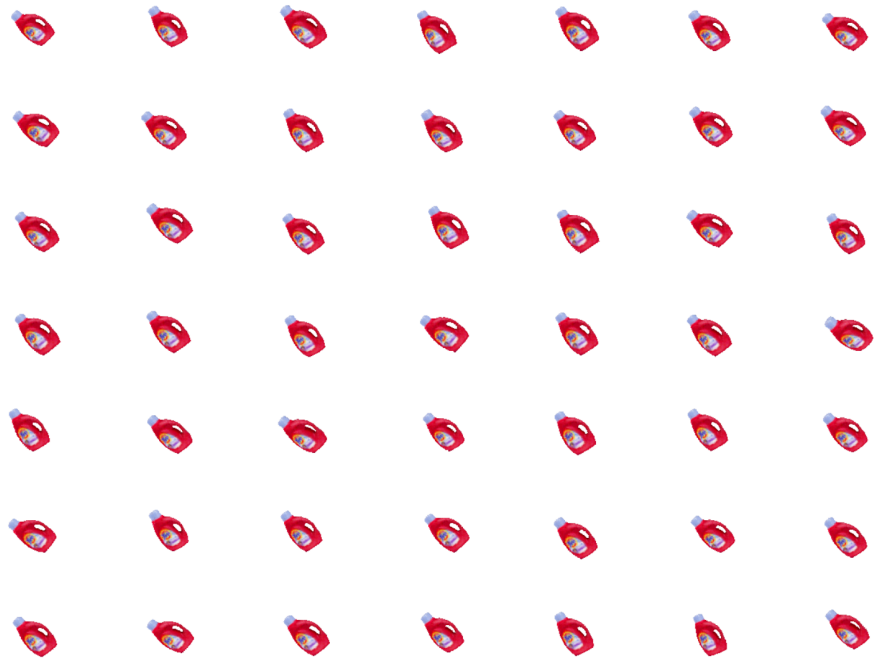
$bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$

endfor

return $bel(x_t)$

Example Observation model for 3D object





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 $\propto w_t^{[i]}$

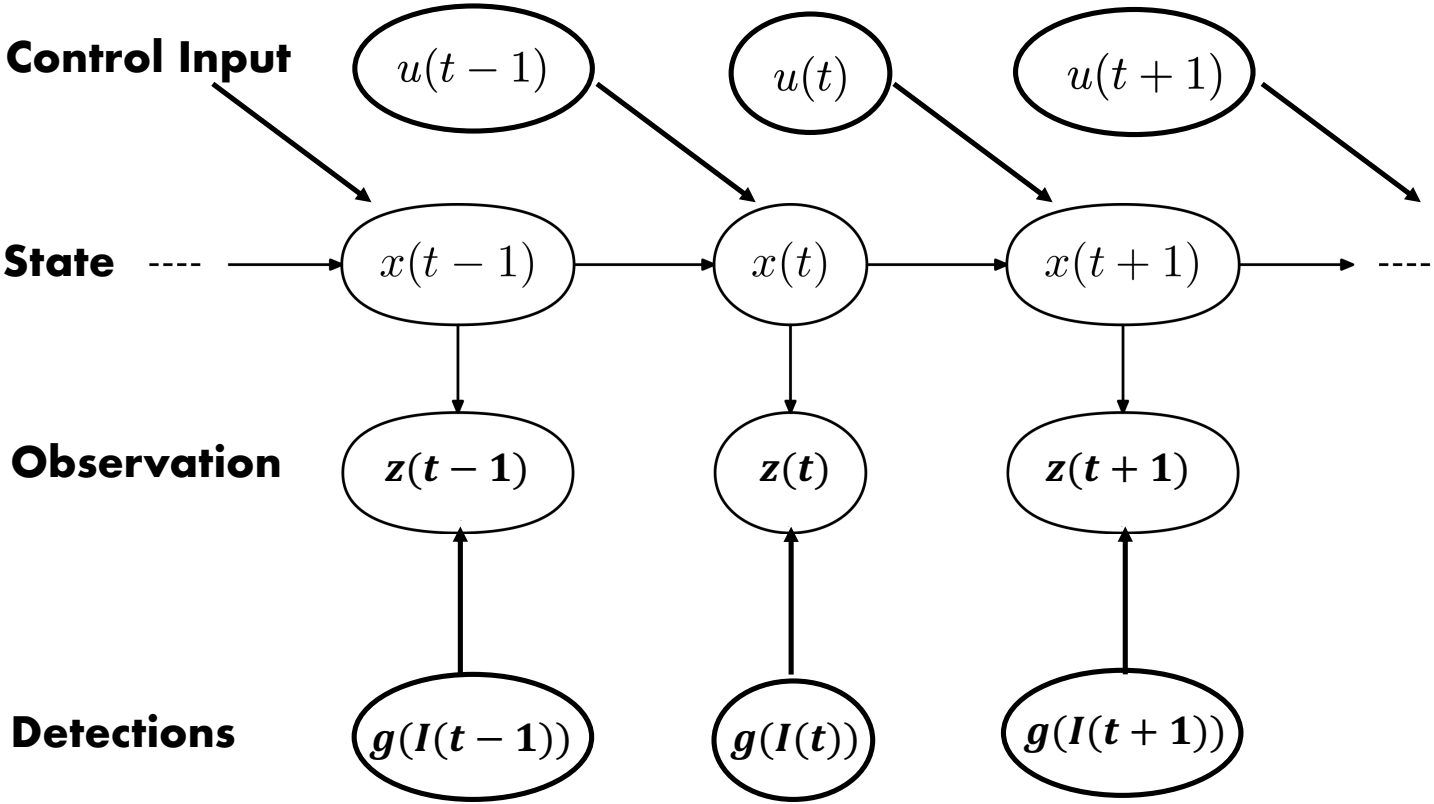
  add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

Changhyun Choi and Henrik I. Christensen. Rgb-d object tracking: A particlefilter approach on gpu. InIROS, pages 1084–1091, 2013

Tracking by Detection

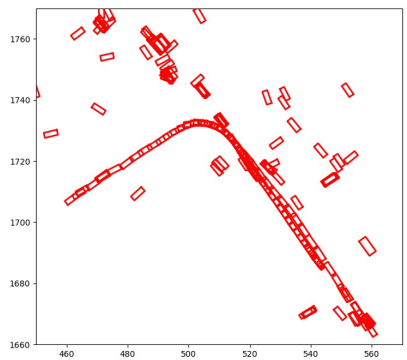


Object detector, $g(I(t))$ Sensory data

Problem Statement: Input

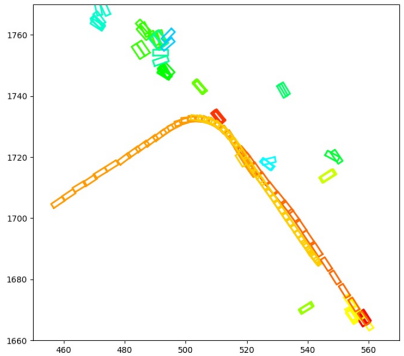
Probabilistic 3d multi-object tracking for autonomous driving. H Chiu, A Prioletti, J Li, J Bohg
arXiv preprint arXiv:2001.05673

- 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

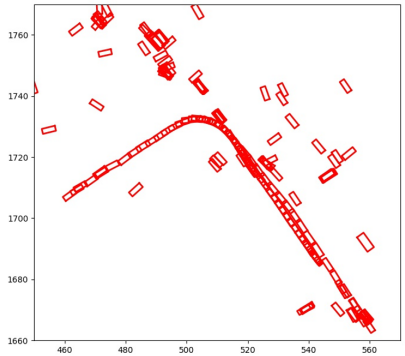
- Tracking object bounding boxes at each frame in a sequence
- Each tracking bounding box is represented by:
  - center position (x, y, z), rotation angle along the z-axis (a), and the scale (l, w, h)
  - category label (car, pedestrian, ...), confidence score (c)
  - tracking id: one unique tracking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



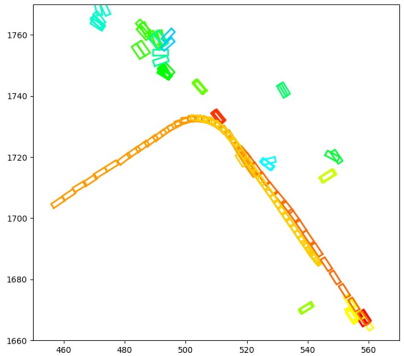
Why Tracking?

- 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

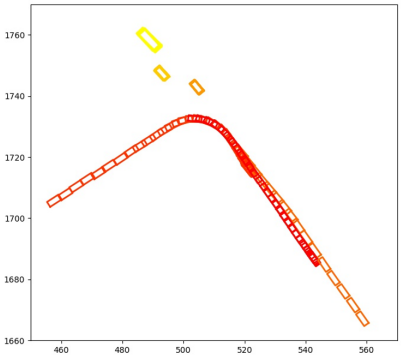
Detection

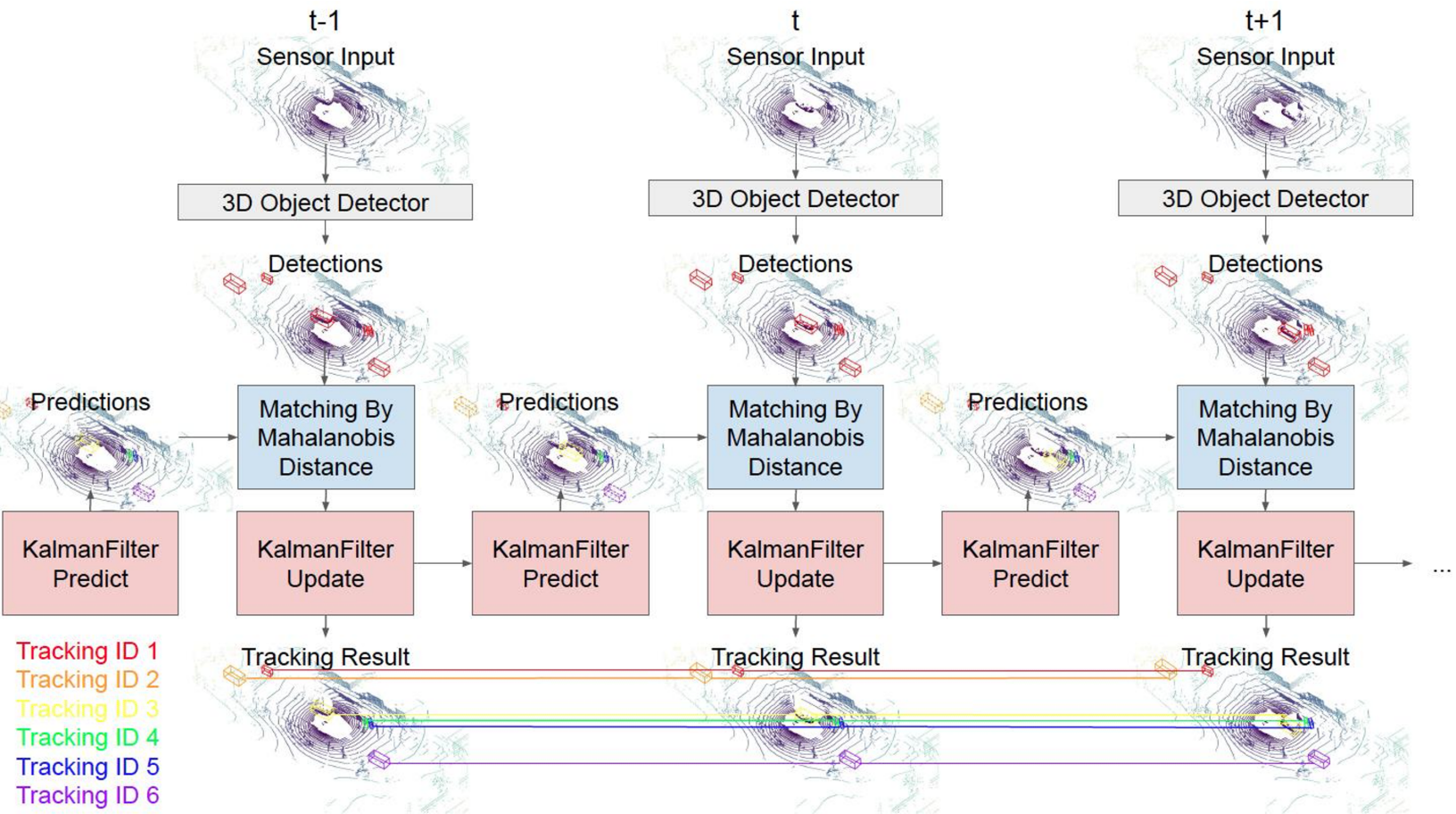


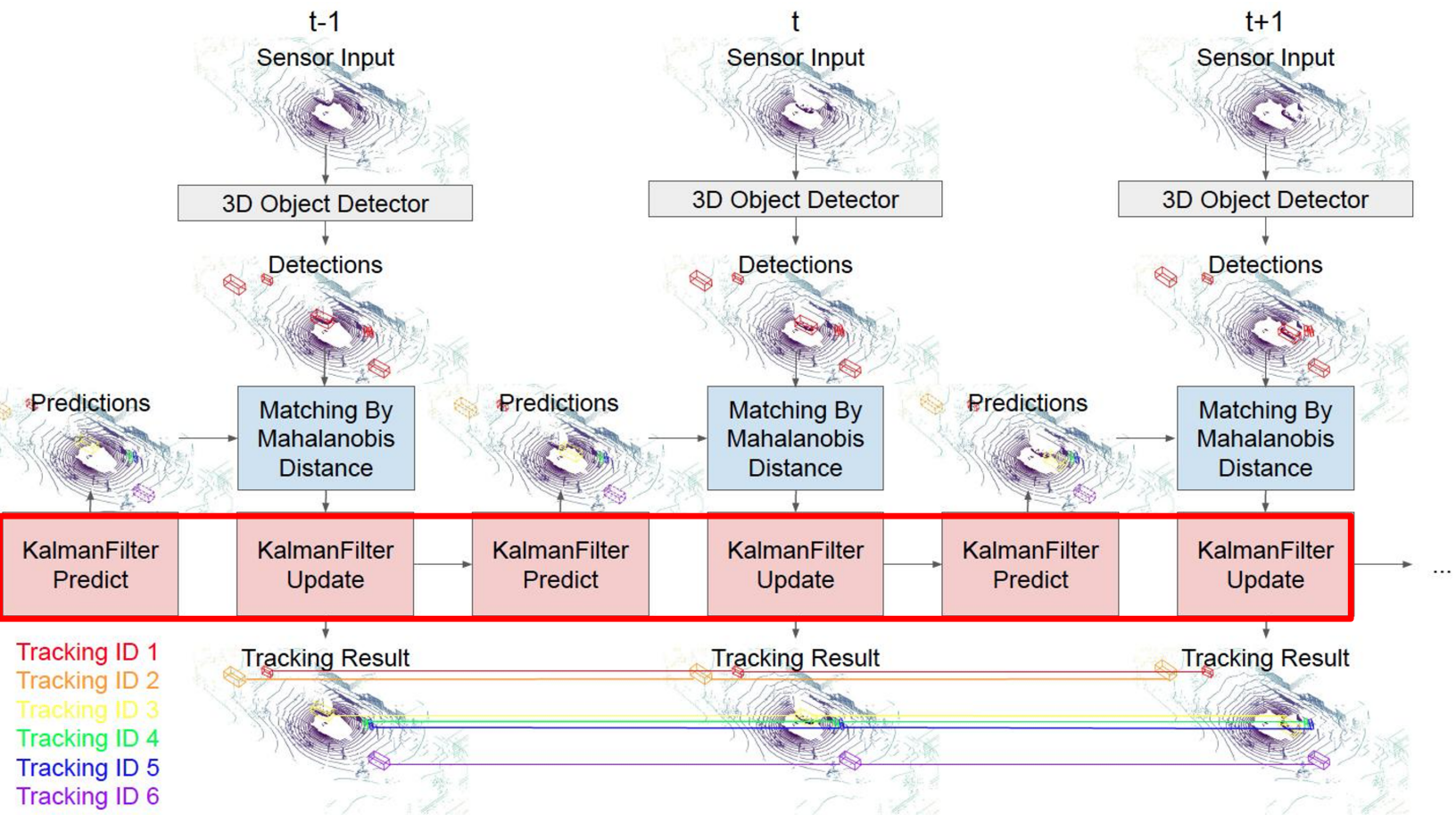
Tracking



Ground-truth




t-1
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


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

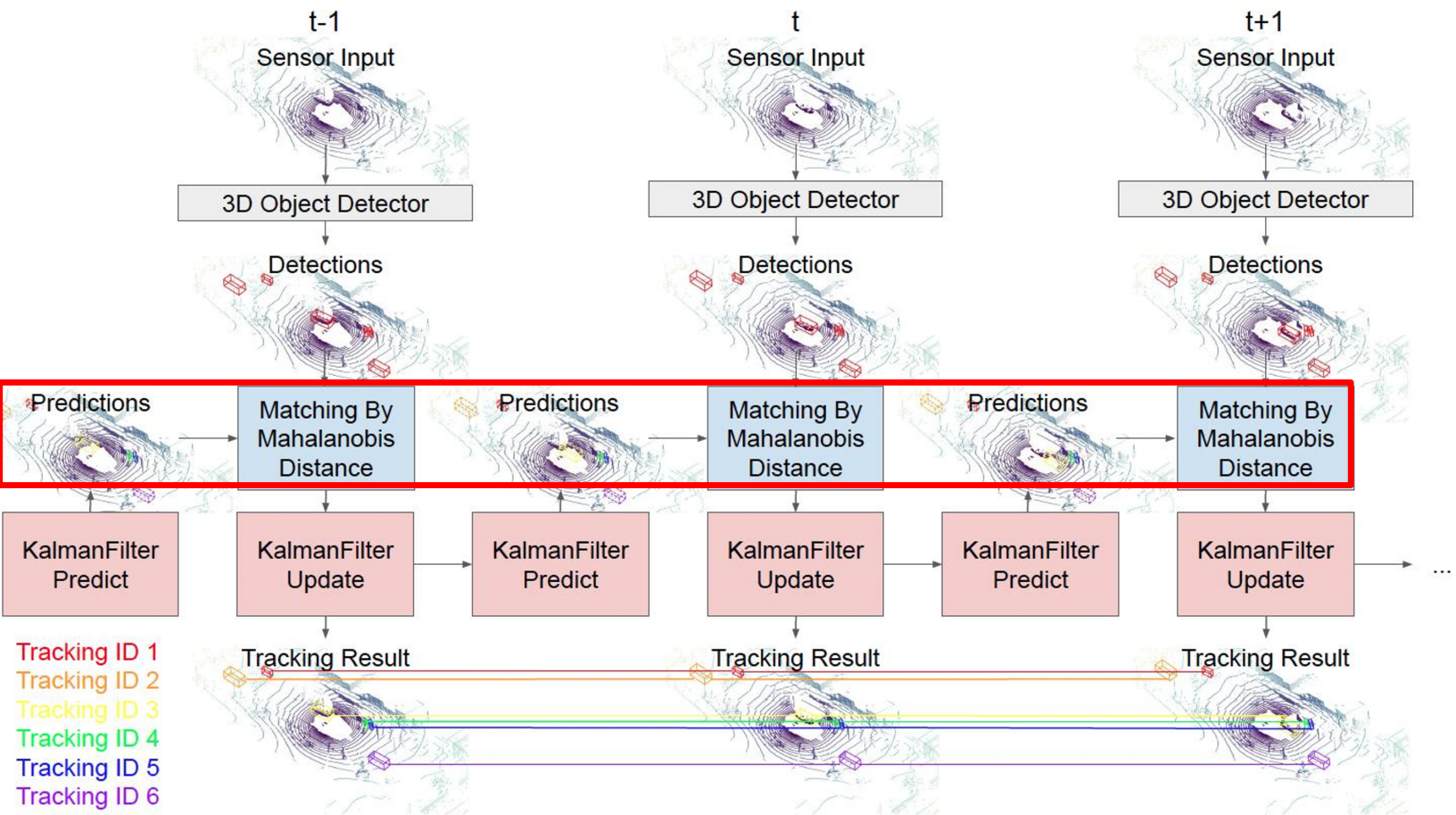
Kalman Filter for Tracking

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

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

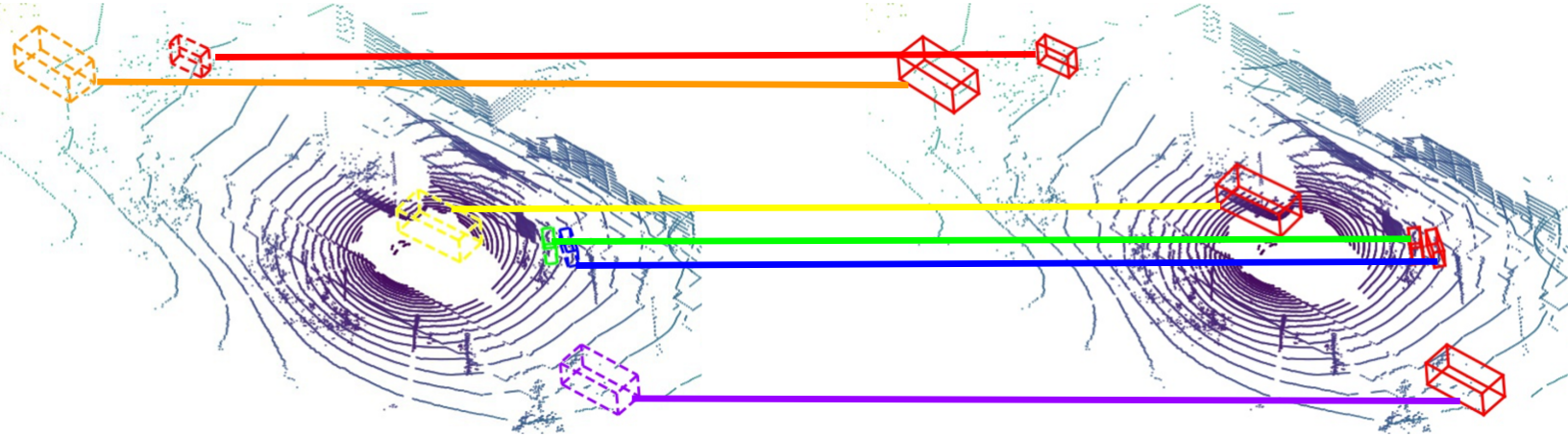
Define the **Process Model** for prediction based on the constant velocity motion:

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


t-1
Sensor Input
3D Object Detector
Detections
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

Data Association

Mahalanobis Distance $$m = \sqrt{(\mathbf{o}_{t+1} - \mathbf{H}\hat{\mu}_{t+1})^T \mathbf{S}_{t+1}{}^{-1} (\mathbf{o}_{t+1} - \mathbf{H}\hat{\mu}_{t+1})}$$



Predictions **Detections**

Kalman Filter

$$\hat{\mu}_{t+1} = \mathbf{A}\mu_t$$

$$\hat{\Sigma}_{t+1} = \mathbf{A}\Sigma_t\mathbf{A}^T + \mathbf{Q}$$

$$\hat{\mathbf{o}}_{t+1} = \mathbf{H}\hat{\mu}_{t+1}$$

$$\boxed{\mathbf{S}_{t+1} = \mathbf{H}\hat{\Sigma}_{t+1}\mathbf{H}^T + \mathbf{R}}$$

$$\mathbf{K}_{t+1} = \hat{\Sigma}_{t+1}\mathbf{H}^T\mathbf{S}_{t+1}^{-1}$$

$$\mu_{t+1} = \hat{\mu}_{t+1} + \mathbf{K}_{t+1}(\mathbf{o}_{t+1} - \mathbf{H}\hat{\mu}_{t+1})$$

$$\Sigma_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})\hat{\Sigma}_{t+1}$$

Data Association

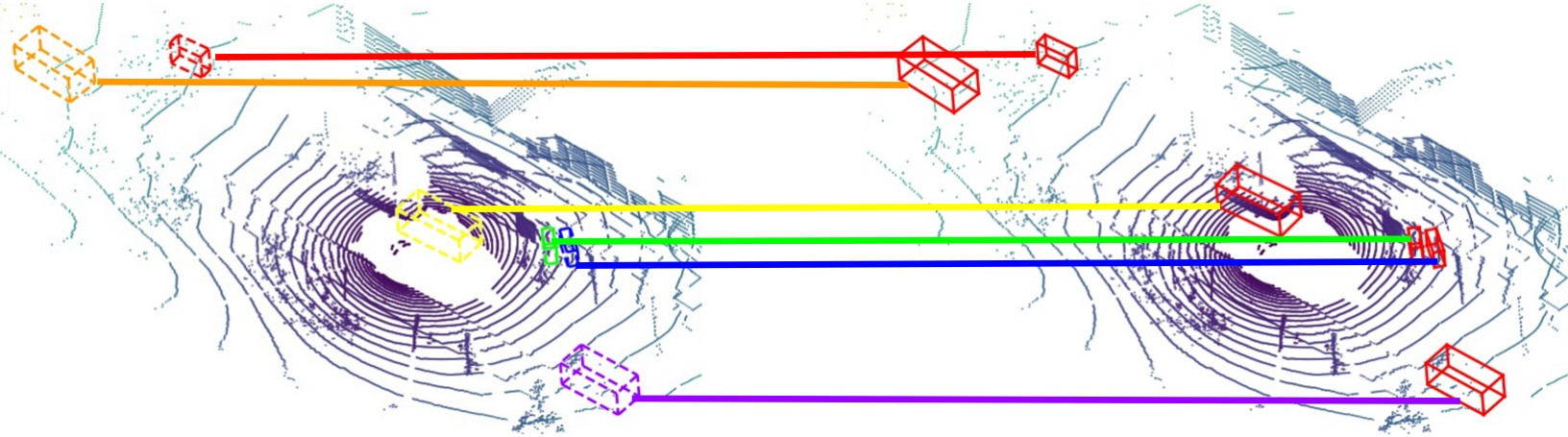
Mahalanobis Distance $$m = \sqrt{(\mathbf{o}_{t+1} - \mathbf{H}\hat{\mu}_{t+1})^T \mathbf{S}_{t+1}{}^{-1} (\mathbf{o}_{t+1} - \mathbf{H}\hat{\mu}_{t+1})}$$

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

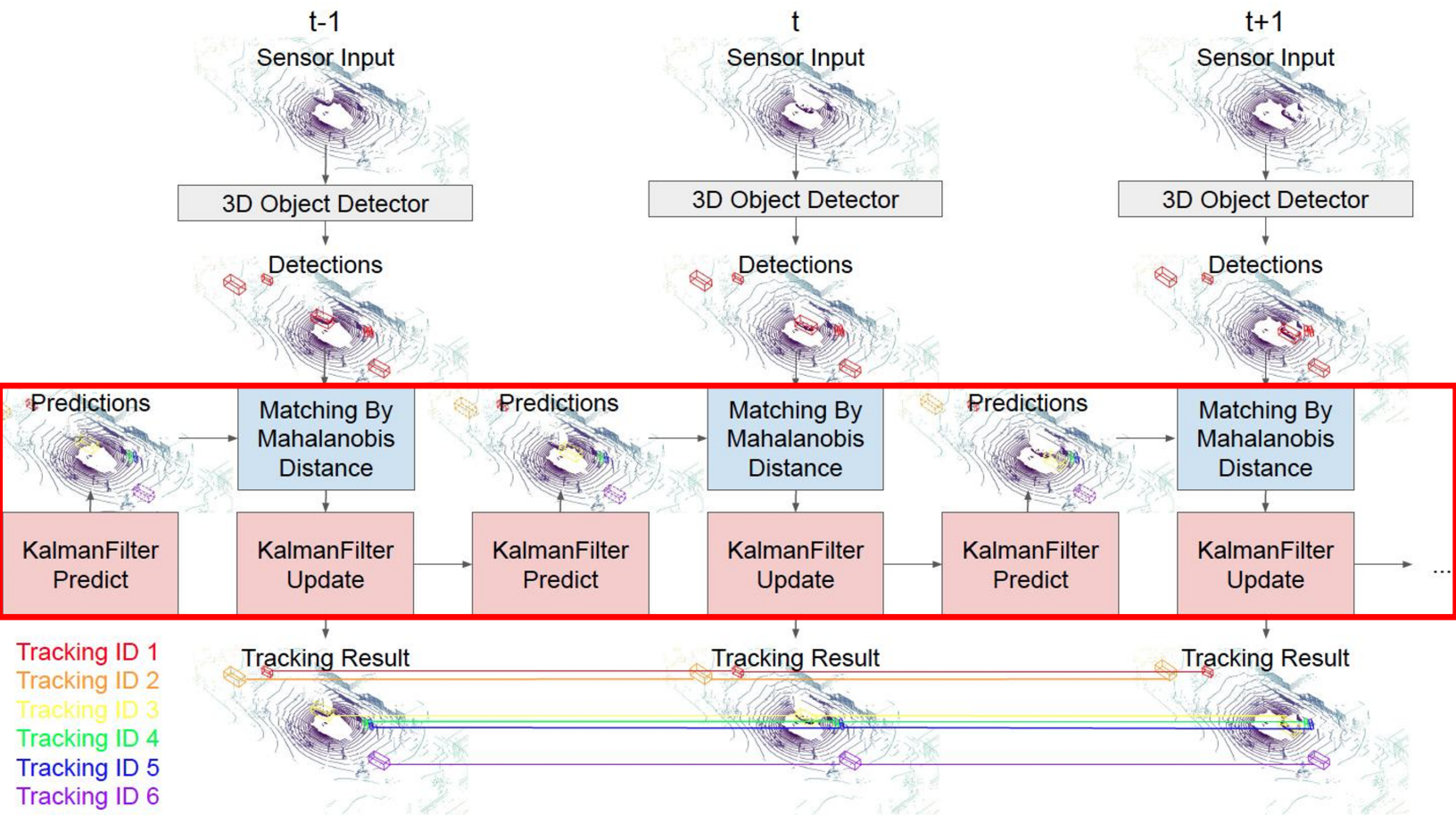
Providing distance measurement when there is no overlapping between the prediction and detection.

Taking the uncertainty information from the prediction into account.

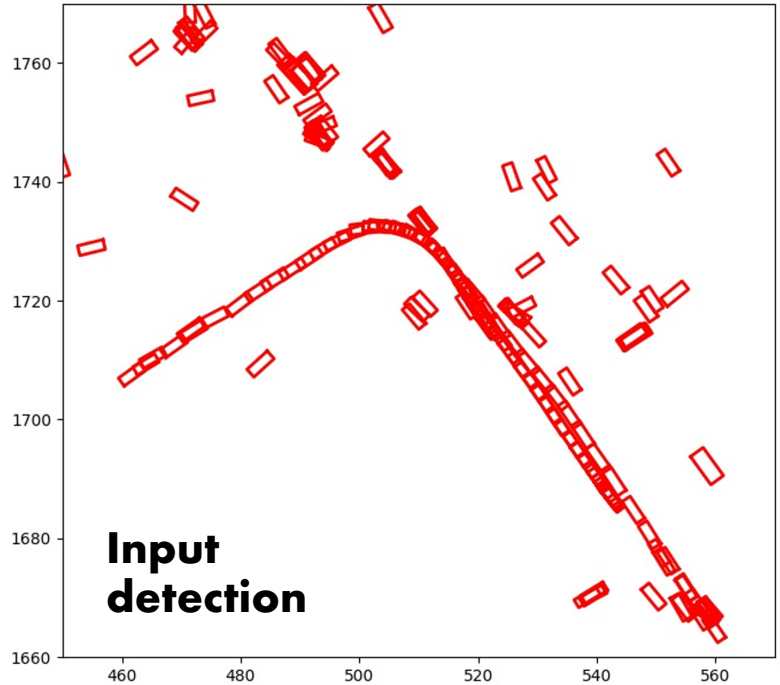
Data Association - Greedy

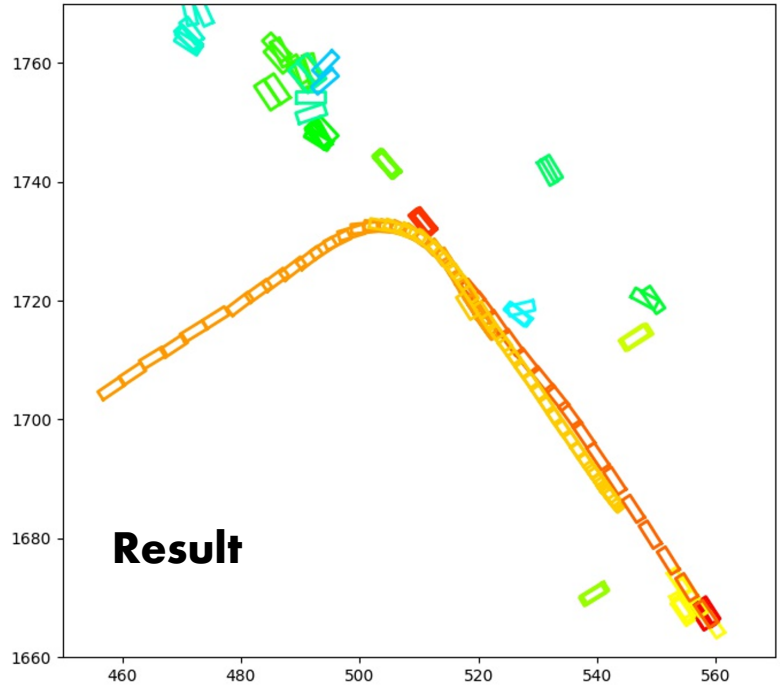


Predictions **Detections**

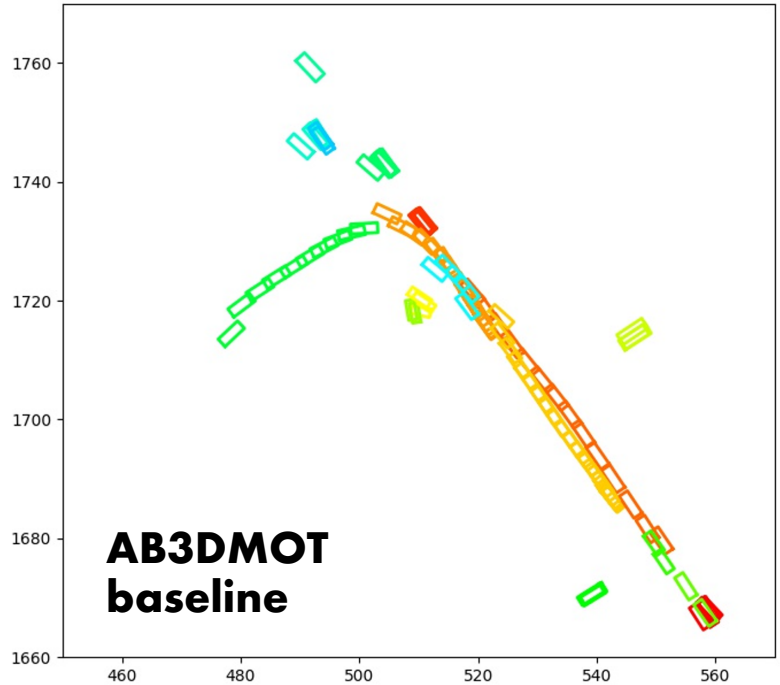

t-1
Sensor Input
3D Object Detector
Detections
Predictions
Matching By Mahalanobis Distance
KalmanFilter Predict
KalmanFilter Update
Tracking Result
t
Sensor Input
3D Object Detector
Detections
Predictions
Matching By Mahalanobis Distance
KalmanFilter Predict
KalmanFilter Update
Tracking Result
t+1
Sensor 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

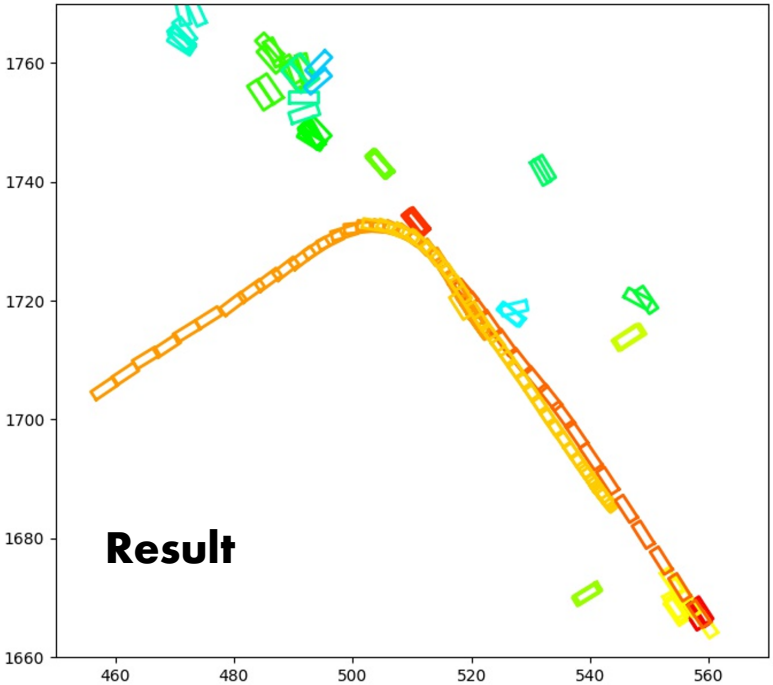
Qualitative Results



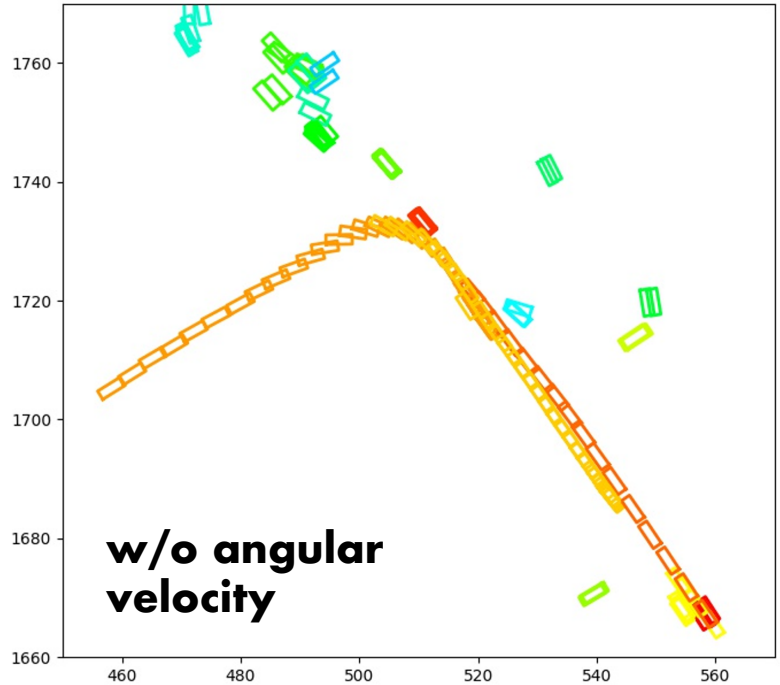


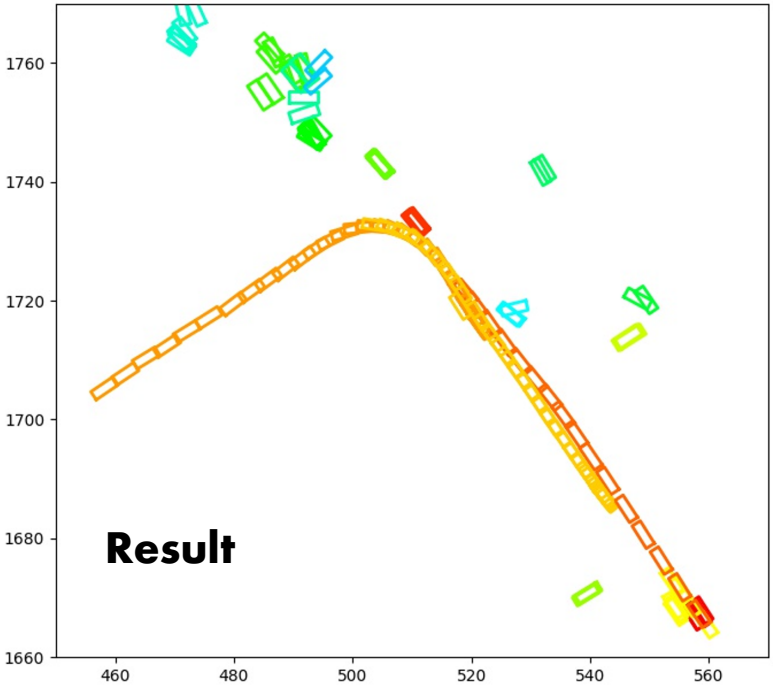
Qualitative Results



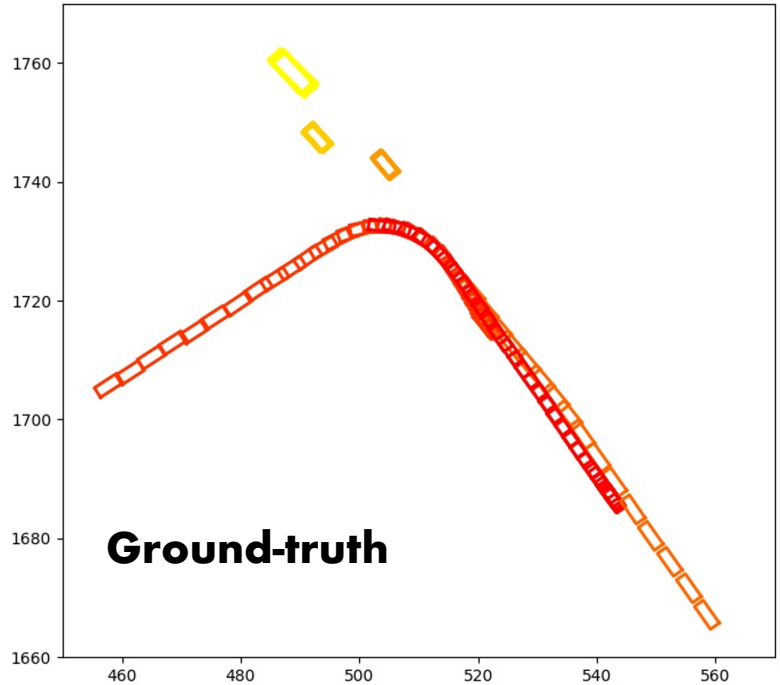


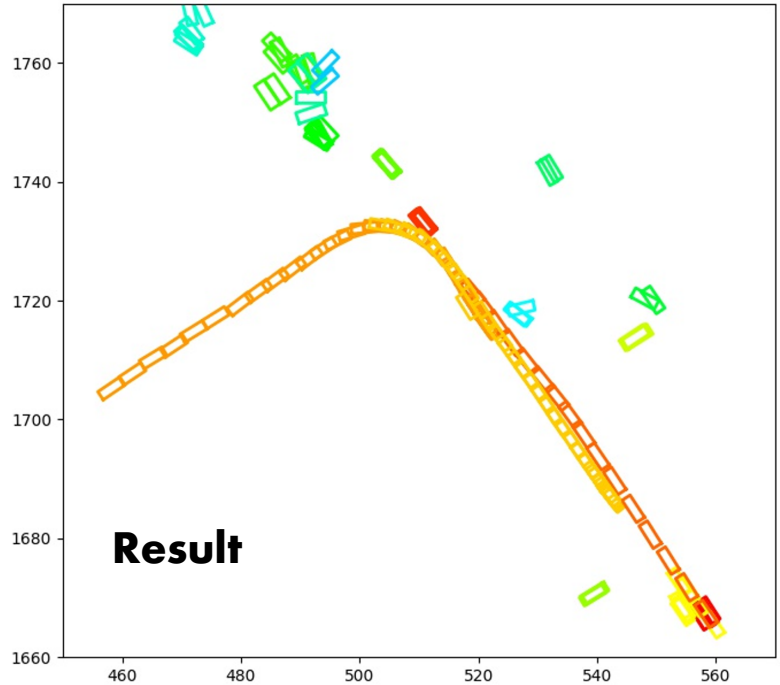
Qualitative Results





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

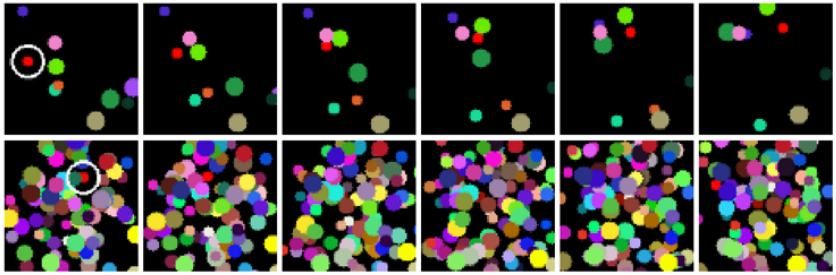
Can we learn models and hyperparameters from data?

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

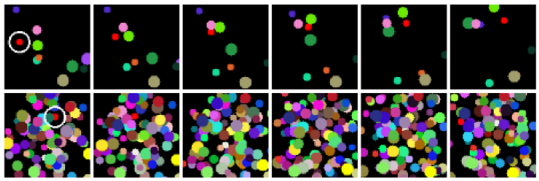
- prevents overfitting through regularization
- Avoids manual tuning and modeling

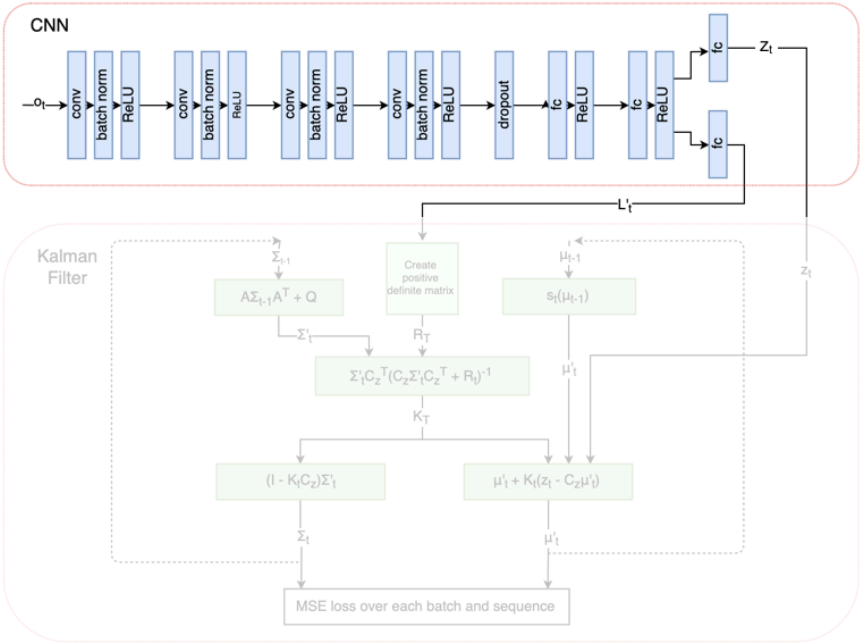
Estimators. Haarnoja et al. NeurIPS 2016

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

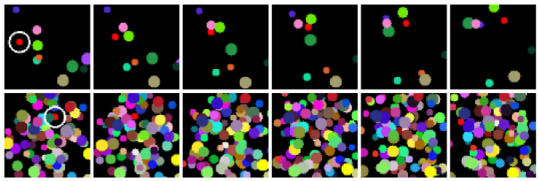


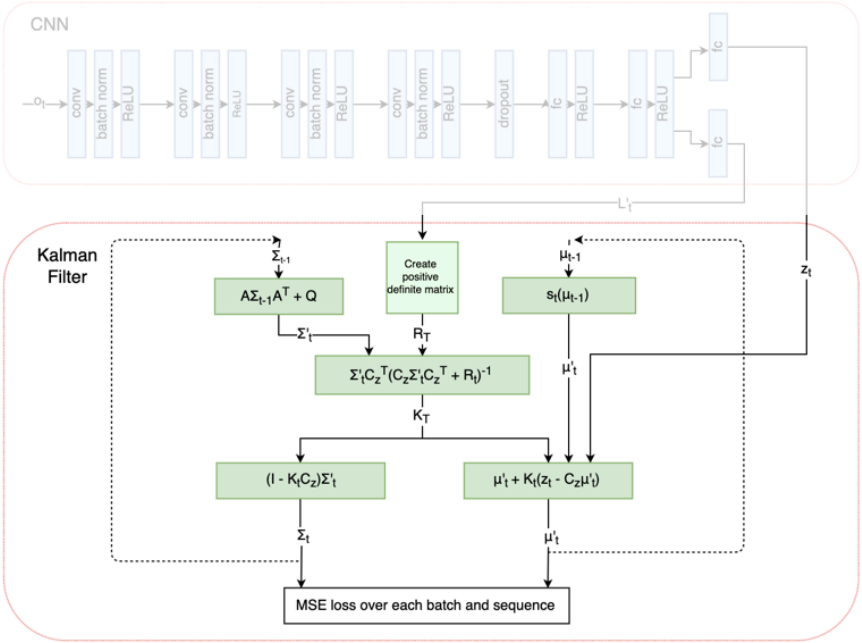
Differentiable Kalman Filter - Structure





Differentiable Kalman Filter - Structure





Differentiable Kalman Filter – Loss Function

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

$$\lambda_1 \sum_{t=0}^{T} \frac{1}{2}((\mathbf{l}_t - \mu_t)^T \Sigma_t^{-1}(l_t - \mu_t) + \log(|\Sigma_t|)) + \lambda_2 \sum_{t=0}^{T} \parallel (l_t - \mu_t) \parallel_2 + \lambda_3 \parallel \mathbf{w} \parallel_2$$

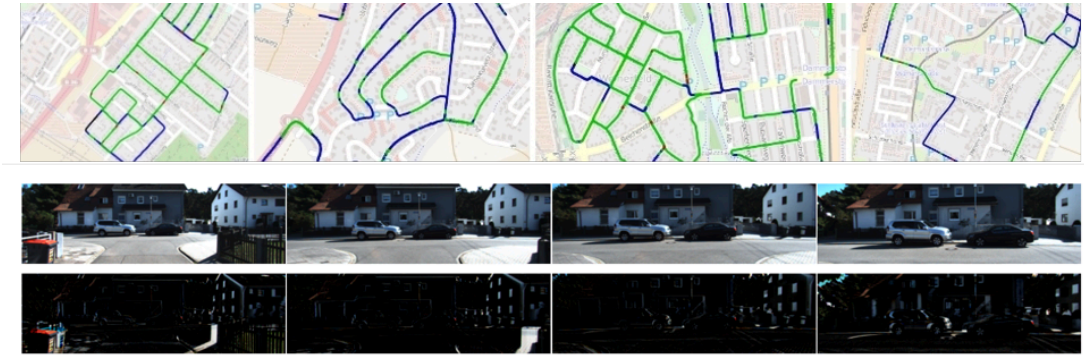
Differentiable Kalman Filter – Experiments and Baselines

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**	$\mathbf{0.0537 \pm 0.1235}$

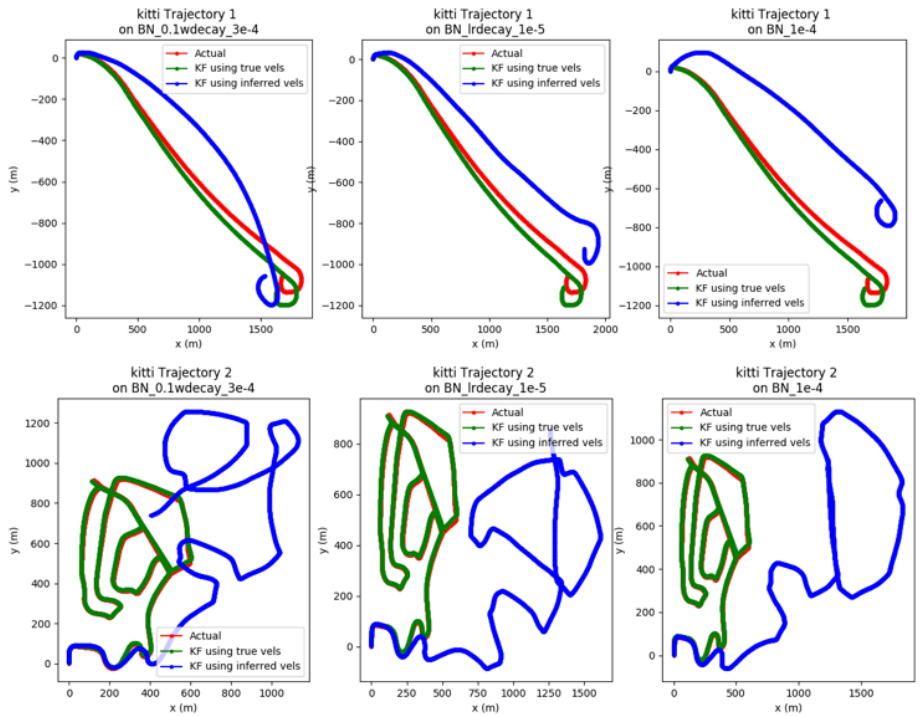
Differentiable Kalman Filter – Experiments and Baselines

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



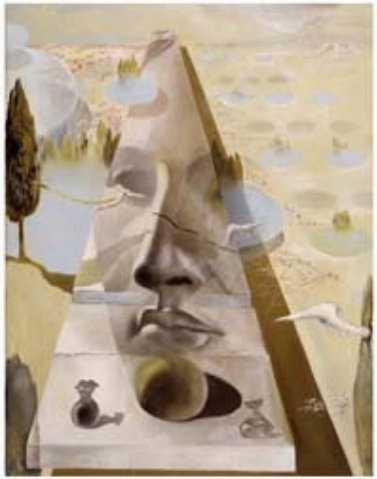
Differentiable Kalman Filter – Experiments and Baselines

Results reproduced by Claire Chen



CS231

Introduction to Computer Vision



Next lecture:

Optimal Smoothing