Computer Vision 2 — Exercise 2

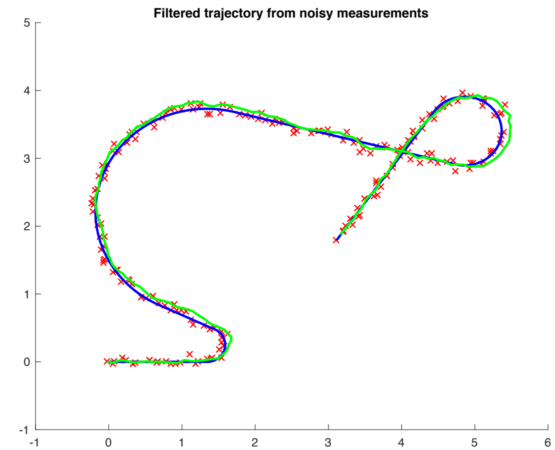
Extended Kalman Filter & Particle Filter

M.Sc. Francis Engelmann, Dr. Jörg Stückler
engelmann@vision.rwth-aachen.de, stueckler@vision.rwth-aachen.de

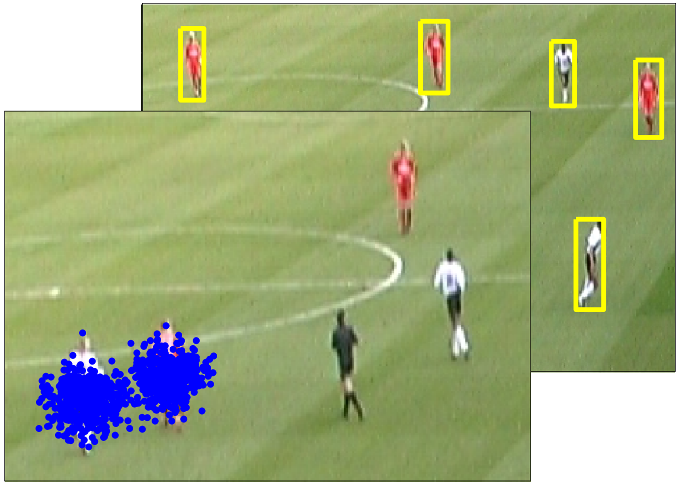
RWTH Aachen University, Computer Vision Group
http://www.vision.rwth-aachen.de

Content Exercise 2

- **Question 1: Extended Kalman Filter**
  - Compared to basic KF
  - Unicycle motion model
  - Nonlinearities & Jacobians



- **Question 2: Particle Filter**
  - Compared to general KF
  - SIR Algorithm
  - Implementation Details



Question 1: Extended Kalman Filter

Q1: EKF - Compared to basic KF

- Kalman Filter
  - Assumption: linear model

$$\mathbf{x}_t = \mathbf{D}_t \mathbf{x}_{t-1} + \varepsilon_t$$
$$\mathbf{y}_t = \mathbf{M}_t \mathbf{x}_t + \delta_t$$

  - Prediction step

$$\mathbf{x}_t^- = \mathbf{D}_t \mathbf{x}_{t-1}^+$$
$$\mathbf{\Sigma}_t^- = \mathbf{D}_t \mathbf{\Sigma}_{t-1}^+ \mathbf{D}_t^T + \mathbf{\Sigma}_{d_t}$$

  - Correction step

$$\mathbf{K}_t = \mathbf{\Sigma}_t^- \mathbf{M}_t^T \left(\mathbf{M}_t \mathbf{\Sigma}_t^- \mathbf{M}_t^T + \mathbf{\Sigma}_{m_t}\right)^{-1}$$
$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - \mathbf{M}_t \mathbf{x}_t^-\right)$$
$$\mathbf{\Sigma}_t^+ = \left(\mathbf{I} - \mathbf{K}_t \mathbf{M}_t\right) \mathbf{\Sigma}_t^-$$

- Extended Kalman Filter
  - Nonlinear model

$$\mathbf{x}_t = \mathbf{g}\left(\mathbf{x}_{t-1}\right) + \varepsilon_t$$
$$\mathbf{y}_t = \mathbf{h}\left(\mathbf{x}_t\right) + \delta_t$$

$$\mathbf{G}_t = \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}$$
$$\mathbf{H}_t = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}$$

  - Prediction step

$$\mathbf{x}_t^- = \mathbf{g}\left(\mathbf{x}_{t-1}^+\right)$$
$$\mathbf{\Sigma}_t^- = \mathbf{G}_t \mathbf{\Sigma}_{t-1}^+ \mathbf{G}_t^T + \mathbf{\Sigma}_{d_t}$$

  - Correction step

$$\mathbf{K}_t = \mathbf{\Sigma}_t^- \mathbf{H}_t^T \left(\mathbf{H}_t \mathbf{\Sigma}_t^- \mathbf{H}_t^T + \mathbf{\Sigma}_{m_t}\right)^{-1}$$
$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - \mathbf{h}\left(\mathbf{x}_t^-\right)\right)$$
$$\mathbf{\Sigma}_t^+ = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right) \mathbf{\Sigma}_t^-$$

Q1: EKF - Unicycle Motion Model

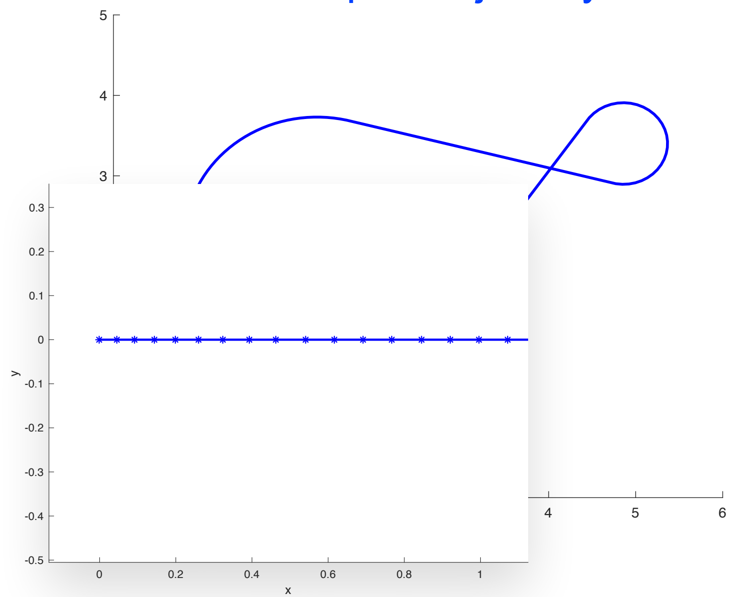
- The ”unicycle” motion model is an approximation often used for bicycle or car motions.

- State vector:

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ v_t \end{bmatrix}$$

  - ➢ position x
  - ➢ position y
  - ➢ orientation angle
  - ➢ velocity

Example Trajectory



Q1: EKF

a) Point out, which steps contain nonlinearities and give the definition of the functions $\mathbf{g}$ and $\mathbf{h}$.

Dynamic Model

$$\mathbf{x}_t = \mathbf{g}(\mathbf{x}_{t-1}) + \epsilon_t$$

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ v_t \end{bmatrix} \qquad \mathbf{x}_{t+1} = \begin{bmatrix} x_t + \Delta t v_t \cos\theta_t & +\varepsilon_x \\ y_t + \Delta t v_t \sin\theta_t & +\varepsilon_y \\ \theta_t & +\varepsilon_\theta \\ v_t & +\varepsilon_v \end{bmatrix}$$

nonlinear functions

Measurement Model

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t) + \delta_t$$

$$\mathbf{y}_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix}$$

$$g : \mathbb{R}^4 \to \mathbb{R}^4, \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ v_t \end{bmatrix} \mapsto \begin{bmatrix} x_t + \Delta t v_t \cos\theta_t \\ y_t + \Delta t v_t \sin\theta_t \\ \theta_t \\ v_t \end{bmatrix}$$

$$h : \mathbb{R}^4 \to \mathbb{R}^2, \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ v_t \end{bmatrix} \mapsto \begin{bmatrix} x_t \\ y_t \end{bmatrix}$$

Q1: EKF b)

b) Compute the Jacobians $\mathbf{G}_t$ and $\mathbf{H}_t$ of $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ respectively.

$$g : \mathbb{R}^4 \to \mathbb{R}^4, \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ v_t \end{bmatrix} \mapsto \begin{bmatrix} x_t + \Delta t v_t \cos\theta_t \\ y_t + \Delta t v_t \sin\theta_t \\ \theta_t \\ v_t \end{bmatrix} \qquad h : \mathbb{R}^4 \to \mathbb{R}^2, \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ v_t \end{bmatrix} \mapsto \begin{bmatrix} x_t \\ y_t \end{bmatrix}$$

$$\mathbf{G}_t = \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \qquad \mathbf{H}_t = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}$$

$$\mathbf{G}_t = \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial g_1}{\partial x_t} & \frac{\partial g_1}{\partial y_t} & \frac{\partial g_1}{\partial \theta_t} & \frac{\partial g_1}{\partial v_t} \\ \frac{\partial g_2}{\partial x_t} & \frac{\partial g_2}{\partial y_t} & \frac{\partial g_2}{\partial \theta_t} & \frac{\partial g_2}{\partial v_t} \\ \frac{\partial g_3}{\partial x_t} & \frac{\partial g_3}{\partial y_t} & \frac{\partial g_3}{\partial \theta_t} & \frac{\partial g_3}{\partial v_t} \\ \frac{\partial g_4}{\partial x_t} & \frac{\partial g_4}{\partial y_t} & \frac{\partial g_4}{\partial \theta_t} & \frac{\partial g_4}{\partial v_t} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta t v_t \sin\theta_t & \Delta t \cos\theta_t \\ 0 & 1 & \Delta t v_t \cos\theta_t & \Delta t \sin\theta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

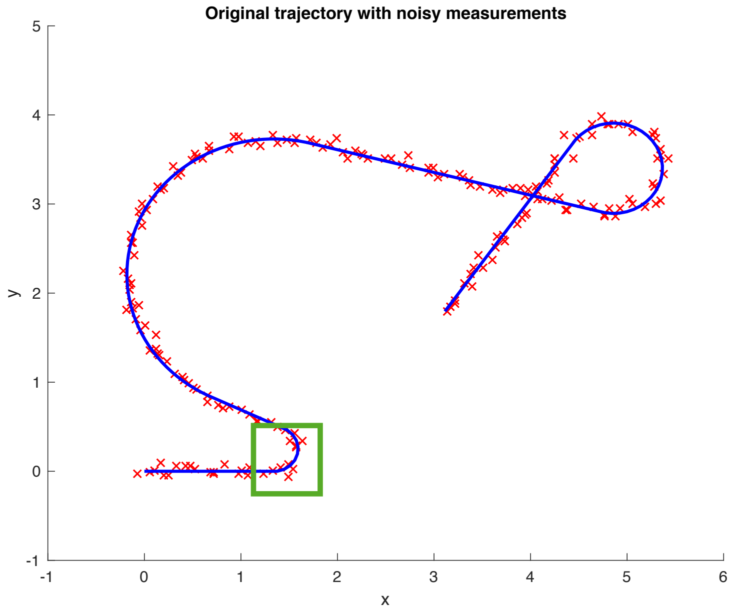
$$\mathbf{H}_t = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial x_t}{\partial x_t} & \frac{\partial x_t}{\partial y_t} & \frac{\partial x_t}{\partial \theta_t} & \frac{\partial x_t}{\partial v_t} \\ \frac{\partial y_t}{\partial x_t} & \frac{\partial y_t}{\partial y_t} & \frac{\partial y_t}{\partial \theta_t} & \frac{\partial y_t}{\partial v_t} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

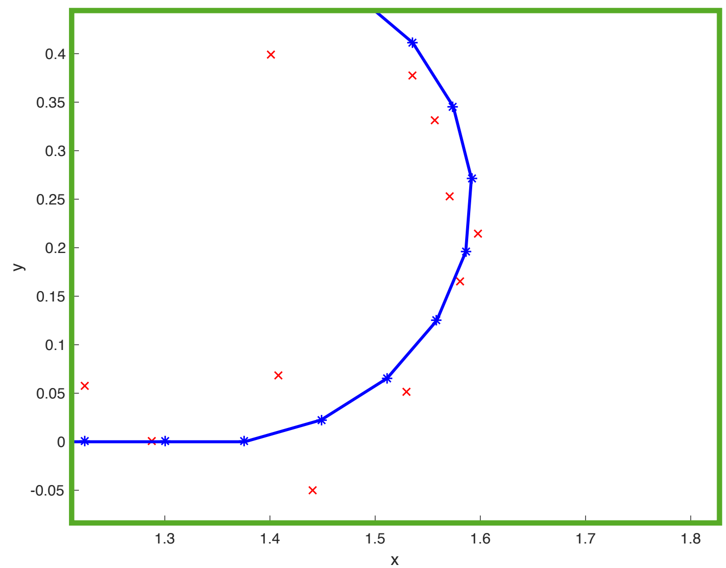
← same as in linear case

Q1: EKF c)

c) Implement extended Kalman filter.

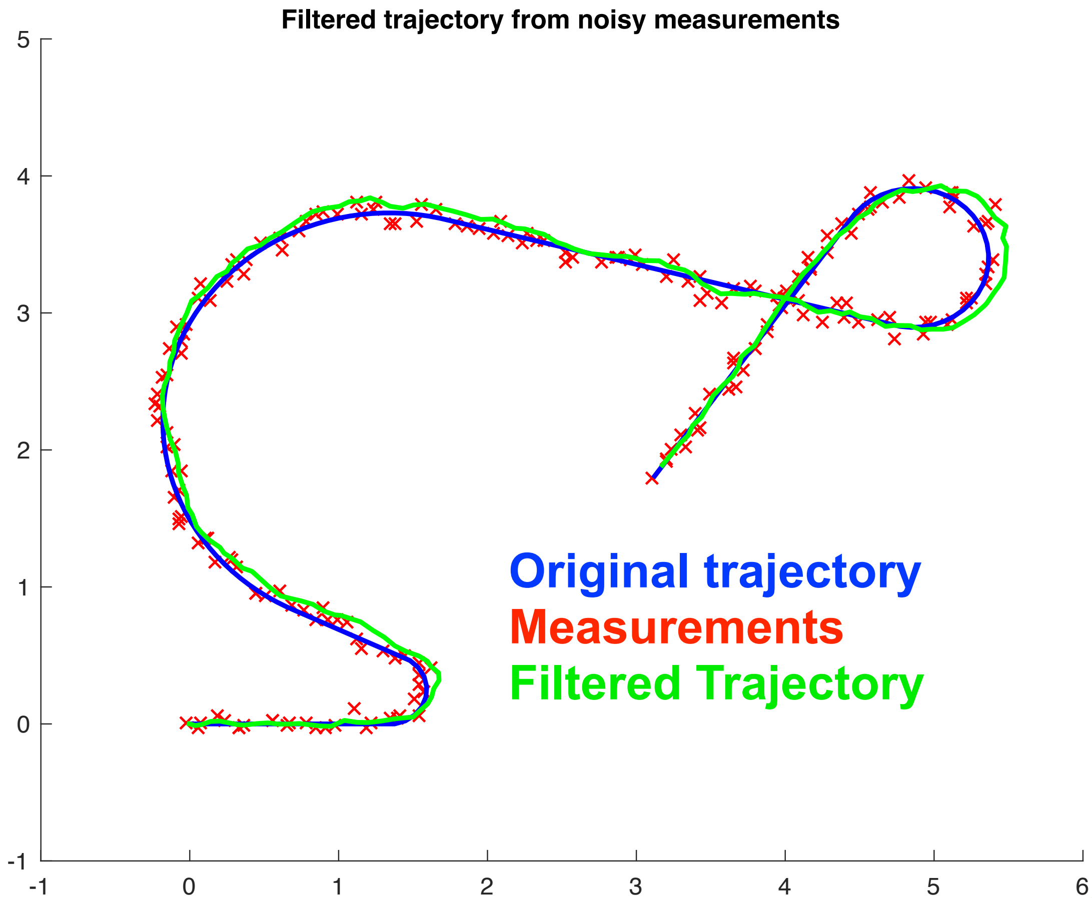
- Generate measurements by adding Gaussian noise to original state at each time step.
- Use EKF to estimate **original trajectory** based on **noisy measurements**.





Q1: EKF c)

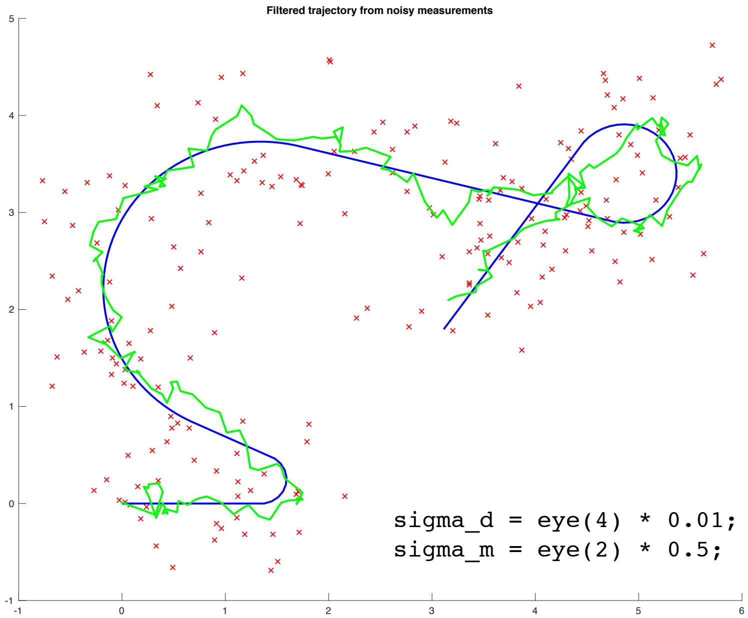
c) Implement extended Kalman filter.

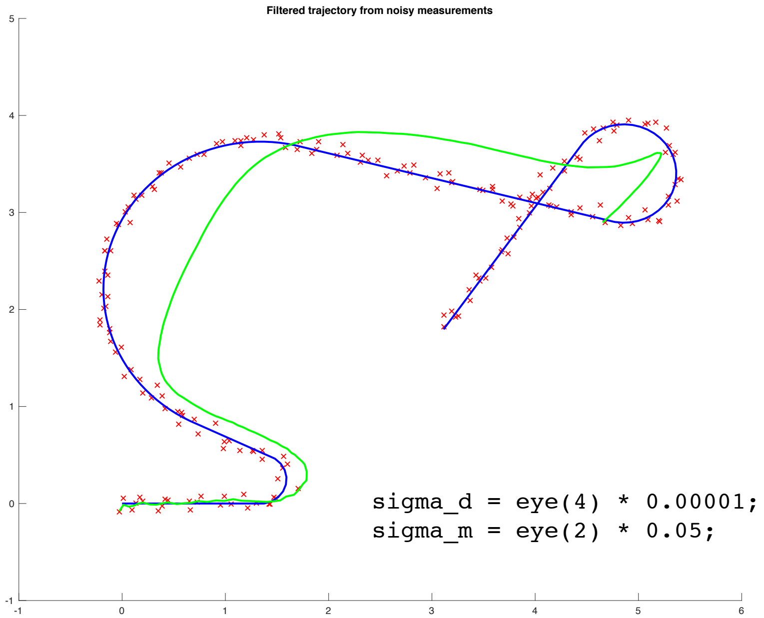


Q1: EKF c)

c) Implement extended Kalman filter.

- Influence of Σ_{d_t} and Σ_{m_t}



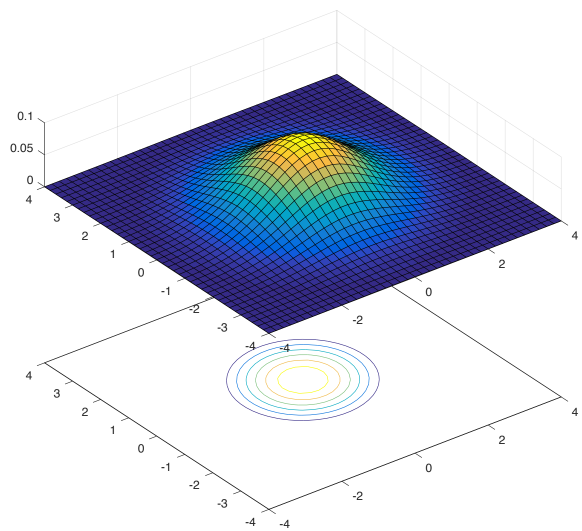


Question 2: Particle Filter

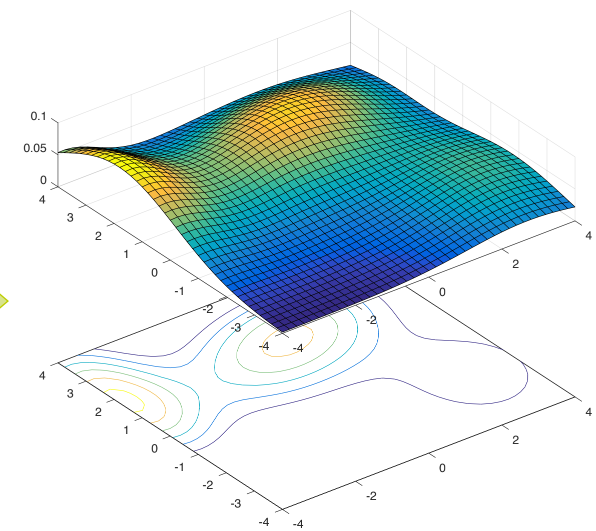
Q2: Particle Filter

- **What is different from particle filters to Kalman filters?**
  - Kalman Filter: all probability distributions are normal distributions.
  - Particle Filter: any distribution is possible.
  - In particular, this allows us to model multiple hypothesis for the state.

Normal Distribution



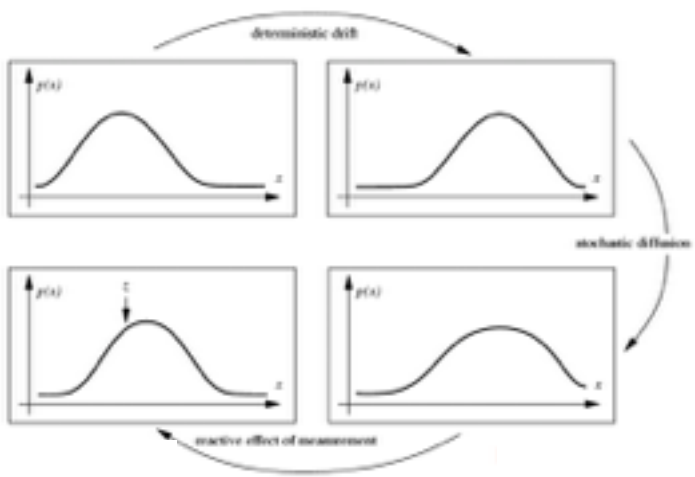
Arbitrary Distribution



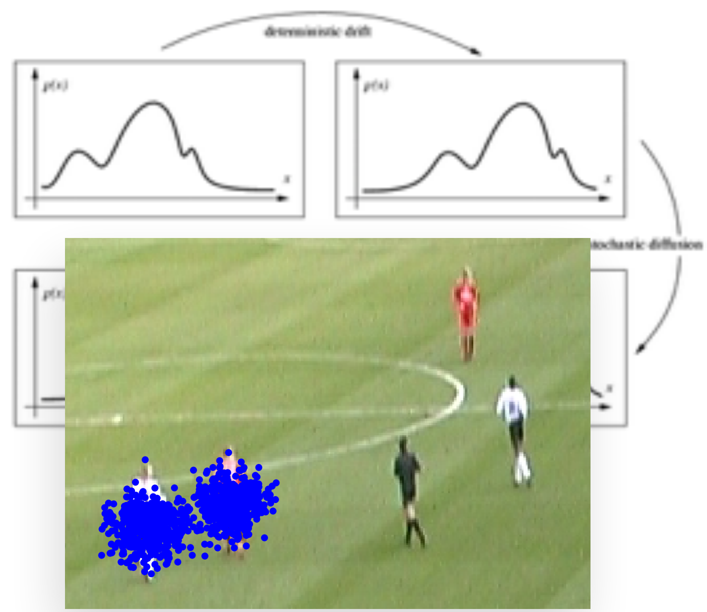
Q2: Particle Filter

- **What is different from particle filters to Kalman filters?**
  - In particular, this allows us to model multiple hypothesis for the state.

Kalman Filter: Single Mode



Particle Filter: Multiple Modes



Q2: Particle Filter

- **SIR** (Sampling Importance Resampling)

Set of particles (samples from posterior at time t)

Sampling

function $[\mathcal{X}_t] = SIR[\mathcal{X}_{t-1}, \mathbf{y}_t]$

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

for $i = 1:N$

$\quad$ *Sample* $\mathbf{x}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)$

$\quad$ $w_t^i = p(\mathbf{y}_t|\mathbf{x}_t^i)$

end

Re-sampling

for $i = 1:N$

$\quad$ *Draw i with probability* $\propto w_t^i$

$\quad$ *Add* $\mathbf{x}_t^i$ *to* $\mathcal{X}_t$

end

Number of particles

Particle

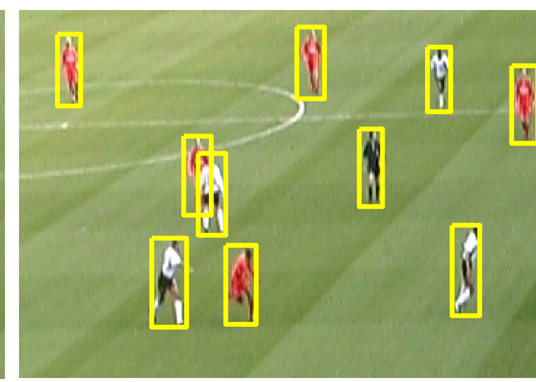
State transition distribution, according to dynamic model
here: constant position model

Weight of particle

Resampling

Particles $\mathbf{x}$

Detections $\mathbf{y}$



Q2: Particle Filter

- **SIR** (Sampling Importance Resampling)

function $[\mathcal{X}_t] = SIR\,[\mathcal{X}_{t-1}, \mathbf{y}_t]$

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

for $i = 1{:}N$

a) *Sample* $\mathbf{x}_t^i \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^i)$ — `generate_particles.m`

b) $w_t^i = p(\mathbf{y}_t | \mathbf{x}_t^i)$ — `compute_particle_likelihood.m`

end

for $i = 1{:}N$

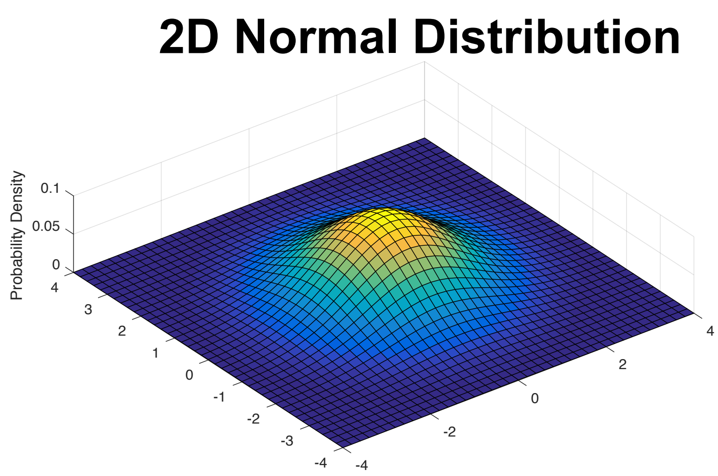
c) *Draw* i *with probability* $\propto w_t^i$ — `inverse_transform_sampling.m`

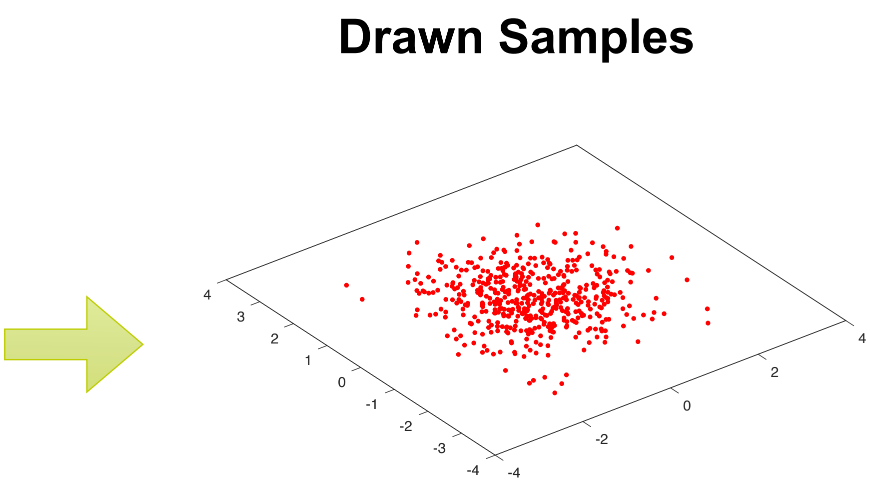
Add $\mathbf{x}_t^i$ *to* $\mathcal{X}_t$

end

Q2: Particle Filter a)

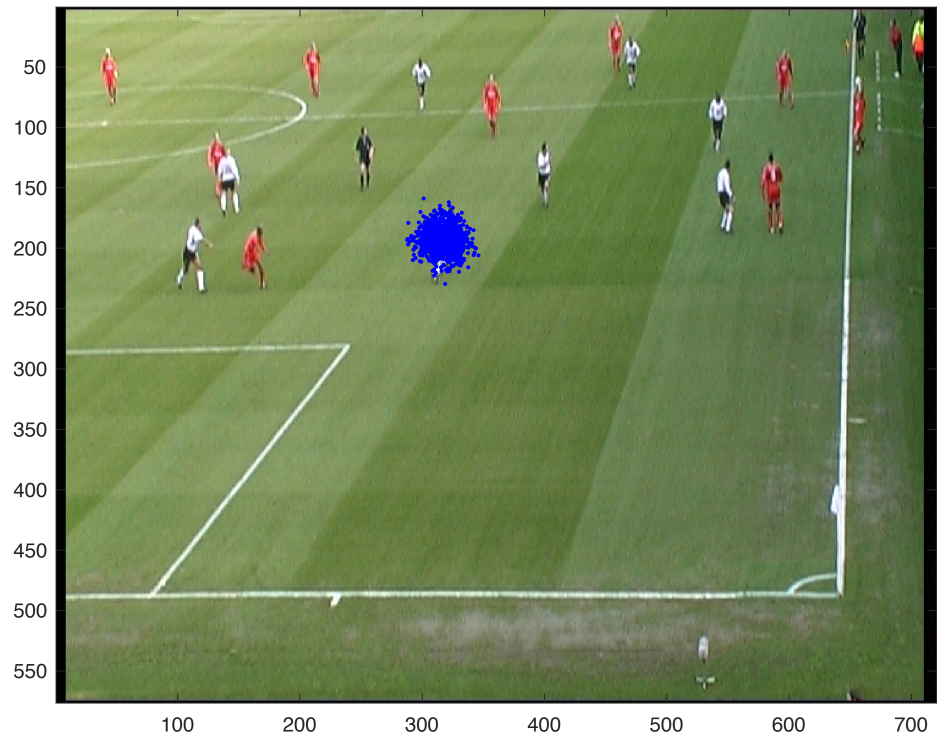
- **Generate particles (samples)**
  - Draw random samples from a 2D Normal distribution.
  - MATLABs `randn` returns normally distributed samples.





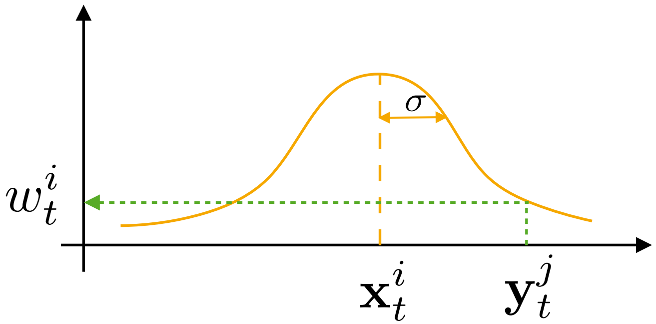
Q2: Particle Filter a)

- **Generate particles (samples)**
  - Draw random samples from a 2D Normal distribution.
  - MATLABs **`randn`** returns normally distributed samples.

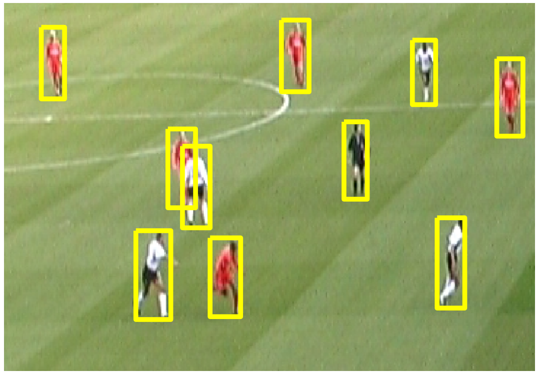


Q2: Particle Filter b)

- **Compute particle likelihood (weights)**
  - How likely does a particle correspond to a detection?
  - Measure: Parzen density estimation with Gaussian kernel



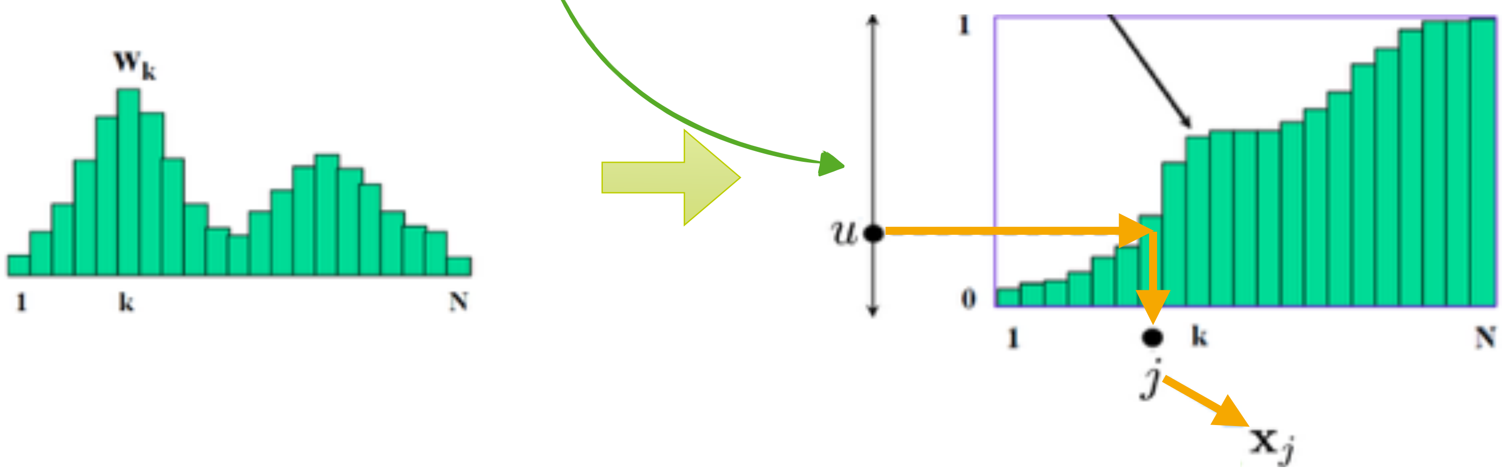
$$w_t^i = \sum_j exp\Big(-\frac{1}{2}(\frac{\mathbf{y}_x^j - \mathbf{x}_x^i}{\sigma_x^2} + \frac{\mathbf{y}_y^j - \mathbf{x}_y^i}{\sigma_y^2})\Big)$$



Q2: Particle Filter - Resampling

- **Inverse transform sampling**
  - **Goal:** resample N particles from existing set of N particles, favor particles with larger weight.

1. From discrete particle distribution compute cumulative distribution. Each bin corresponds to a particle, the height of the bin corresponds to the weight.
2. Sample u from uniform distribution between 0 and 1
3. Look up bin in cumulative distribution and pick resulting particle $\mathbf{x}_j$



Q2: Particle Filter - Result