Autonomous Mobile Robot Design

Topic: Extended Kalman Filter

Dr. Kostas Alexis (CSE)

These slides relied on the lectures from C. Stachniss, J. Sturm and the book “Probabilistic Robotics” from Thurn et al.

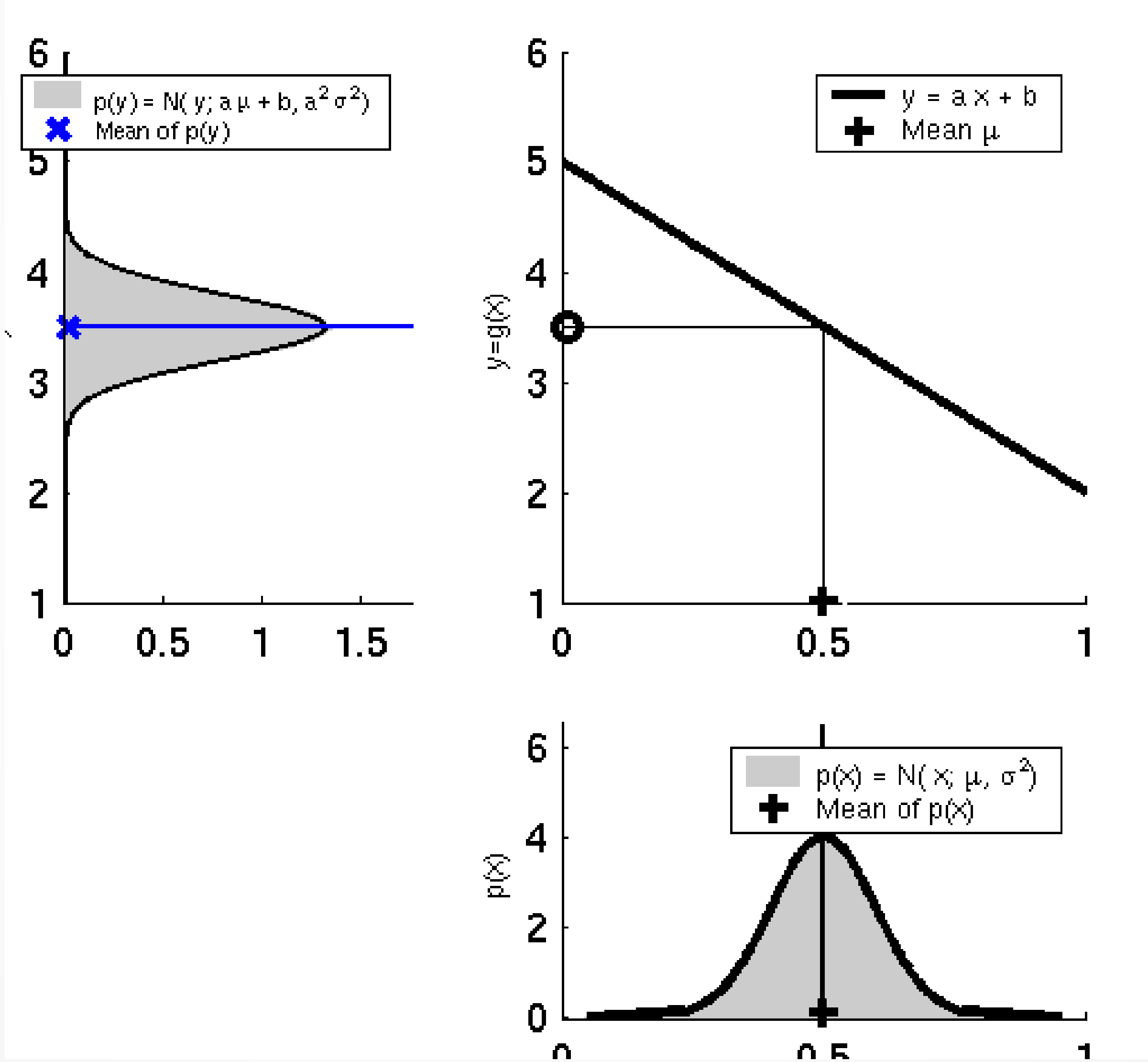
Kalman Filter Assumptions

- Gaussian distributions and noise
- Linear motion and observation model
  - **What if this is not the case?**

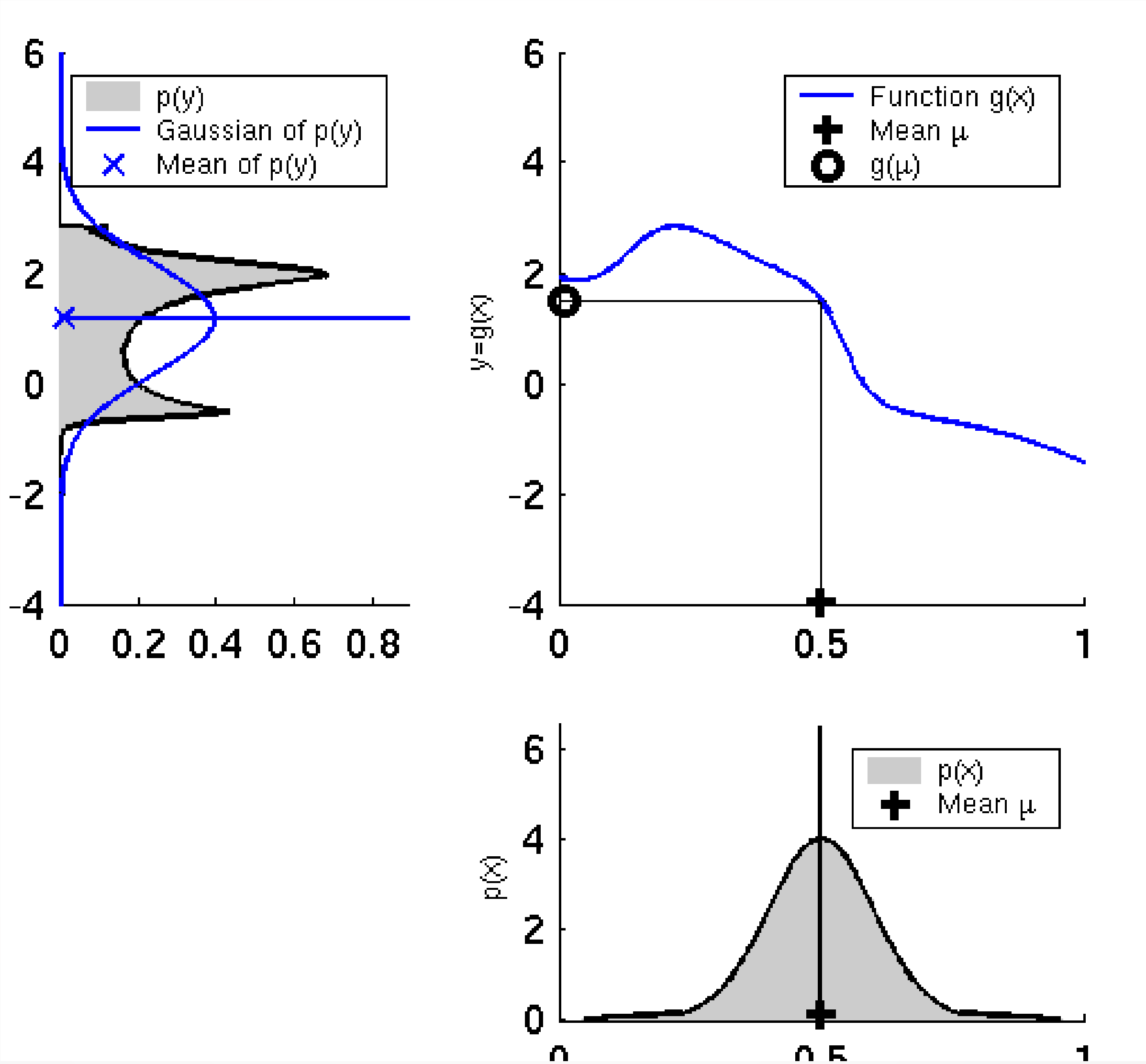
$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

$$z_t = C_t x_t + \delta_t$$

Linearity Assumption Revisited



Nonlinear Function



Nonlinear Dynamical Systems

- Real-life robots are mostly nonlinear systems.
- The **motion equations** are expressed as **nonlinear differential (or difference) equations**:

$$x_t = g(u_t, x_{t-1})$$

- Also leading to a **nonlinear observation function:**

$$z_t = h(x_t)$$

Taylor Expansion

- Solution: approximate via linearization of both functions
- **Motion Function:**

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1})$$
$$= g(\mu_{t-1}, u_t) + G_t(x_{t-1} - \mu_{t-1})$$

- **Observation Function:**

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t}(x_t - \mu_t)$$
$$= h(\bar{\mu}_t) + H_t(x_t - \mu_t)$$

Reminder: Jacobian Matrix

- It is a non-square matrix *mxn* in general
- Given a vector-valued function:

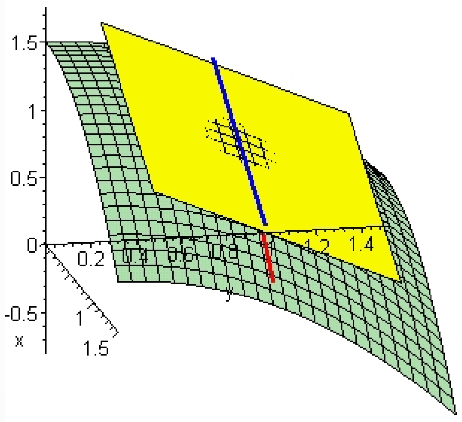
$$g(x) = \begin{pmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{pmatrix}$$

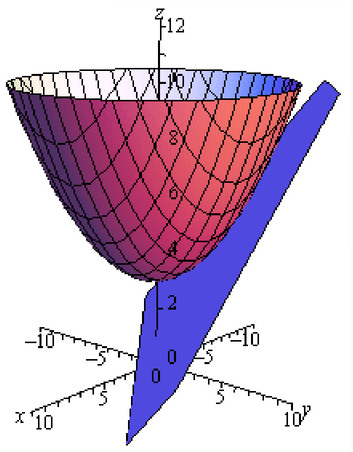
- The **Jacobian matrix** is defined as:

$$G_x = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_n} \end{pmatrix}$$

Reminder: Jacobian Matrix

- It is the orientation of the tangent plane to the vector-valued function at a given point





Courtesy: K. Arras

- Generalizes the gradient of a scaled-valued function.

Extended Kalman Filter

- For each time step, do:
- **Apply Motion Model:**

$$\bar{\mu}_t = g(\mu_{t-1}, u_t)$$

$$\bar{\Sigma}_t = G_t \Sigma G_t^\top + Q \quad \text{with} \quad G_t = \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}$$

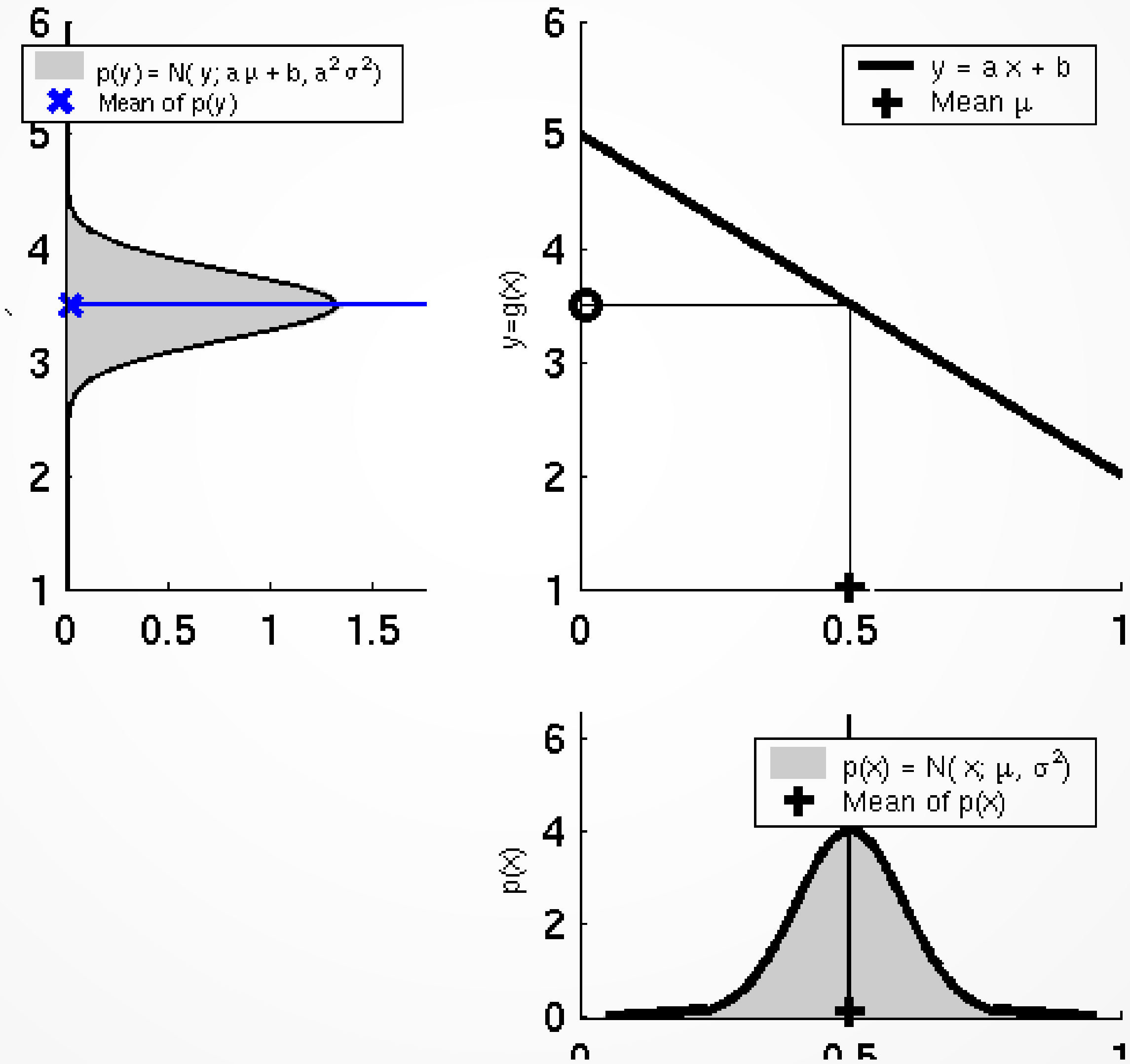
- **Apply Sensor Model:**

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

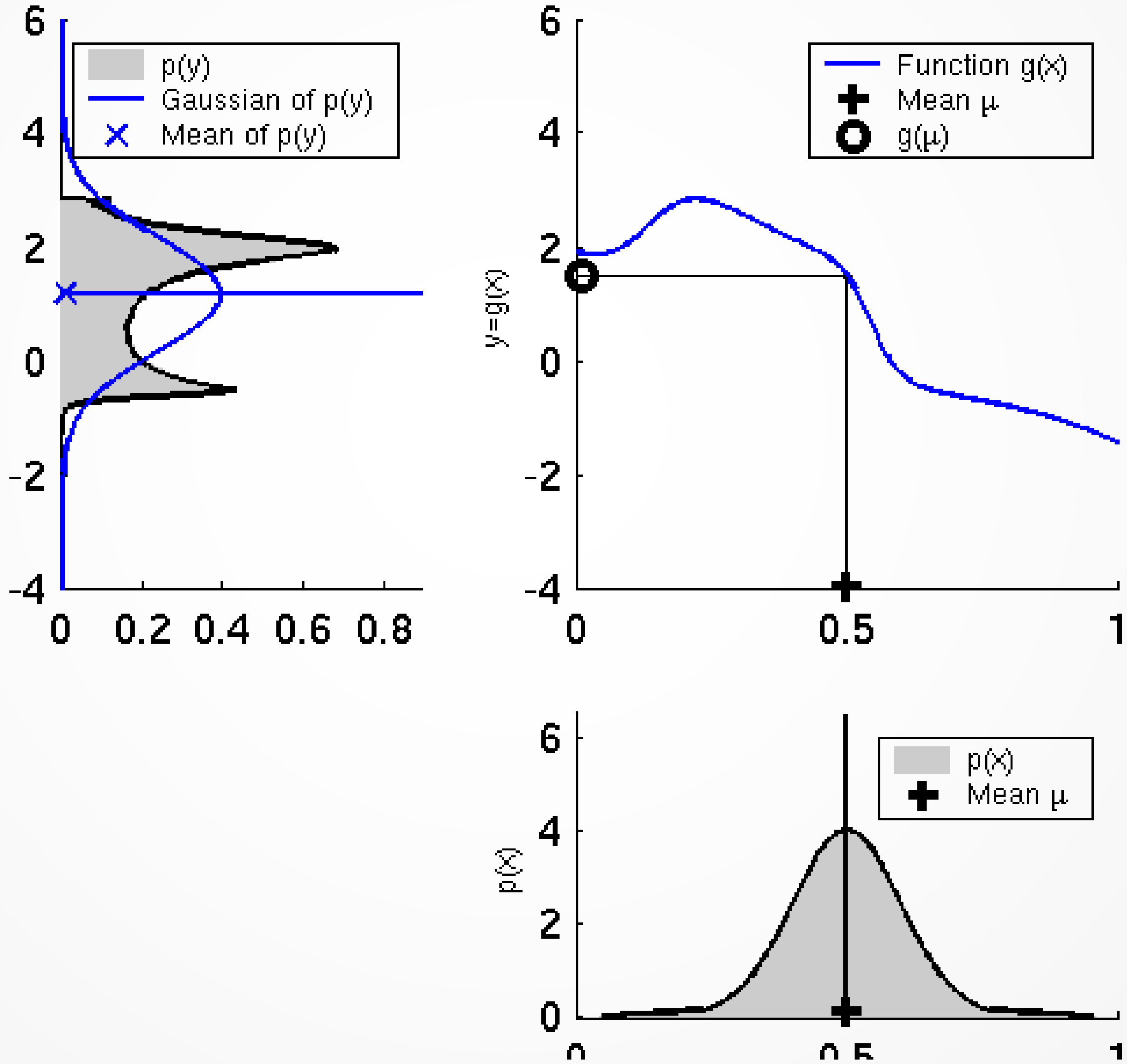
$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$$

where $K_t = \bar{\Sigma}_t H_t^\top (H_t \bar{\Sigma}_t H_t^\top + R)^{-1}$ and $H_t = \dfrac{\partial h(\bar{\mu}_t)}{\partial x_t}$

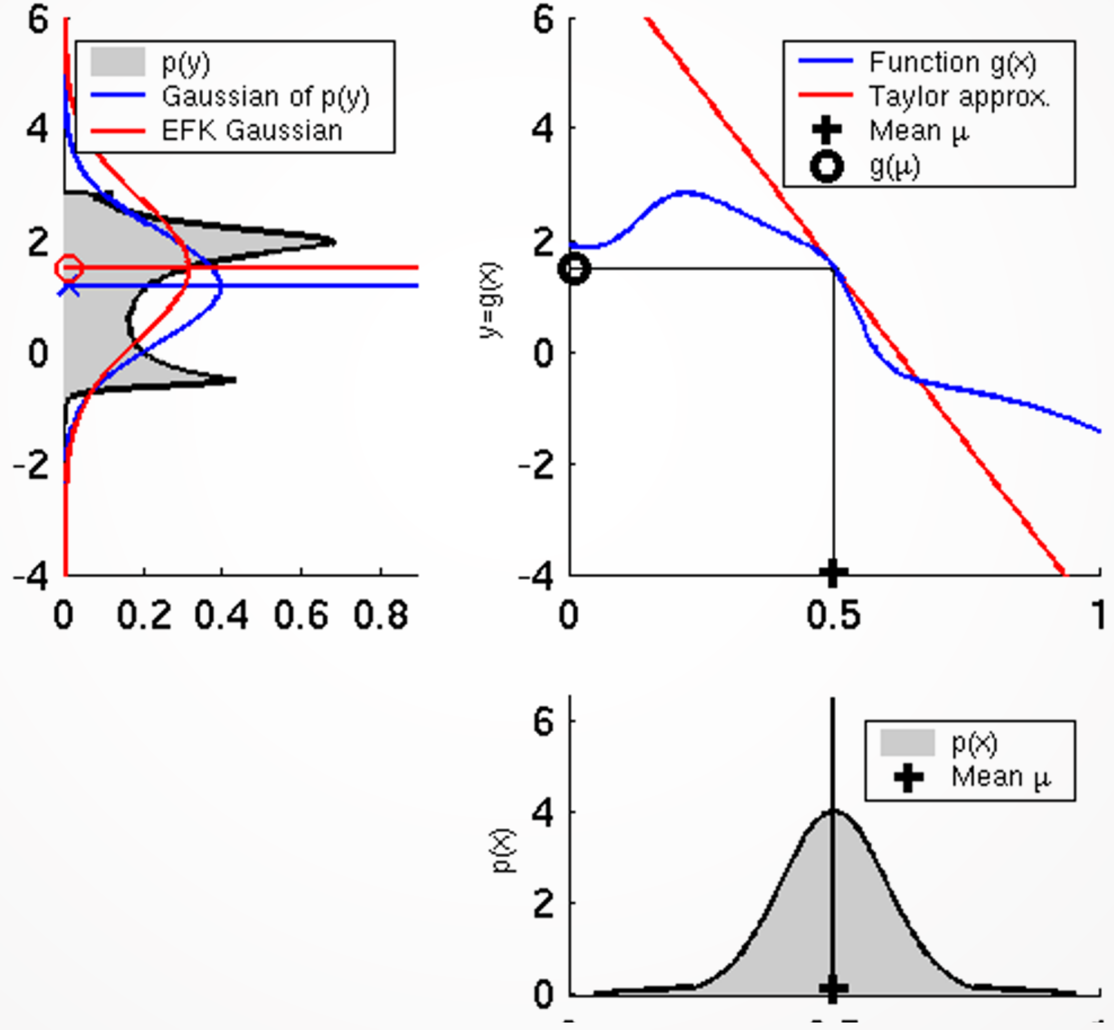
Linearity Assumption Revisited



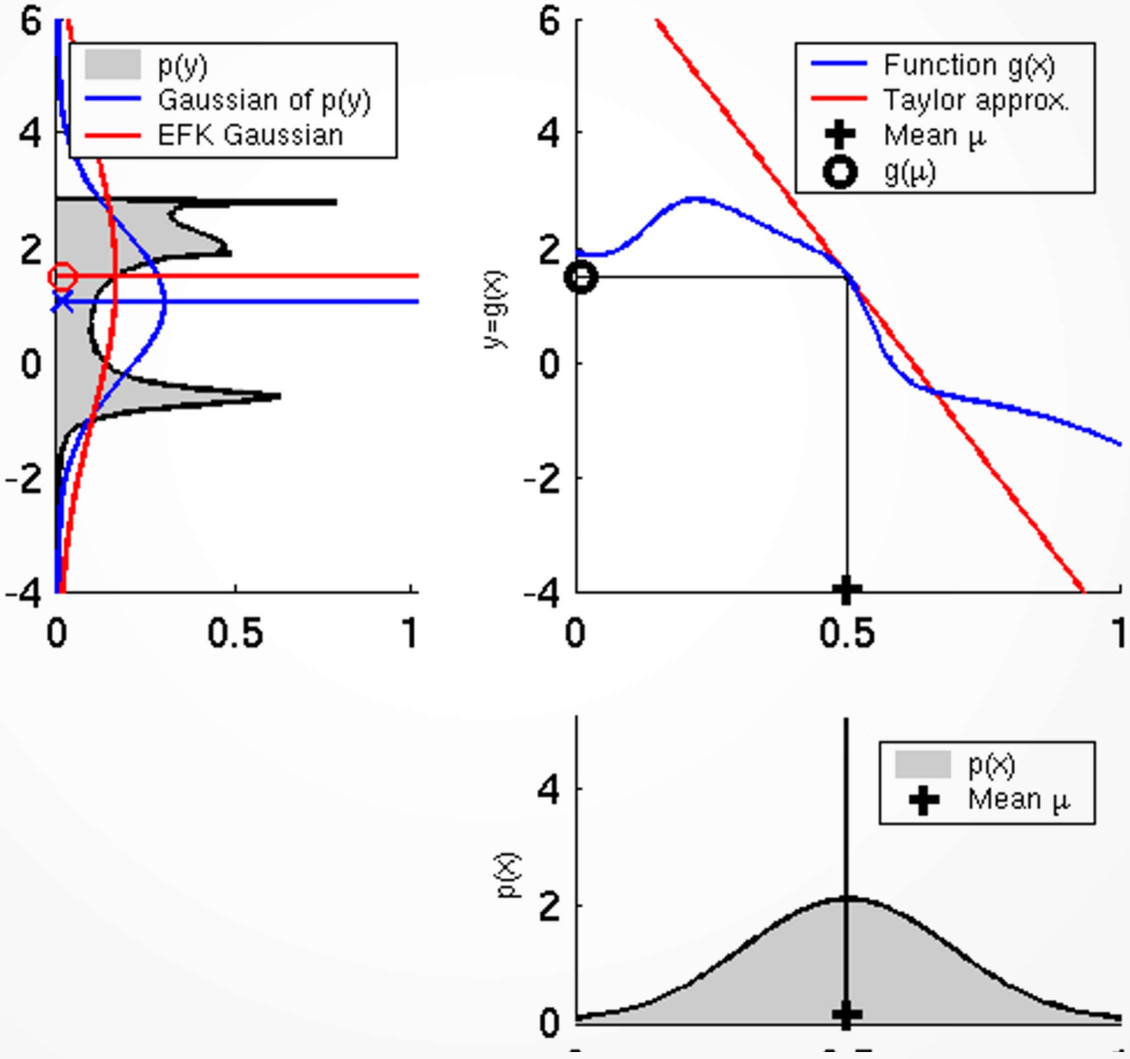
Nonlinear Function



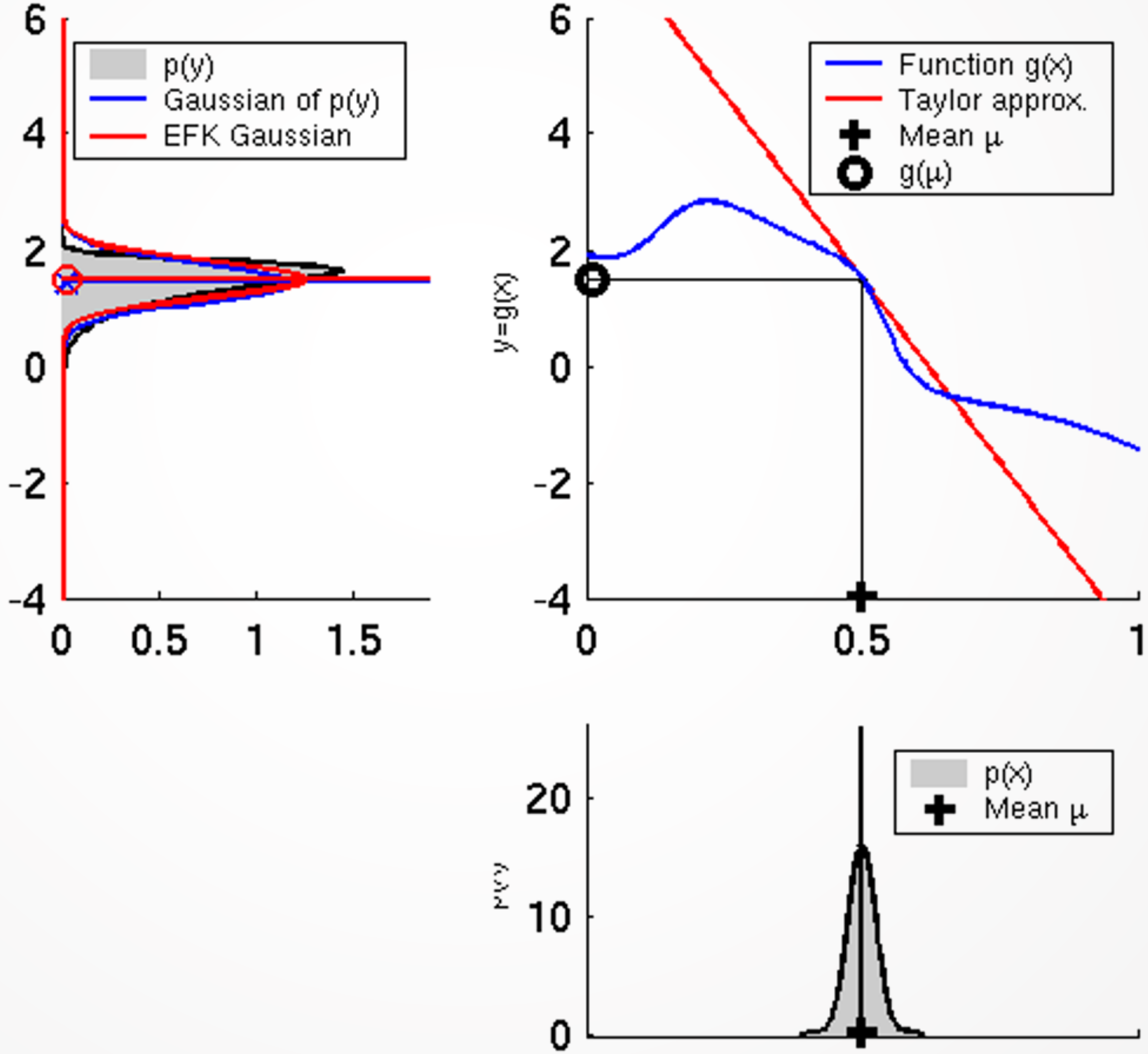
EKF Linearization (1)



EKF Linearization (2)



EKF Linearization (3)



Linearized Motion Model

- The linearized model leads to:

$$p(x_t \mid u_t, x_{t-1}) \approx \det\left(2\pi R_t\right)^{-\frac{1}{2}}$$
$$\exp\Big(-\frac{1}{2}\left(x_t - g(u_t, \mu_{t-1}) - G_t\left(x_{t-1} - \mu_{t-1}\right)\right)^T$$
$$R_t^{-1}\left(x_t - \underbrace{g(u_t, \mu_{t-1}) - G_t\left(x_{t-1} - \mu_{t-1}\right)}_{\text{linearized model}}\right)\Big)$$

- R_t describes the noise of the motion.

Linearized Observation Model

- The linearized model leads to:

$$p(z_t \mid x_t) = \det\left(2\pi Q_t\right)^{-\frac{1}{2}}$$
$$\exp\left(-\frac{1}{2}\left(z_t - h(\bar{\mu}_t) - H_t\left(x_t - \bar{\mu}_t\right)\right)^T\right.$$
$$\left.Q_t^{-1}\left(z_t - \underbrace{h(\bar{\mu}_t) - H_t\left(x_t - \bar{\mu}_t\right)}_{\text{linearized model}}\right)\right)$$

- Q_t describes the noise of the motion.

EKF Algorithm

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

2: $\bar{\mu}_t = \underline{g(u_t, \mu_{t-1})}$

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

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

5: $\mu_t = \bar{\mu}_t + K_t(z_t - \underline{h(\bar{\mu}_t)})$

6: $\Sigma_t = (I - K_t\ H_t)\ \bar{\Sigma}_t$

7: $return\ \mu_t, \Sigma_t$

$A_t \leftrightarrow G_t$

$C_t \leftrightarrow H_t$

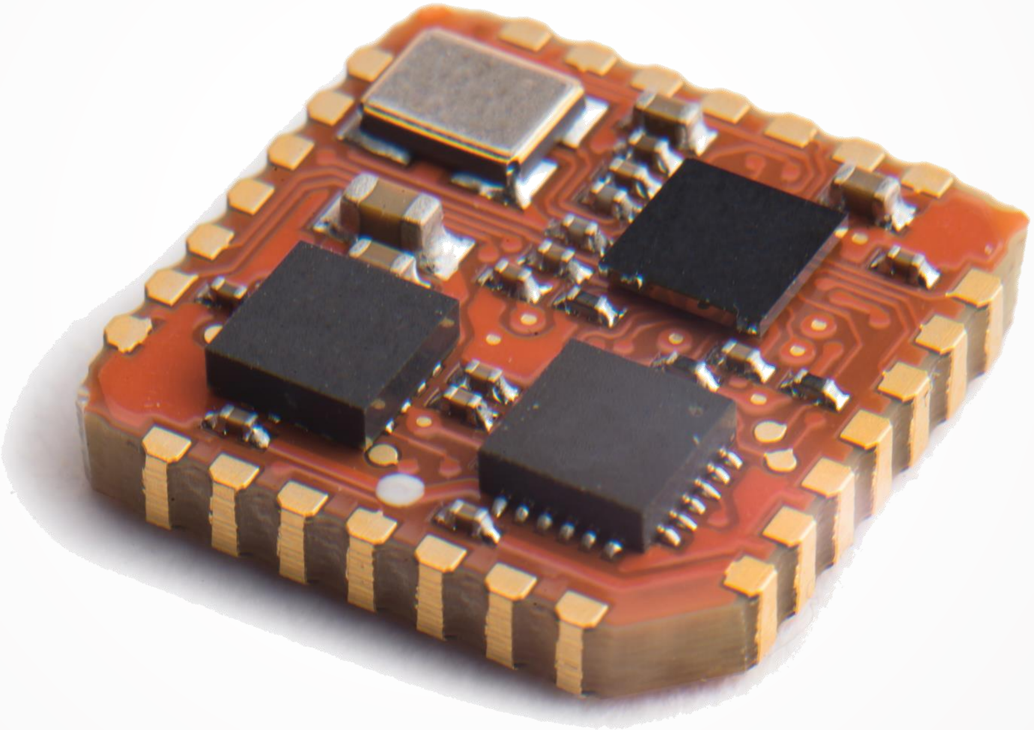
KF vs EKF

EKF Summary

- Extension of the Kalman Filter.
- One way to deal with nonlinearities.
- Performs local linearizations.
- Works well in practice for moderate nonlinearities.
- Large uncertainty leads to increased approximation error.

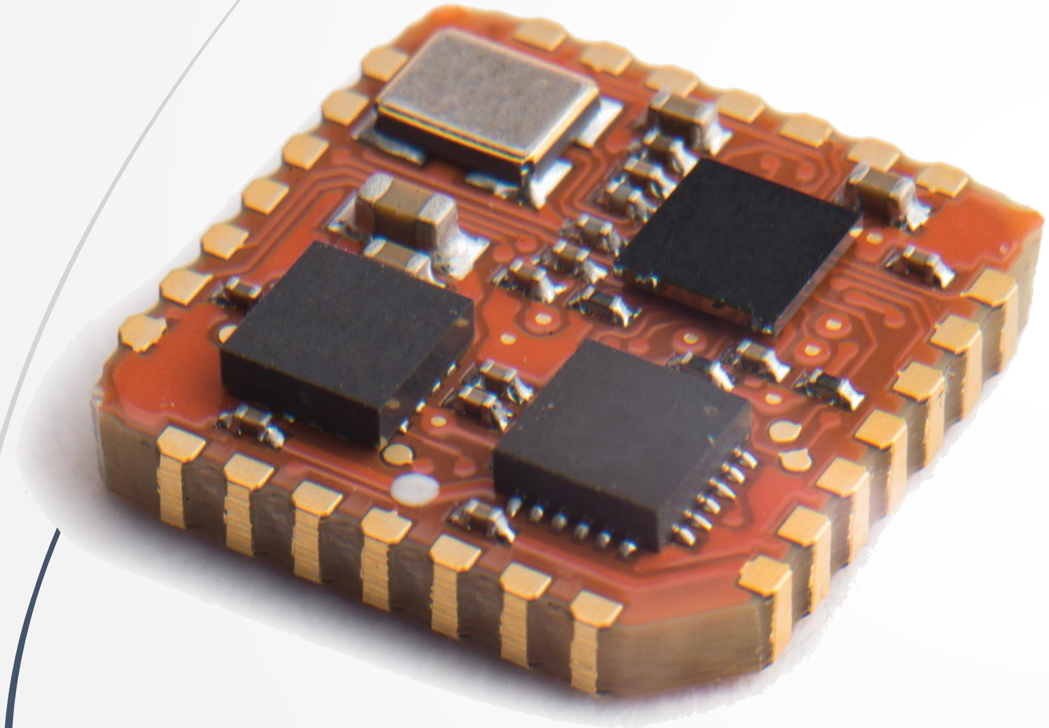
EKF Discussion

IMU



EKF Discussion

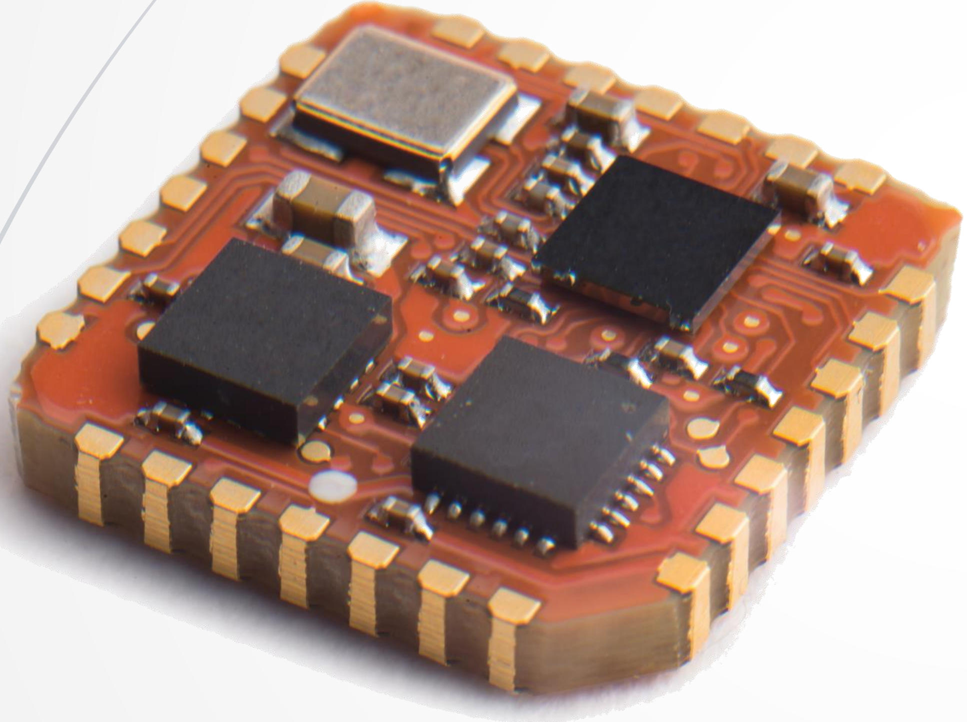
IMU + Compass



GPS

EKF Discussion

IMU + Compass



Camera

Autonomous Mobile Robot Design

Topic: Unscented Kalman Filter

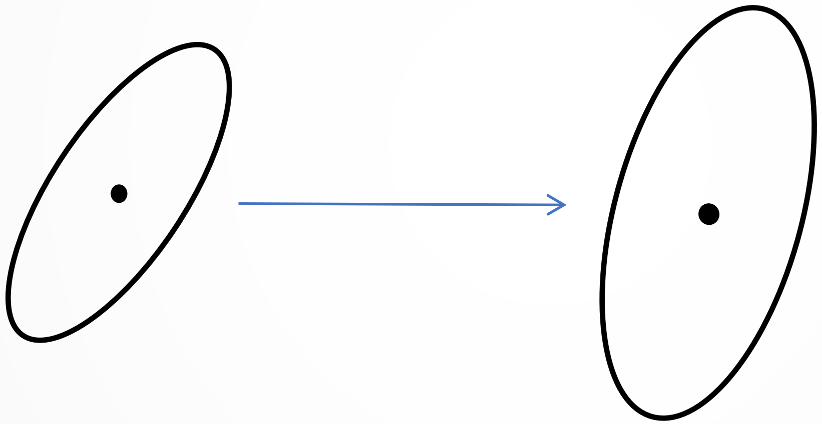
Dr. Kostas Alexis (CSE)

These slides relied on the lectures from C. Stachniss, J. Sturm and the book “Probabilistic Robotics” from Thurn et al.

KF, EKF and UKF

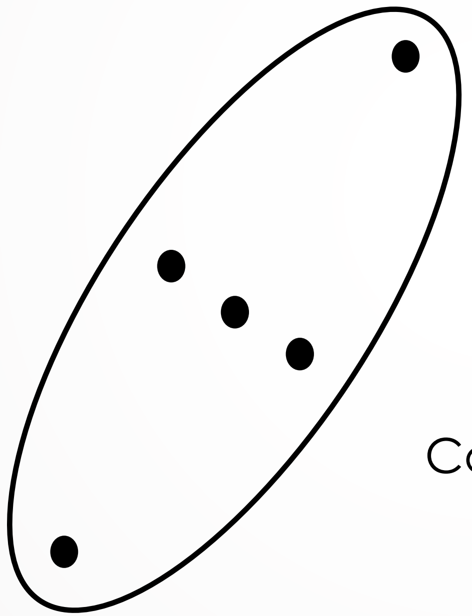
- Kalman Filter requires linear models
- Extended Kalman Filter linearizes via Taylor expansion
- Is there a better way to deal with nonlinearities?
  - Unscented Transform
  - Unscented Kalman Filter

Taylor Approximation



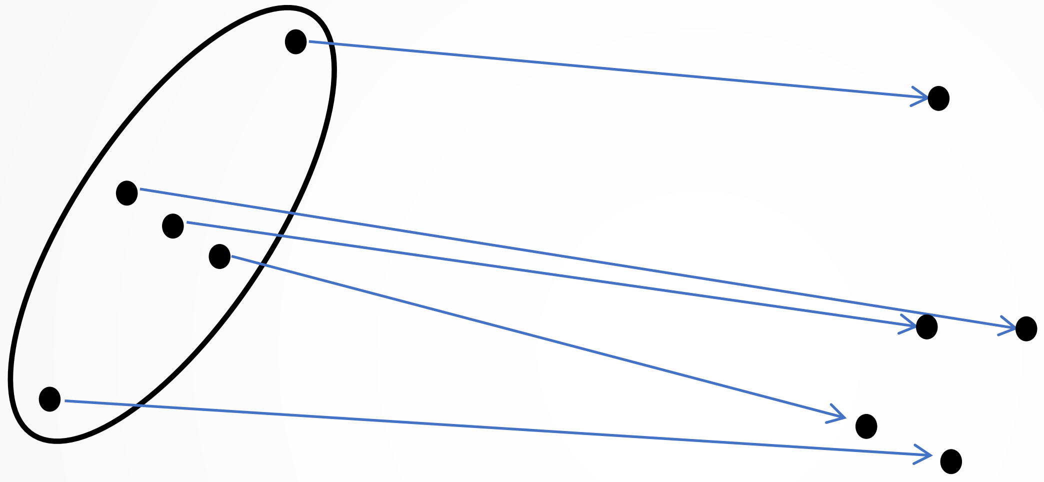
Linearization of the non-linear
function through Taylor expansion

Unscented Transform



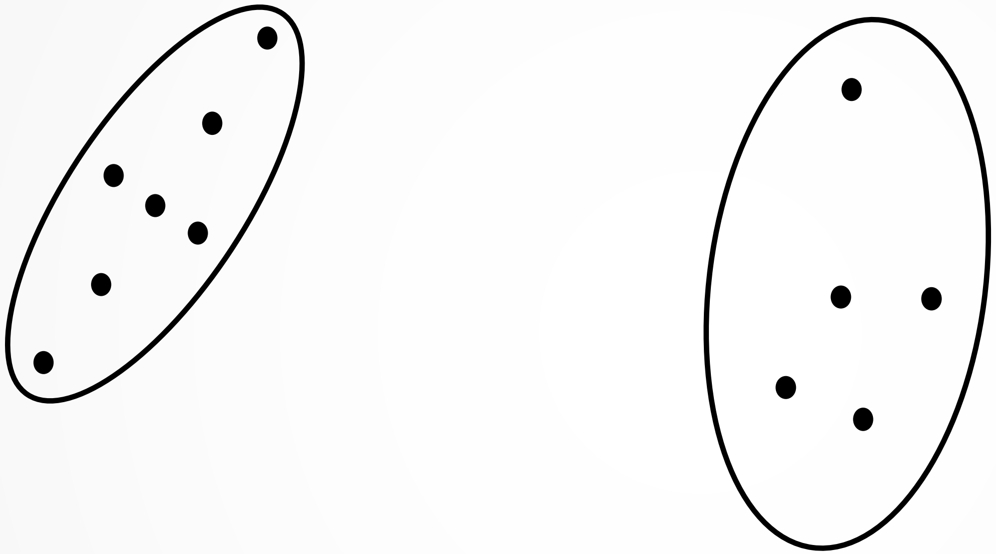
Computes a set of so-called sigma points.

Unscented Transform



Transforms each sigma point through the non-linear function.

Unscented Transform



Computes the Gaussian from the transformed and weighted sigma points.

Unscented Transform Overview

- Compute a set of sigma points
- Each sigma points has a weight
- Transform the point through the non-linear function
- Compute a Gaussian from weighted points

- Avoids to linearize around the mean as Taylor expansion (and EKF) does

Sigma points properties

- How to choose the sigma points?
- How to se the weights?
- Select $\mathcal{X}^{[i]}, w^{[i]}$ so that:

$$\sum_i w^{[i]} = 1$$

$$\mu = \sum_i w^{[i]} \mathcal{X}^{[i]}$$

$$\Sigma = \sum_i w^{[i]} (\mathcal{X}^{[i]} - \mu)(\mathcal{X}^{[i]} - \mu)^T$$

- There is no unique solution for $\mathcal{X}^{[i]}, w^{[i]}$

Sigma points

- Choosing the sigma points

$$\mathcal{X}^{[0]} = \mu$$

  - First sigma point is the mean

Sigma points

- Choosing the sigma points

$$
\begin{aligned}
\mathcal{X}^{[0]} &= \mu \\
\mathcal{X}^{[i]} &= \mu + \left(\sqrt{(n+\lambda)\,\Sigma}\right)_i \qquad \text{for } i = 1, \ldots, n \\
\mathcal{X}^{[i]} &= \mu - \left(\sqrt{(n+\lambda)\,\Sigma}\right)_{i-n} \qquad \text{for } i = n+1, \ldots, 2n
\end{aligned}
$$

matrix square root

dimensionality

scaling parameter

column vector

Matrix Square Root

- Defined as S with $\Sigma = SS$
- Computed via diagonalization

$$\begin{aligned}
\Sigma &= VDV^{-1} \\
&= V \begin{pmatrix} d_{11} & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & d_{nn} \end{pmatrix} V^{-1} \\
&= V \begin{pmatrix} \sqrt{d_{11}} & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \sqrt{d_{nn}} \end{pmatrix} \begin{pmatrix} \sqrt{d_{11}} & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \sqrt{d_{nn}} \end{pmatrix} V^{-1}
\end{aligned}$$

Matrix Square Root

- Thus we can define:

$$S = V \underbrace{\begin{pmatrix} \sqrt{d_{11}} & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \sqrt{d_{nn}} \end{pmatrix}}_{D^{1/2}} V^{-1}$$

- so that:

$$SS = (VD^{1/2}V^{-1})(VD^{1/2}V^{-1}) = VDV^{-1} = \Sigma$$

Cholesky Matrix Square Root

- Alternative definition of the matrix square root:

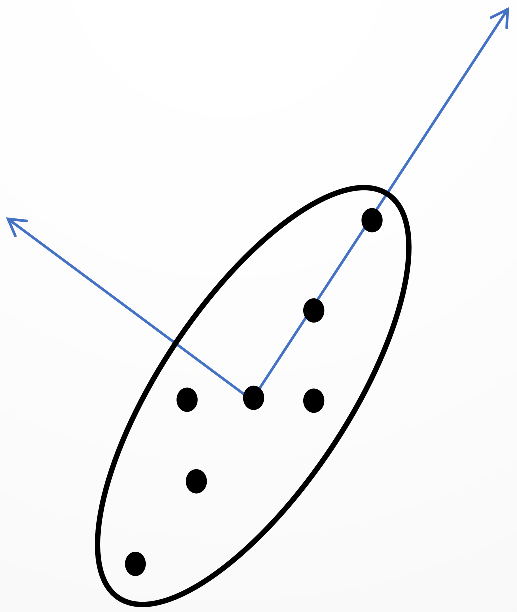
$$L \text{ with } \Sigma = LL^T$$

- Result of the Cholesky decomposition
- Numerically stable solution
- Often used in UKF implementations
- L and Σ have the same Eigenvectors

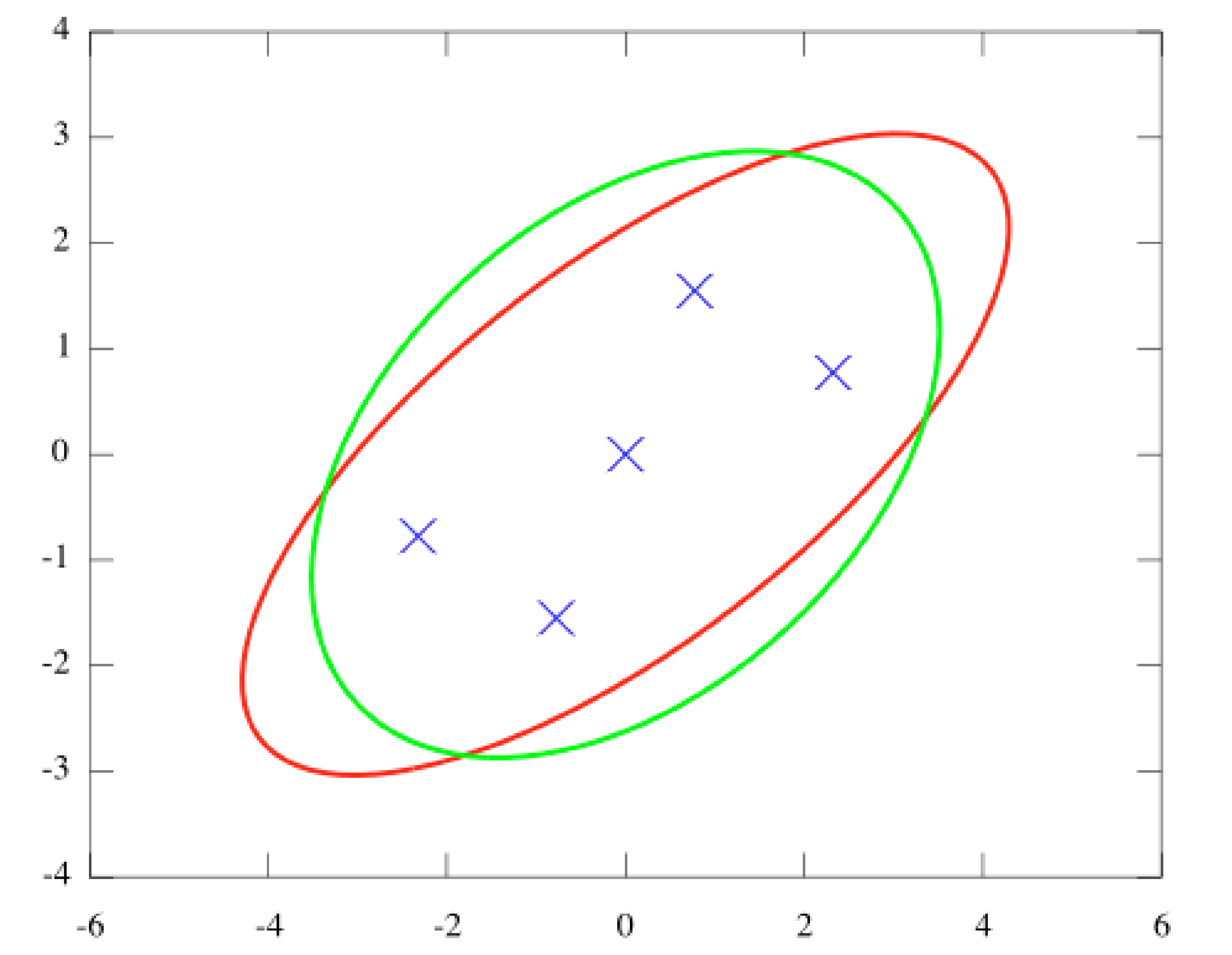
Sigma Points and Eigenvectors

- Sigma point can but does not have to lie on the main axes of Σ

$$
\begin{aligned}
\mathcal{X}^{[i]} &= \mu + \left(\sqrt{(n+\lambda)\,\Sigma}\right)_i & \text{for } i = 1, \ldots, n \\
\mathcal{X}^{[i]} &= \mu - \left(\sqrt{(n+\lambda)\,\Sigma}\right)_{i-n} & \text{for } i = n+1, \ldots, 2n
\end{aligned}
$$



Sigma Points Example



Sigma Points Weights

- Weight sigma points

for computing the mean

parameters

$$w_m^{[0]} = \frac{\lambda}{n+\lambda}$$

$$w_c^{[0]} = w_m^{[0]} + (1 - \alpha^2 + \beta)$$

$$w_m^{[i]} = w_c^{[i]} = \frac{1}{2(n+\lambda)} \quad \text{for } i = 1, \ldots, 2n$$

for computing the covariance

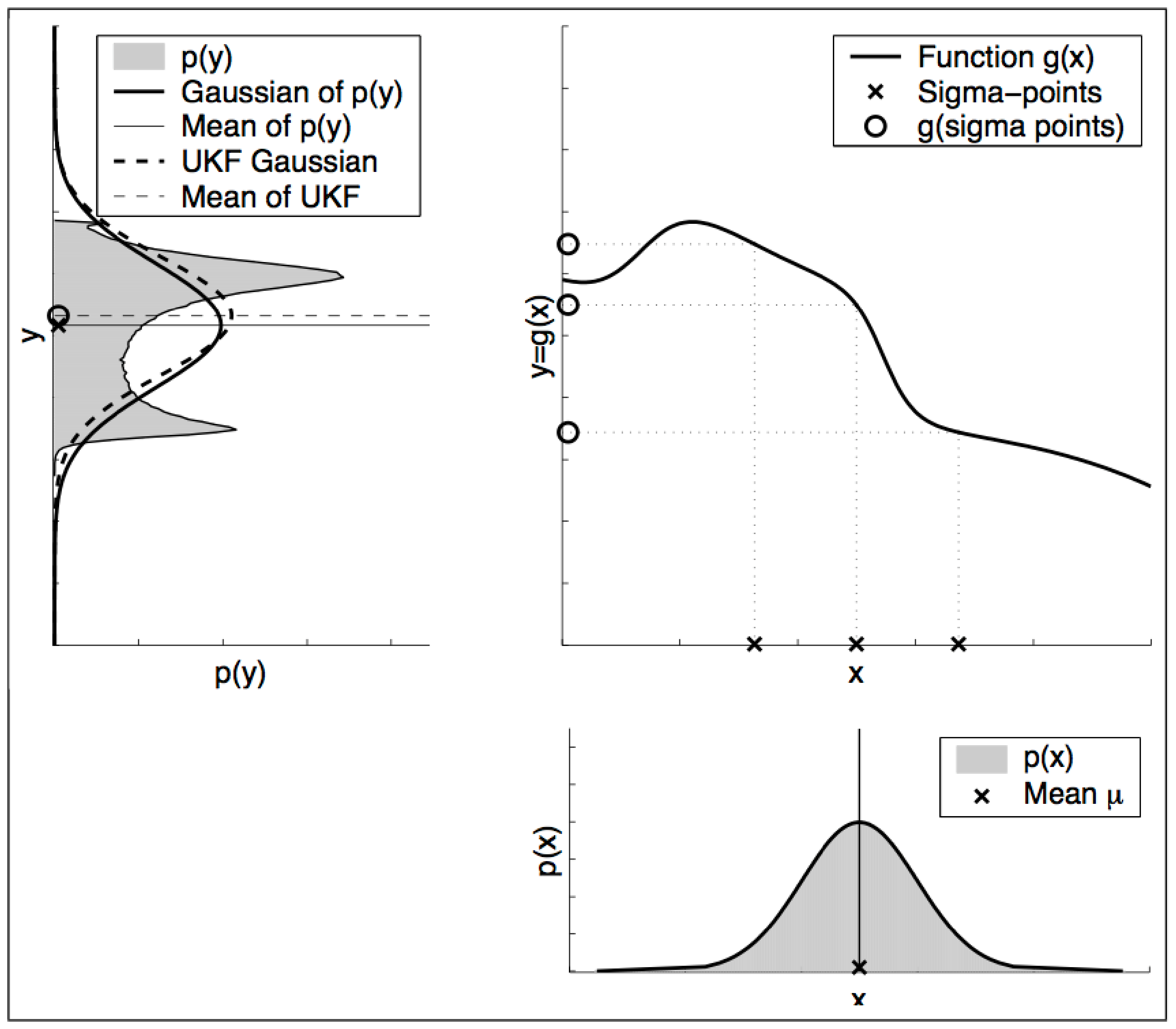
Recover the Gaussian

- Compute Gaussian from weighted and transformed points

$$\mu' = \sum_{i=0}^{2n} w_m^{[i]} \, g(\mathcal{X}^{[i]})$$

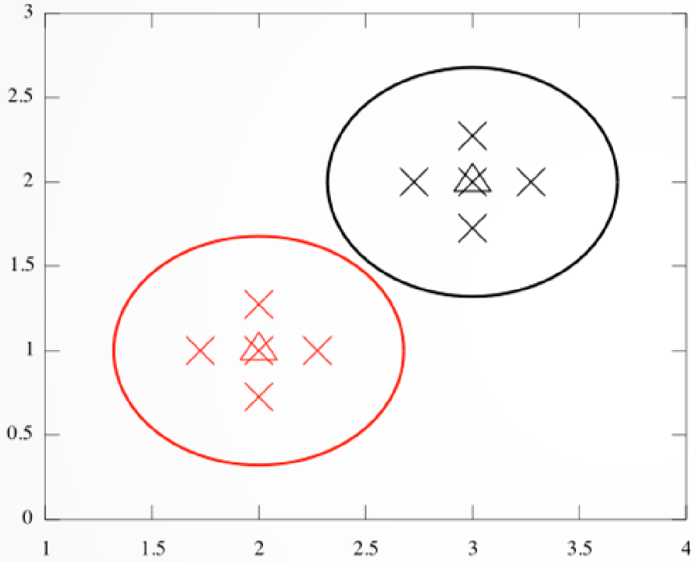
$$\Sigma' = \sum_{i=0}^{2n} w_c^{[i]} \, (g(\mathcal{X}^{[i]}) - \mu')(g(\mathcal{X}^{[i]}) - \mu')^T$$

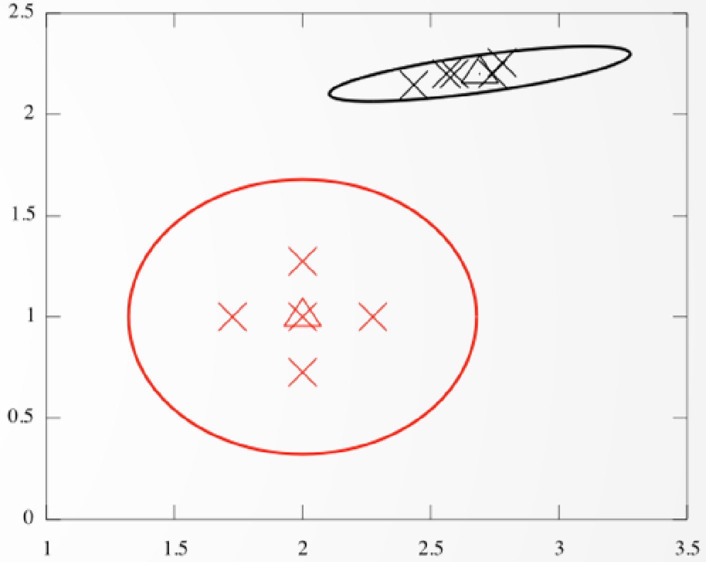
Example



p(y)
Gaussian of p(y)
Mean of p(y)
UKF Gaussian
Mean of UKF
y
p(y)
Function g(x)
Sigma–points
g(sigma points)
y=g(x)
x
p(x)
Mean μ
p(x)
x

Examples





$$g((x,y)^T) = \begin{pmatrix} x+1 \\ y+1 \end{pmatrix}^T$$

$$g((x,y)^T) = \begin{pmatrix} 1+x+\sin(2x)+\cos(y) \\ 2+0.2y \end{pmatrix}^T$$

Unscented Transform Summary

- Sigma points

$$
\begin{aligned}
\mathcal{X}^{[0]} &= \mu \\
\mathcal{X}^{[i]} &= \mu + \left(\sqrt{(n+\lambda)\,\Sigma}\right)_i \qquad \text{for } i = 1, \ldots, n \\
\mathcal{X}^{[i]} &= \mu - \left(\sqrt{(n+\lambda)\,\Sigma}\right)_{i-n} \qquad \text{for } i = n+1, \ldots, 2n
\end{aligned}
$$

- Weights

$$
\begin{aligned}
w_m^{[0]} &= \frac{\lambda}{n+\lambda} \\
w_c^{[0]} &= w_m^{[0]} + (1 - \alpha^2 + \beta) \\
w_m^{[i]} = w_c^{[i]} &= \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1, \ldots, 2n
\end{aligned}
$$

Unscented Transform Parameters

- Free parameters as there is no unique solution
- Scaled Unscented Transform suggests

$$\kappa \geq 0$$

Influence how far the sigma points are away from the mean

$$\alpha \in (0, 1]$$

$$\lambda = \alpha^2(n + \kappa) - n$$

$$\beta = 2$$

Optimal choice for Gaussians

EKF Algorithm

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

2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$

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

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

5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$

6: $\Sigma_t = (I - K_t\, H_t)\, \bar{\Sigma}_t$

7: $return\ \mu_t, \Sigma_t$

EKF to UKF: Prediction

Unscented

1: $\textbf{~~Extended~~\_Kalman\_filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)\textbf{:}$

2: $\bar{\mu}_t =$

3: $\bar{\Sigma}_t =$

replace this by sigma point propagation of the motion

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

5: $\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$

6: $\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$

7: $return \; \mu_t, \Sigma_t$

UKF Algorithm: Prediction

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

2: $\mathcal{X}_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \sqrt{(n+\lambda)\Sigma_{t-1}} \quad \mu_{t-1} - \sqrt{(n+\lambda)\Sigma_{t-1}})$

3: $\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1})$

4: $\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]}$

5: $\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)^T + R_t$

EKF to UKF: Correction

Unscented

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

2: $\bar{\mu}_t =$ replace this by sigma point propagation of

3: $\bar{\Sigma}_t =$ the motion

4: use sigma point propagation for the expected observation and Kalman gain

5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$

6: $\Sigma_t = (I - K_t\, H_t)\, \bar{\Sigma}_t$

7: $return\ \mu_t, \Sigma_t$

UKF Algorithm: Correction (1)

6: $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{(n+\lambda)\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{(n+\lambda)\bar{\Sigma}_t})$

7: $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$

8: $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$

9: $S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$

10: $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$

11: $K_t = \bar{\Sigma}_t^{x,z} \ S_t^{-1}$

UKF Algorithm: Correction (1)

6: $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{(n+\lambda)\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{(n+\lambda)\bar{\Sigma}_t})$

7: $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$

8: $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$

9: $S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$

10: $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$

11: $K_t = \bar{\Sigma}_t^{x,z} \; S_t^{-1}$

$$K_t = \overbrace{\bar{\Sigma}_t \; H_t^T}^{\bar{\Sigma}_t^{x,z}} \overbrace{(H_t \; \bar{\Sigma}_t \; H_t^T + Q_t)}^{S_t}{}^{-1}$$

(from EKF)

UKF Algorithm: Correction (2)

6: $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{(n+\lambda)\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{(n+\lambda)\bar{\Sigma}_t})$

7: $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$

8: $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$

9: $S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$

10: $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$

11: $K_t = \bar{\Sigma}_t^{x,z} \, S_t^{-1}$

12: $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$

13: $\Sigma_t = \bar{\Sigma}_t - K_t \, S_t \, K_t^T$

14: return μ_t, Σ_t

UKF Algorithm: Correction (1)

6: $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{(n+\lambda)\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{(n+\lambda)\bar{\Sigma}_t})$

7: $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$

8: $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$

9: $S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$

10: $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$

11: $K_t = \bar{\Sigma}_t^{x,z} \; S_t^{-1}$

12: $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$

13: $\Sigma_t = \bar{\Sigma}_t - K_t \; S_t \; K_t^T$

14: return μ_t, Σ_t

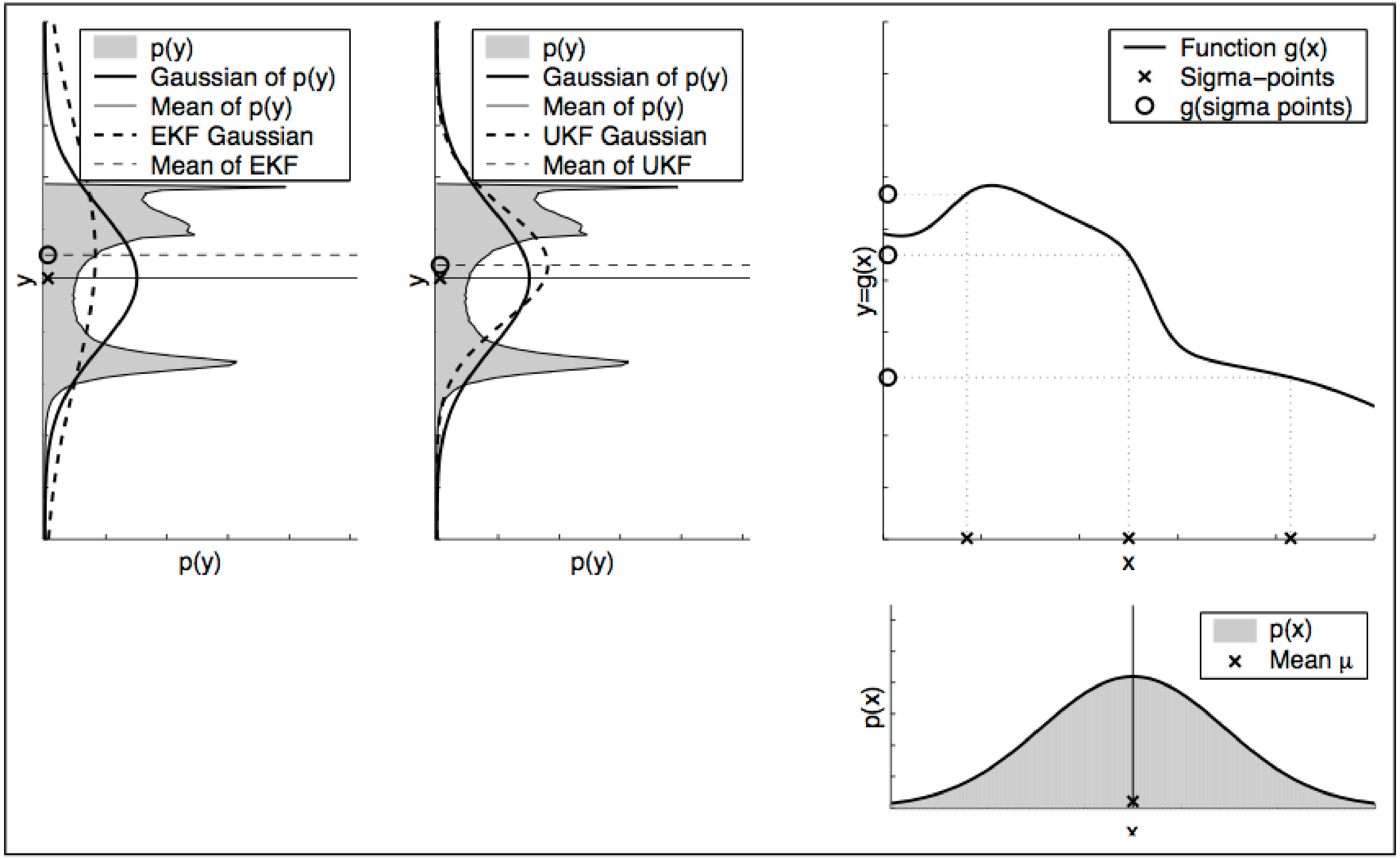
$$\begin{aligned}
\Sigma_t &= (I - K_t H_t)\bar{\Sigma}_t \\
&= \bar{\Sigma}_t - K_t H_t \bar{\Sigma}_t \\
&= \bar{\Sigma}_t - K_t \left(\Sigma^{x,z}\right)^T \\
&= \bar{\Sigma}_t - K_t \left(\Sigma^{x,z} s_t^{-1} S_t\right)^T \\
&= \bar{\Sigma}_t - K_t \left(K_t S_t\right)^T \\
&= \bar{\Sigma}_t - K_t S_t^T K_t^T \\
&= \bar{\Sigma}_t - K_t S_t K_t^T
\end{aligned}$$

(see next slide)

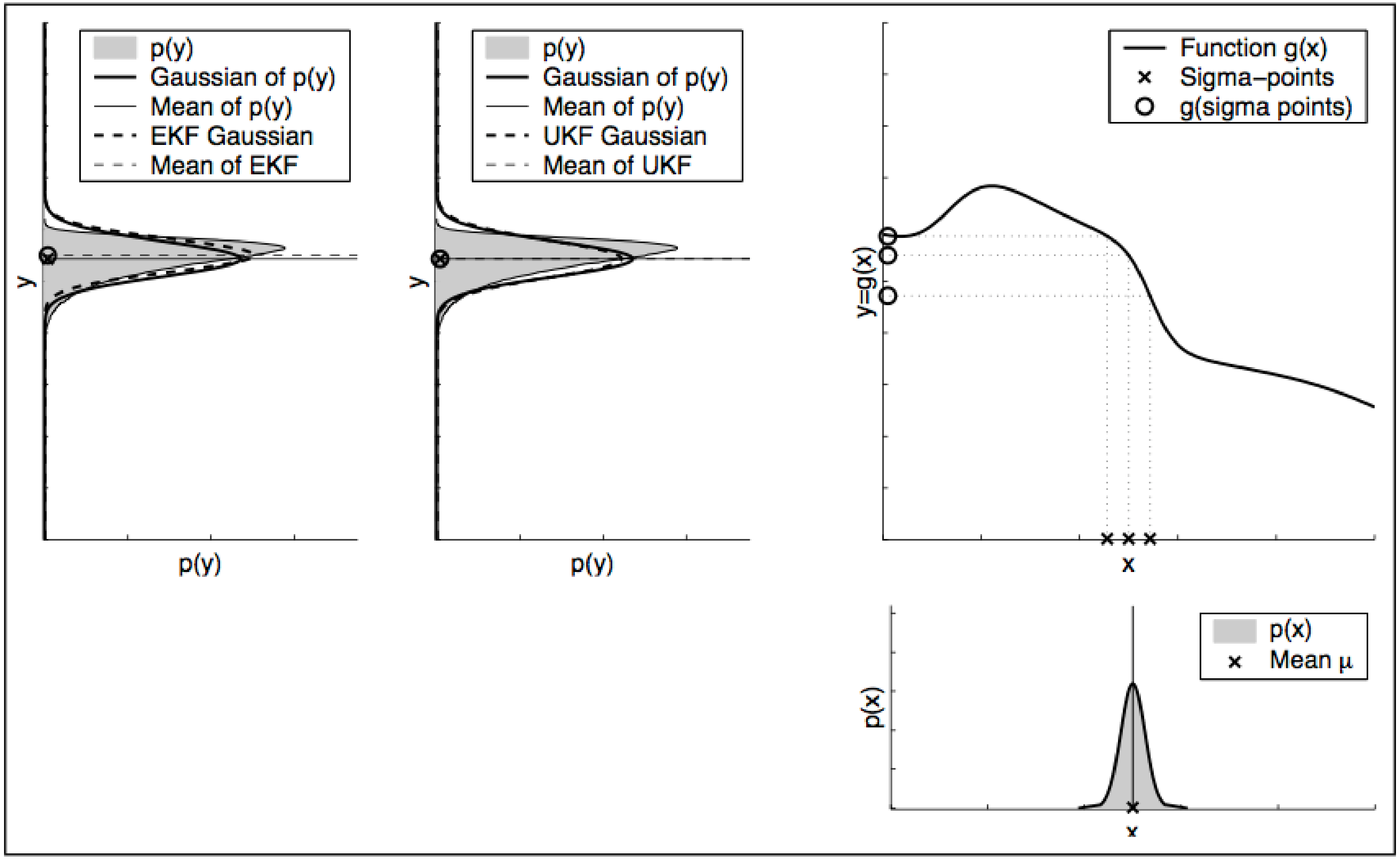
EKF-to-UKF: Computing the Covariance

$$
\begin{aligned}
\Sigma_t &= (I - K_t H_t)\bar{\Sigma}_t \\
&= \bar{\Sigma}_t - K_t \underbrace{H_t \bar{\Sigma}_t} \\
&= \bar{\Sigma}_t - K_t \left(\bar{\Sigma}^{x,z}\right)^T \\
&= \bar{\Sigma}_t - K_t \left(\underbrace{\bar{\Sigma}^{x,z} S_t^{-1}} S_t\right)^T \\
&= \bar{\Sigma}_t - K_t \left(K_t S_t\right)^T \\
&= \bar{\Sigma}_t - K_t S_t^T K_t^T \\
&= \bar{\Sigma}_t - K_t S_t K_t^T
\end{aligned}
$$

UKF vs EKF

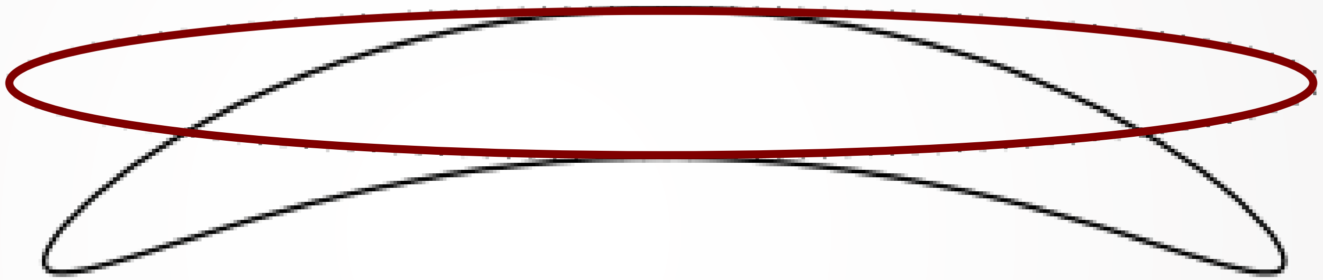

p(y)
Gaussian of p(y)
Mean of p(y)
EKF Gaussian
Mean of EKF
y
p(y)
p(y)
Gaussian of p(y)
Mean of p(y)
UKF Gaussian
Mean of UKF
y
p(y)
Function g(x)
Sigma−points
g(sigma points)
y=g(x)
x
p(x)
Mean μ
p(x)
x

UKF vs EKF (small covariance)

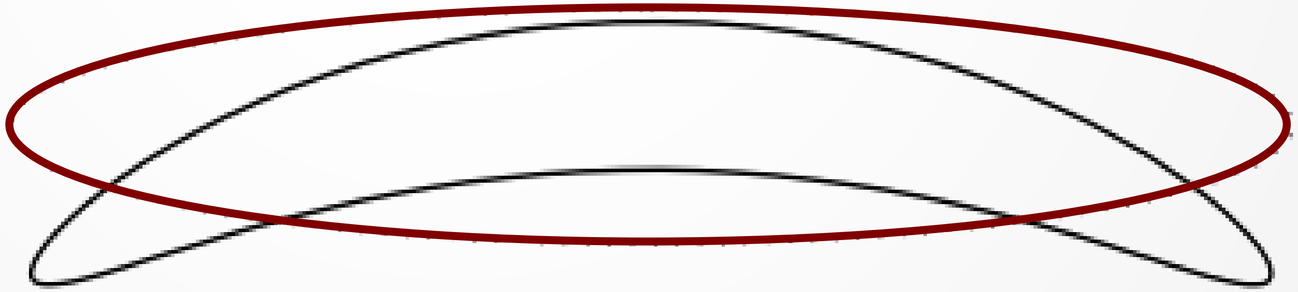


UKF vs EKF – Banana shape

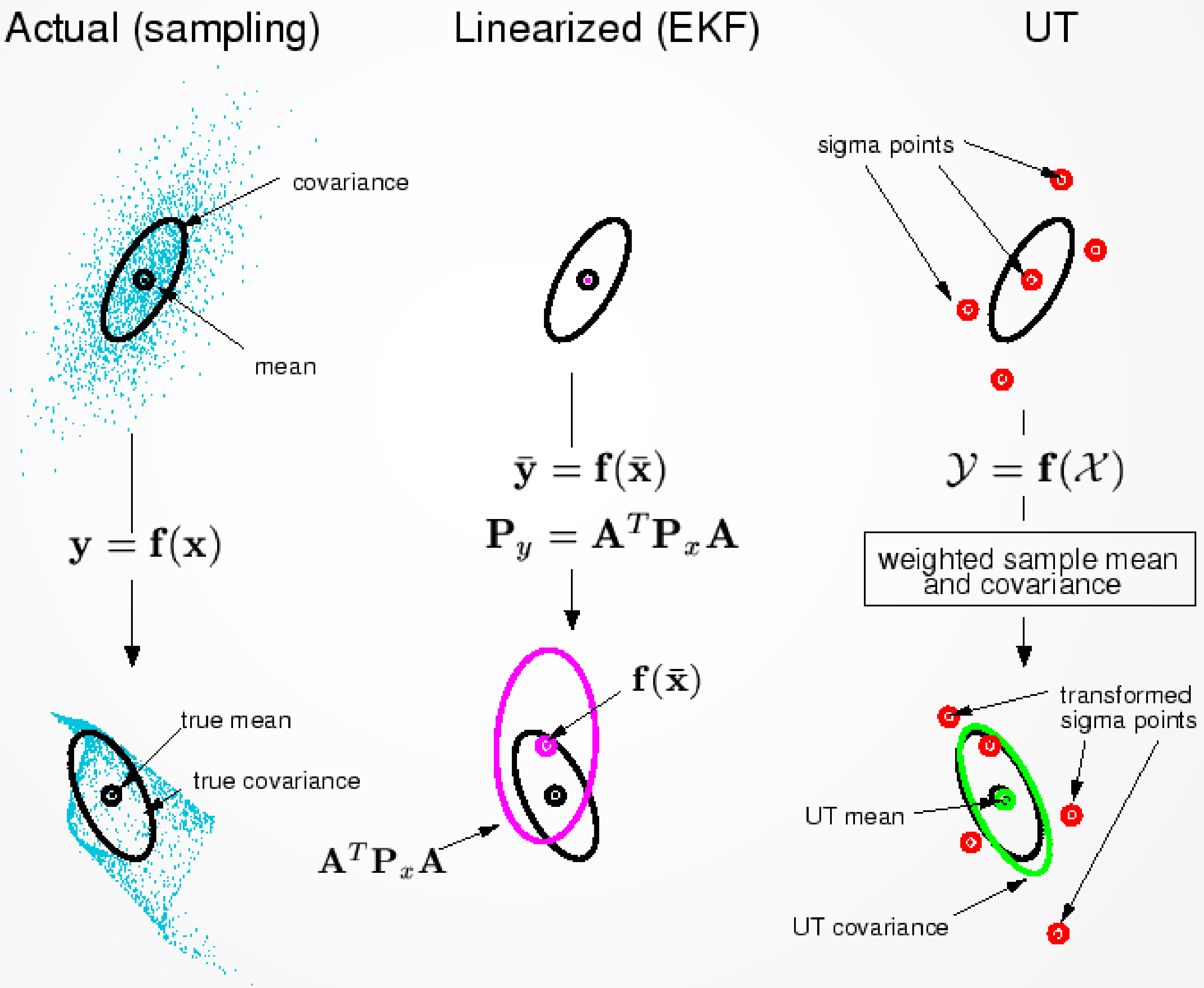
EKF approximation



UKF approximation



UKF vs EKF



Actual (sampling)
Linearized (EKF)
UT
covariance
mean
$\mathbf{y} = \mathbf{f}(\mathbf{x})$
true mean
true covariance
$\bar{\mathbf{y}} = \mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{P}_y = \mathbf{A}^T \mathbf{P}_x \mathbf{A}$
$\mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{A}^T \mathbf{P}_x \mathbf{A}$
sigma points
$\mathcal{Y} = \mathbf{f}(\mathcal{X})$
weighted sample mean and covariance
transformed sigma points
UT mean
UT covariance

UT/UKF Summary

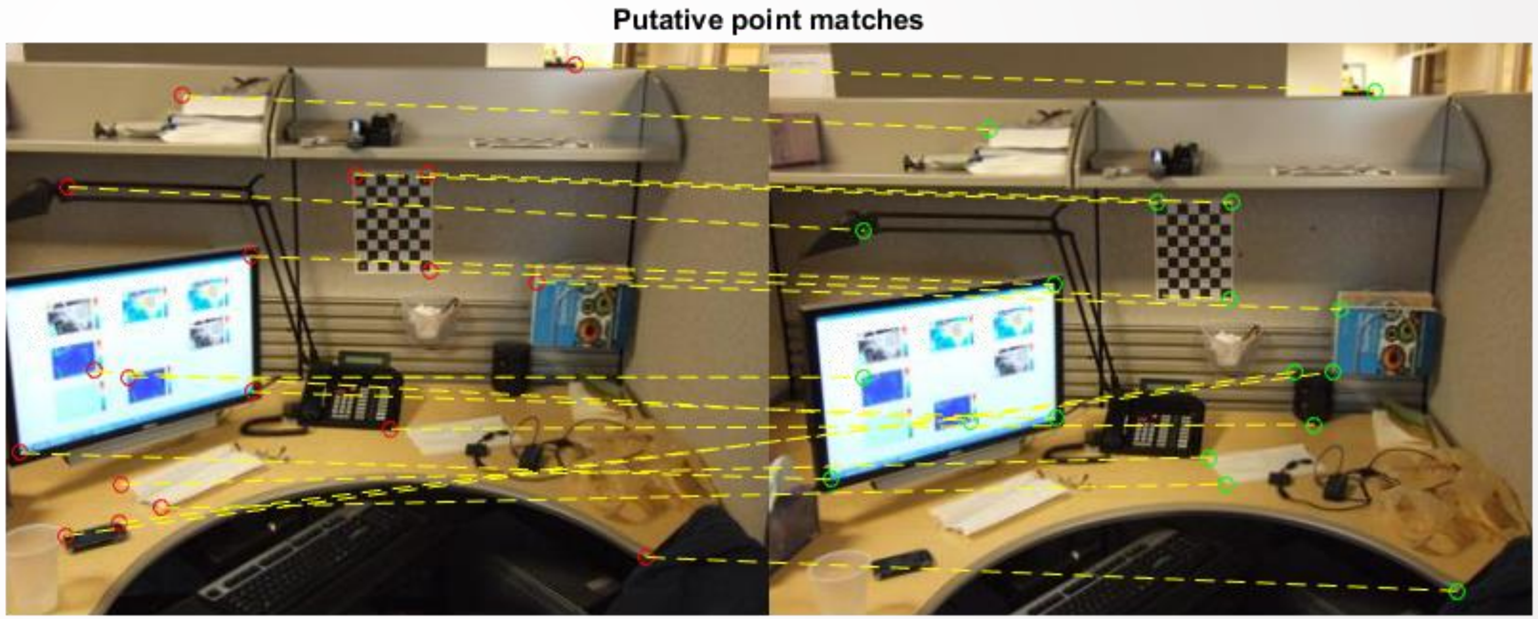
- Unscented transforms as an alternative to linearization
- UT is a better approximation than Taylor expansion
- UT uses sigma point propagation
- Free parameters in UT
- UKF uses the UT in the prediction and correction step

UKF vs EKF

- Same results as EKF for linear models
- Better approximation than EKF for non-linear models
- Differences often “somewhat small”
- No Jacobians needed for the UKF
- Same complexity class
- Slightly slower than the EKF
- Still restricted to Gaussian distributions

Code Examples and Tasks

- https://www.mathworks.com/help/vision/ref/estimatefundamentalmatrix.html



How does this apply to my project?

- State estimation is the way to use robot sensors to infer the robot state. You will use it for estimating your robot pose or its map, to track and object and be able to follow it etc.

Find out more

- Thrun et al.: “Probabilistic Robotics”, Chapter 3
- Schön and Lindsten: “Manipulating the Multivariate Gaussian Density”
- “A New Extension of the Kalman Filter to Nonlinear Systems” by Julier and Uhlmann, 1995
- http://www.cs.unc.edu/~welch/kalman/
- http://home.wlu.edu/~levys/kalman_tutorial/
- https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python
- http://www.kostasalexis.com/literature-and-links.html

Thank you!

Please ask your question!