Carnegie Mellon
THE ROBOTICS INSTITUTE

Extended Kalman Filter

Robot Localization and Mapping 16-833

Michael Kaess

September 25+27, 2023

Slides courtesy of Ryan Eustice

Nonlinear Dynamic Systems

- Most realistic robotic problems involve nonlinear functions

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

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

Projecting Covariances (Nonlinear Case)

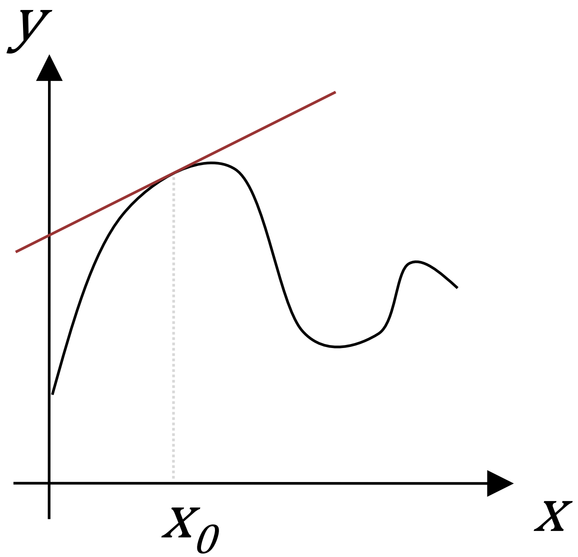
- Again, suppose: $x \sim \mu_x, \Sigma_x$

$$y = \cancel{Ax + b} \qquad y = f(x)$$

- Approach: approximate *f(x)* with Taylor expansion
  - What point should we approximate *f(x)* around?

Projecting Covariances (Nonlinear Case)

- First-order Taylor expansion
  - Let's review 1D case



$$y \approx \left.\frac{df}{dx}\right|_{x_0} (x - x_0) + f(x_0)$$

Projecting Covariances (Nonlinear Case)

- Generalized case:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, \ldots) \\ f_2(x_1, x_2, \ldots) \\ \ldots \end{bmatrix}$$

$$\mathbf{y} \approx \underbrace{\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \ldots \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix}}_{\text{“Jacobian”}} \begin{bmatrix} x_1 - x_{1_0} \\ x_2 - x_{2_0} \\ \ldots \end{bmatrix} + \begin{bmatrix} f_1(x_{1_0}, x_{2_0}) \\ f_2(x_{1_0}, x_{2_0}) \\ \ldots \end{bmatrix}$$

$$\mathbf{y} \approx J|_{\mathbf{x}_0}(\mathbf{x} - \mathbf{x}_0) + \mathbf{f}(\mathbf{x}_0)$$

Projecting Covariances (Nonlinear Case)

$$\mathbf{y} = \mathbf{f}(\mathbf{x})$$

$$\mathbf{y} \approx J|_{\mathbf{x}_0}(\mathbf{x} - \mathbf{x}_0) + \mathbf{f}(\mathbf{x}_0)$$

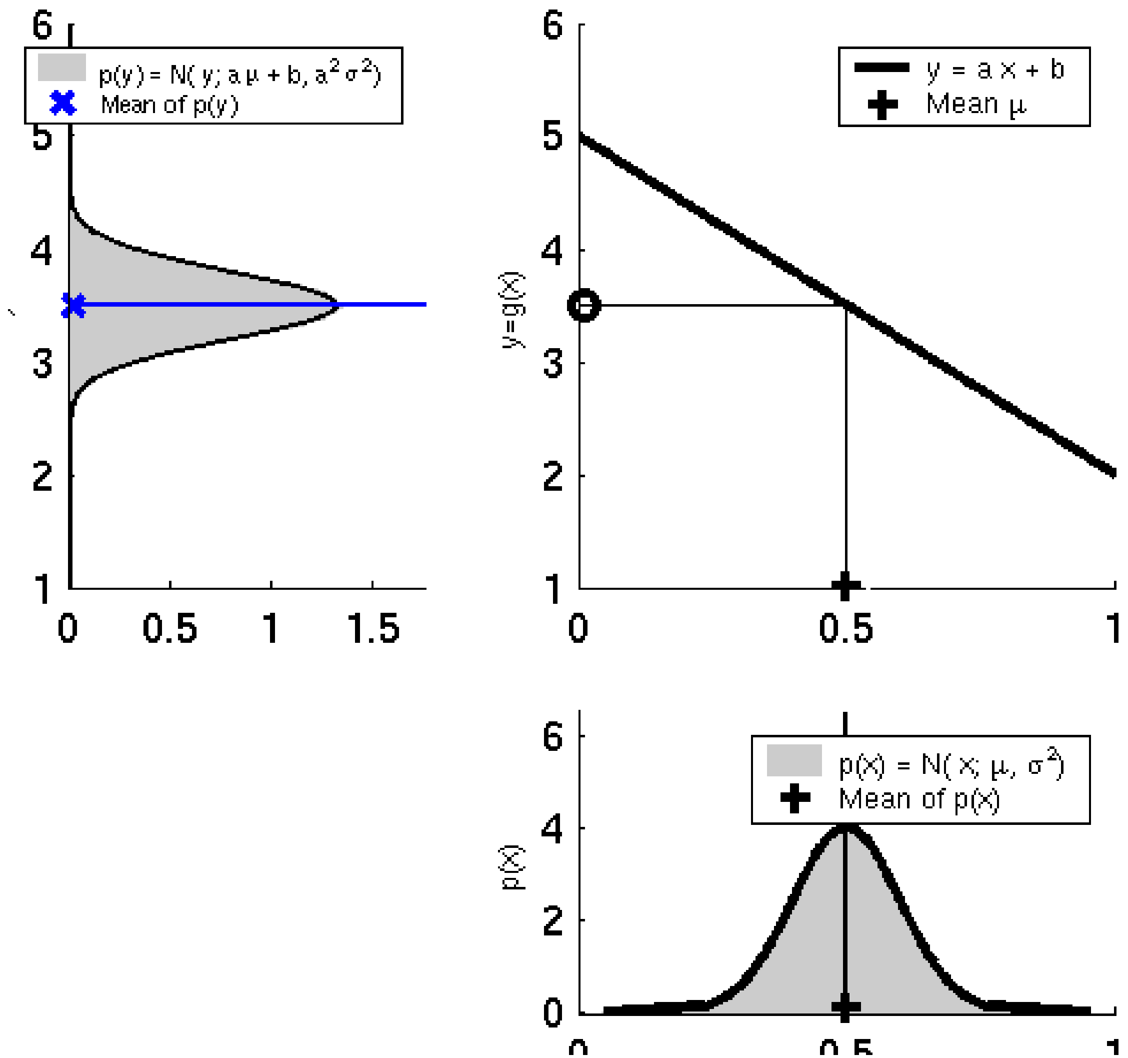
$$\mathbf{y} \approx \underbrace{J|_{\mathbf{x}_0}}_{\mathbf{A}}\mathbf{x} \underbrace{- J|_{\mathbf{x}_0}\mathbf{x}_0 + \mathbf{f}(\mathbf{x}_0)}_{\mathbf{b}}$$

$$y = Ax + b$$
$$\Sigma_y = A\Sigma_x A^T$$

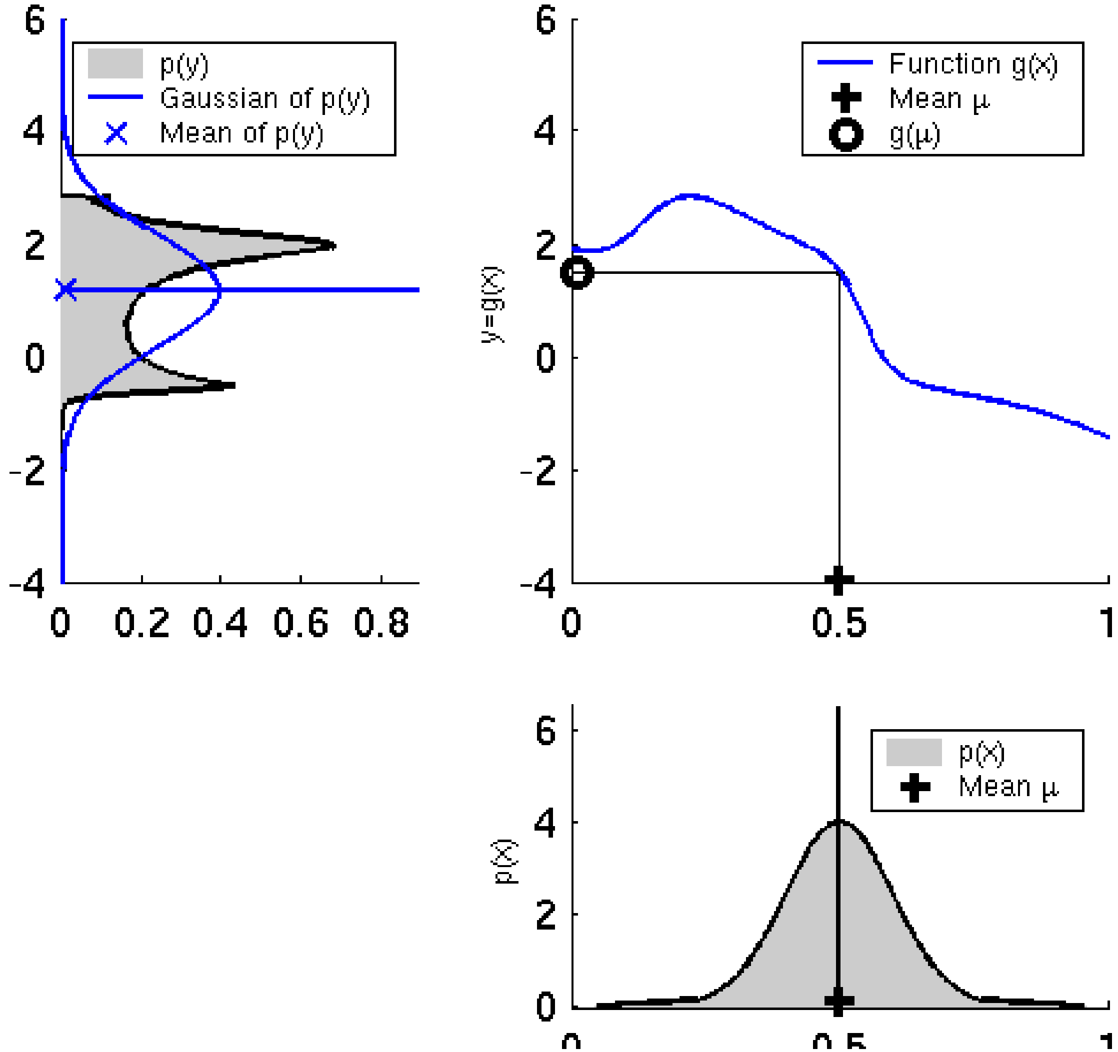
Non-linear case is reduced to linear case via first-order Taylor approximation. Expansion point $\mathbf{x}_0$ is typically taken as the mean.

What do we lose by dropping higher order terms?

Linearity Assumption Revisited



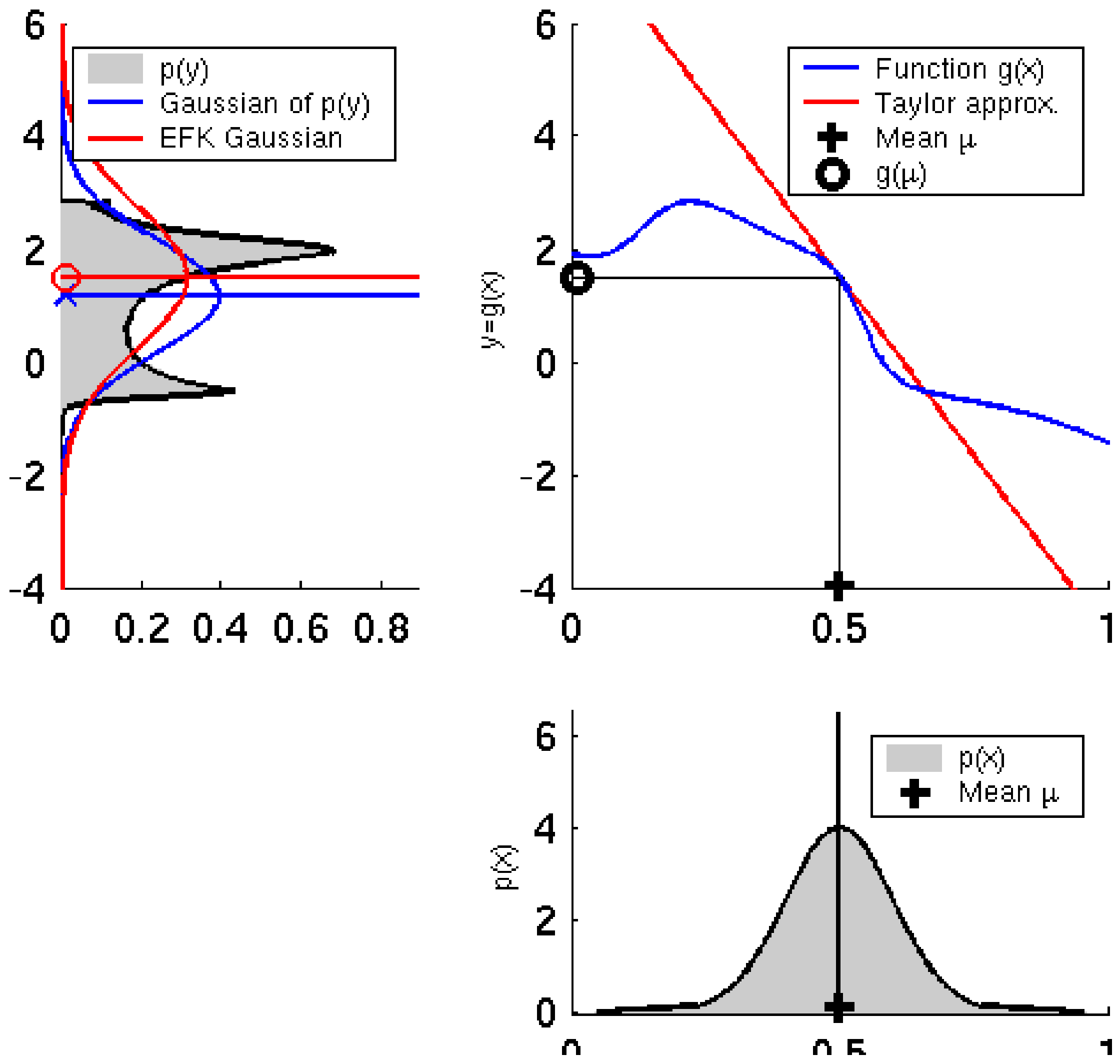
Nonlinear Function



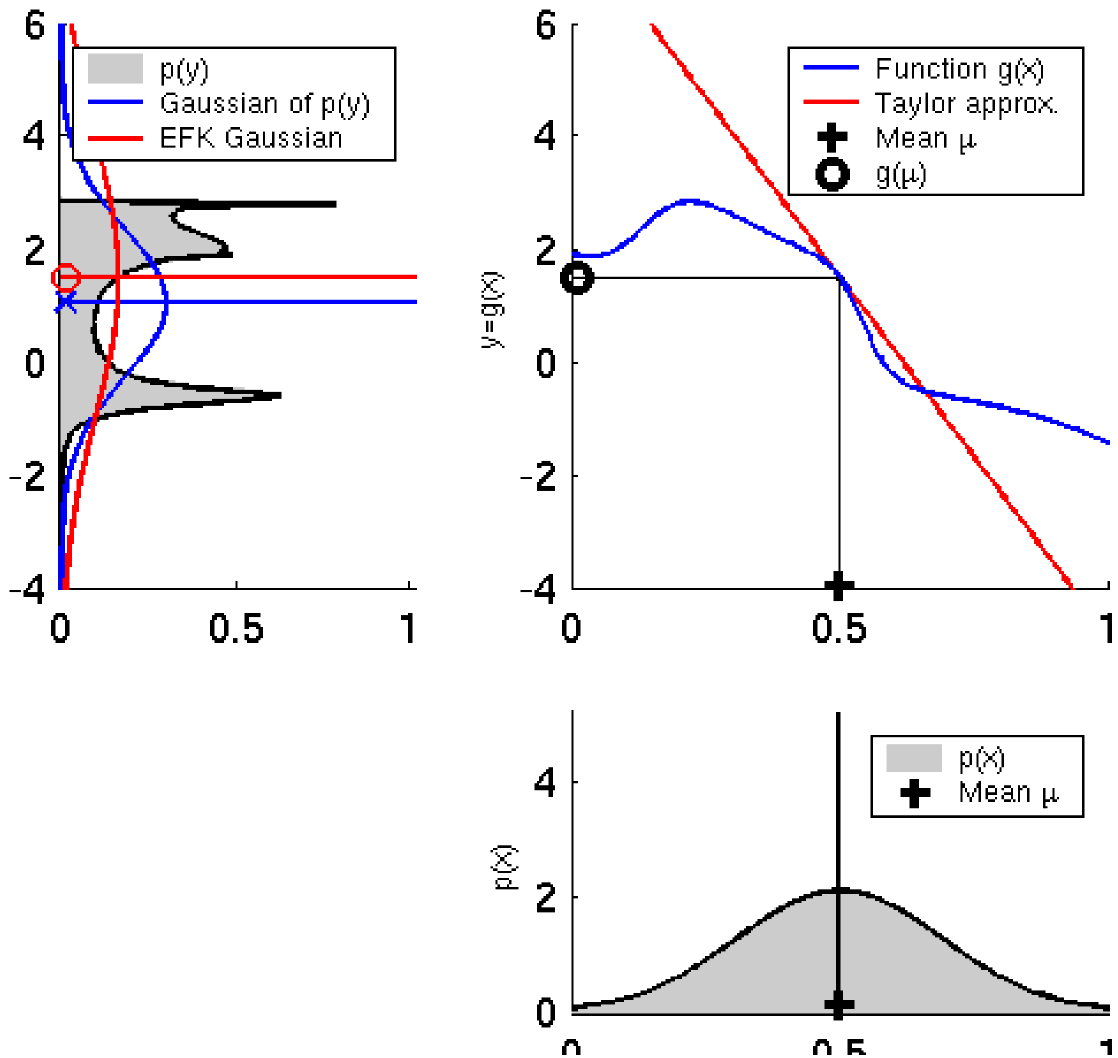
Nonlinear Gaussian Filters

- Approach 1: Extended Kalman Filter
  - Approximate the model!
  - Linearize our nonlinear plant and/or observation model(s) about the current mean and use the linear KF equations.

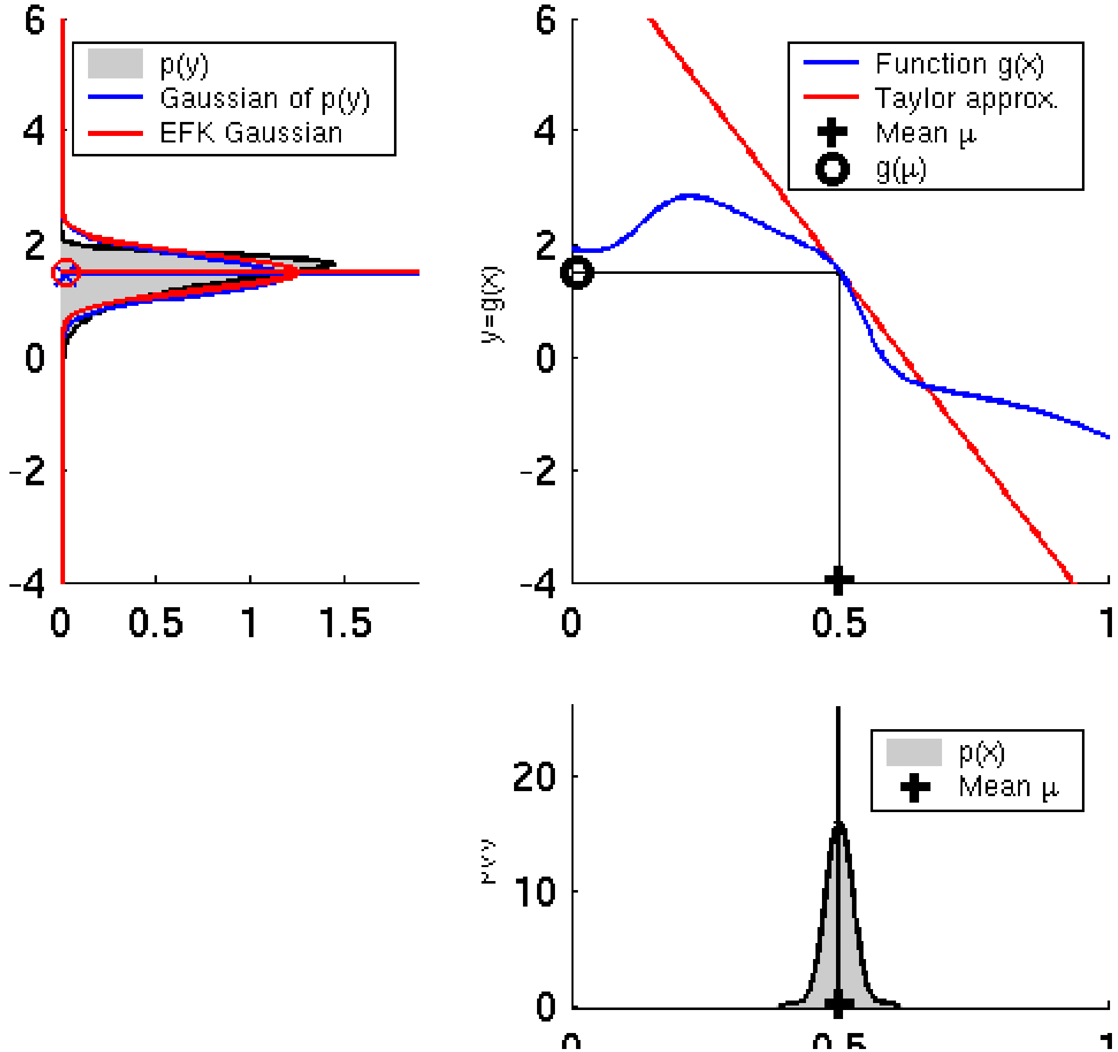
EKF Linearization via First Order Taylor Series



EKF Linearization: Large Variance



EKF Linearization: Narrow Variance



EKF Linearization: First Order Taylor Series Expansion

- Prediction:

$$g(\mathbf{u}_t, \mathbf{x}_{t-1}) \approx g(\mathbf{u}_t, \mu_{t-1}) + \frac{\partial g(\mathbf{u}_t, \mu_{t-1})}{\partial \mathbf{x}_{t-1}}(\mathbf{x}_{t-1} - \mu_{t-1})$$

$$g(\mathbf{u}_t, \mathbf{x}_{t-1}) \approx g(\mathbf{u}_t, \mu_{t-1}) + G_t(\mathbf{x}_{t-1} - \mu_{t-1})$$

- Correction:

$$h(\mathbf{x}_t) \approx h(\overline{\mu}_t) + \frac{\partial h(\overline{\mu}_t)}{\partial \mathbf{x}_t}(\mathbf{x}_t - \overline{\mu}_t)$$

$$h(\mathbf{x}_t) \approx h(\overline{\mu}_t) + H_t(\mathbf{x}_t - \overline{\mu}_t)$$

EKF Algorithm*

1. **Extended_Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t$):

2. Prediction:
3. $\bar{\mu}_t = g(\mathbf{u}_t, \mu_{t-1})$
4. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

5. Correction:
6. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t(\mathbf{z}_t - h(\bar{\mu}_t))$
8. $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$

9. Return μ_t, Σ_t

Linear KF

$\bar{\mu}_t = A_t \mu_{t-1} + B_t \mathbf{u}_t$

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

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

$\mu_t = \bar{\mu}_t + K_t(\mathbf{z}_t - C_t \bar{\mu}_t)$

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

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial \mathbf{x}_t} \qquad G_t = \frac{\partial g(\mathbf{u}_t, \mu_{t-1})}{\partial \mathbf{x}_{t-1}}$$

* The form shown assumes additive process and observation model noise

EKF Summary

- Highly efficient: Polynomial in measurement dimensionality k and state dimensionality n:
  $$O(k^{2.376} + kn^2)$$
- **Not optimal**!
- Can diverge if nonlinearities are large!
- Can work surprisingly well even when all assumptions are violated!

KF, EKF and UKF

- Kalman filter requires linear models
- EKF linearizes via Taylor expansion

Is there a better way to linearize?

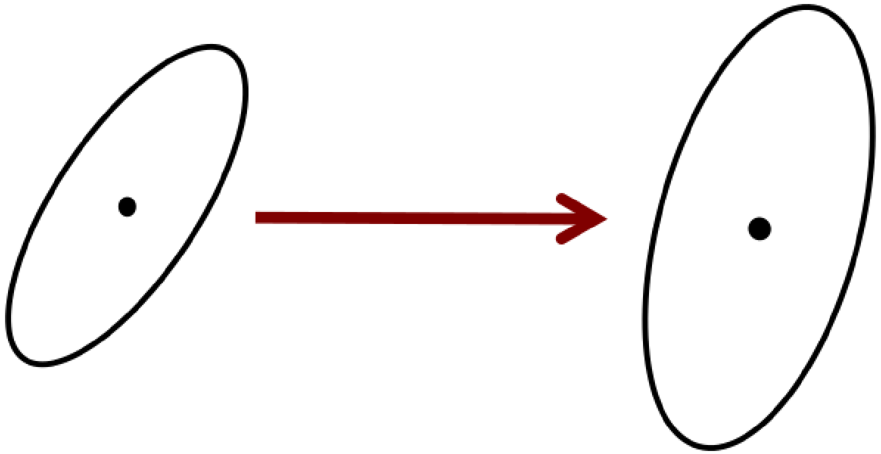
Unscented Transform

⬇

Unscented Kalman Filter (UKF)

Courtesy: Cyrill Stachniss

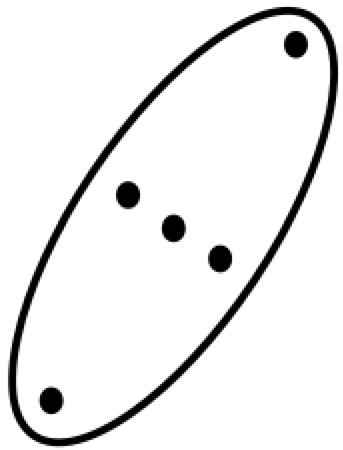
Taylor Approximation (EKF)



Linearization of the non-linear function through Taylor expansion

Courtesy: Cyrill Stachniss

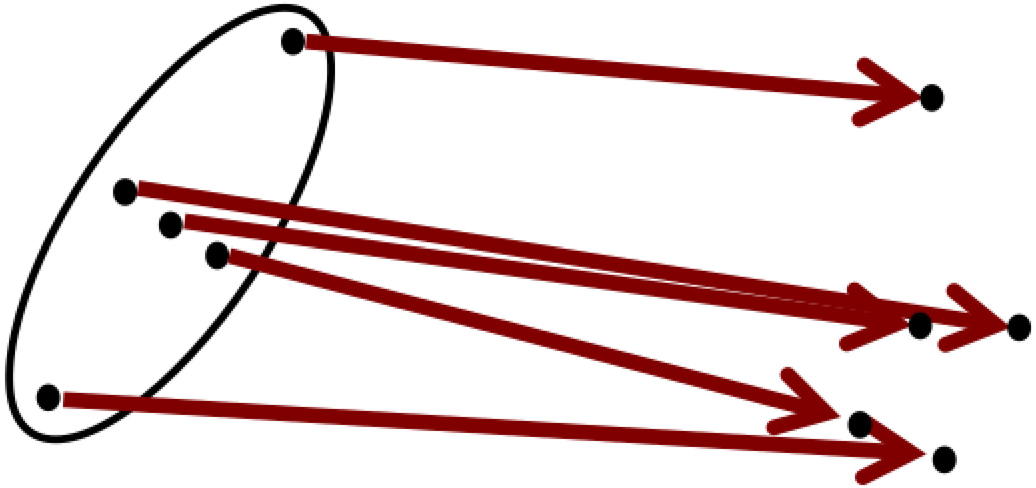
Unscented Transform



Compute a set of (so-called) sigma points

Courtesy: Cyrill Stachniss

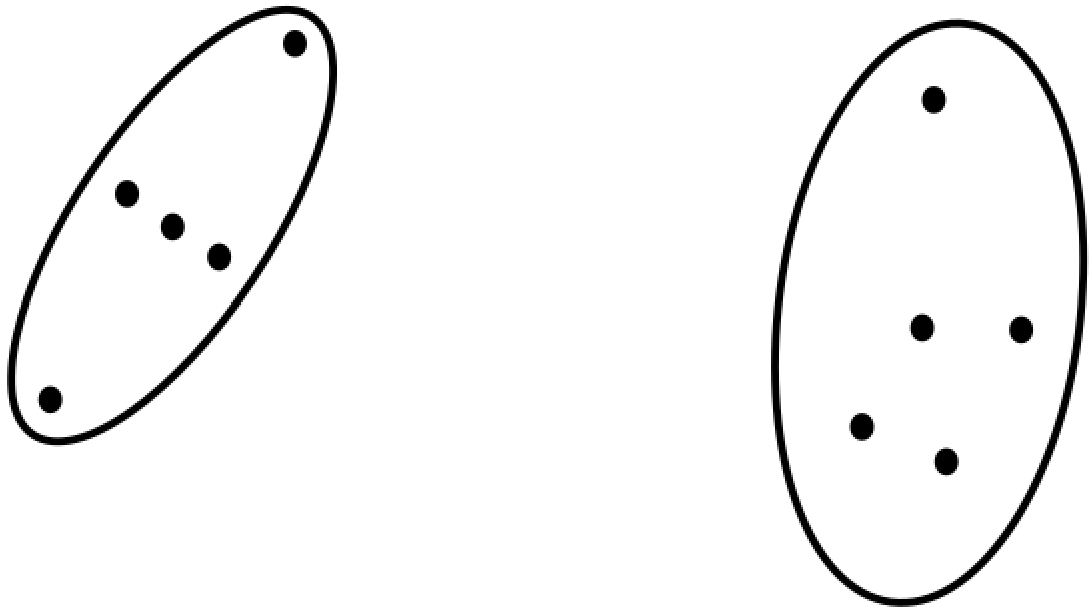
Unscented Transform



Transform each sigma point
through the non-linear function

Courtesy: Cyrill Stachniss

Unscented Transform



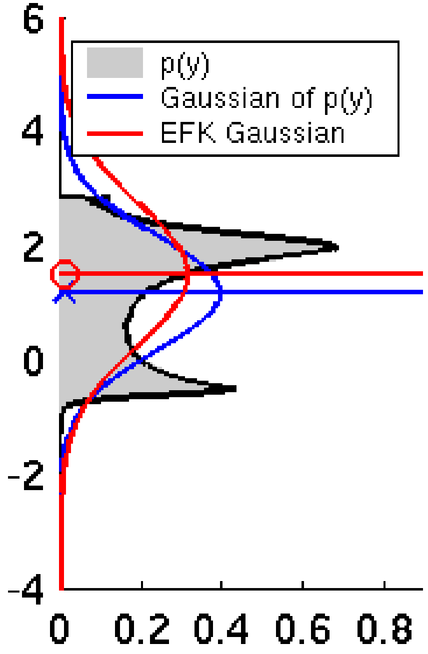
Compute Gaussian from the transformed and weighted sigma points

Courtesy: Cyrill Stachniss

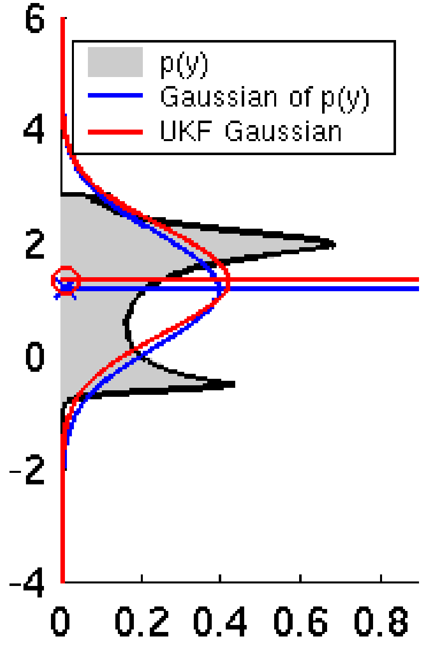
Nonlinear Gaussian Filters

- Approach 2: Unscented Kalman Filter
  - Approximate the PDF!
  - Use the full nonlinear plant and observation models and recompute 1st and 2nd order statistics.

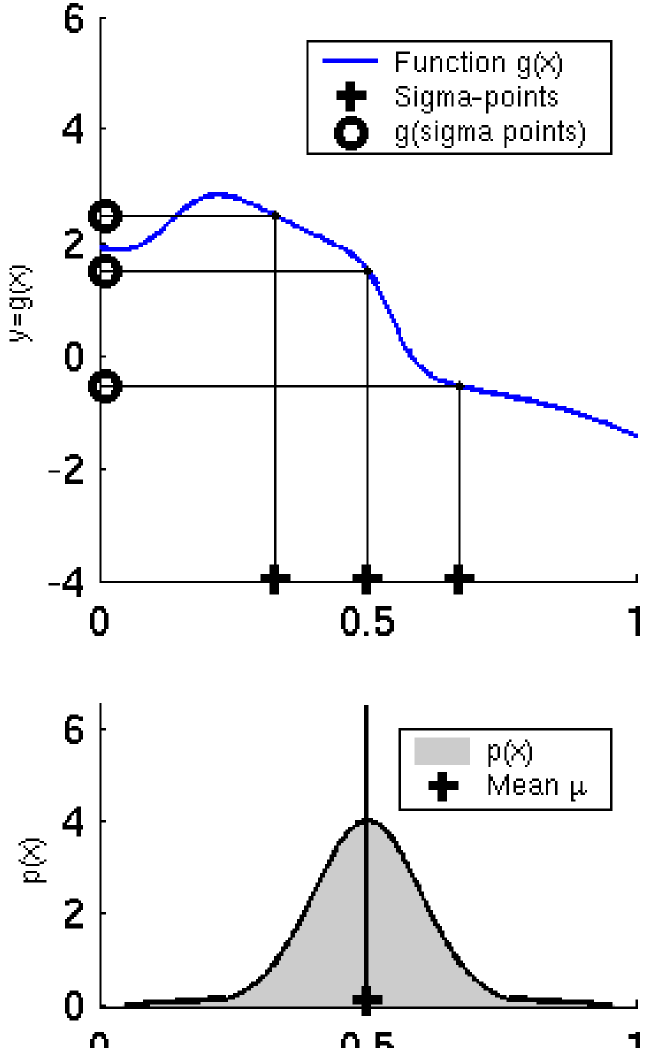
UKF Linearization via Unscented Transform



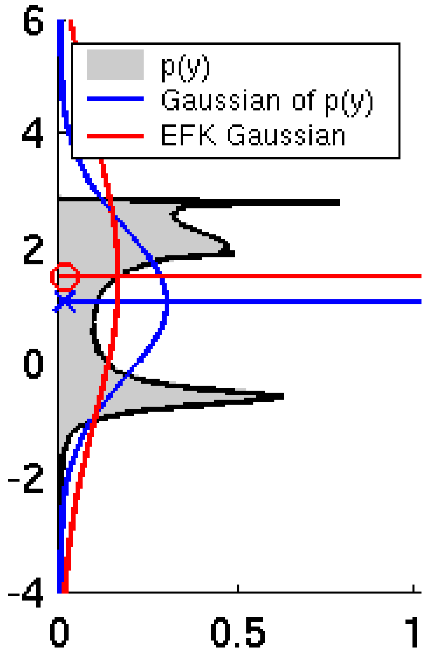
EKF



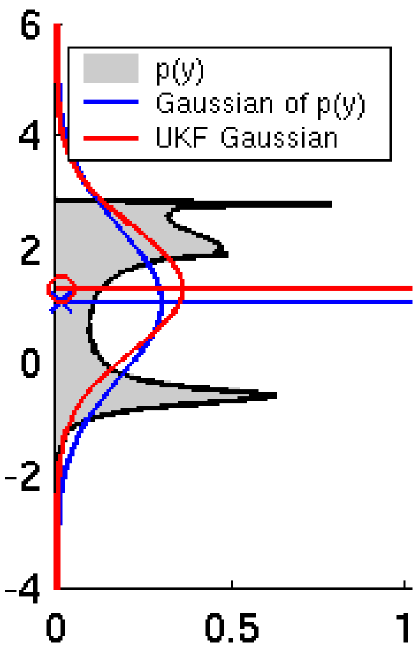
UKF



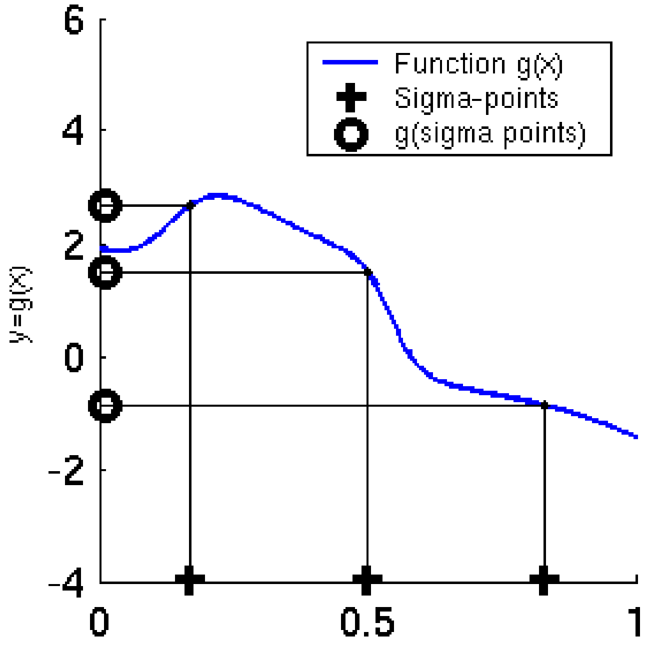
UKF Sigma-Point Estimate: Large Variance

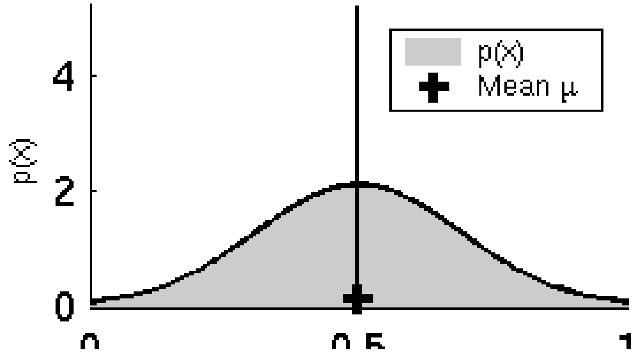


EKF

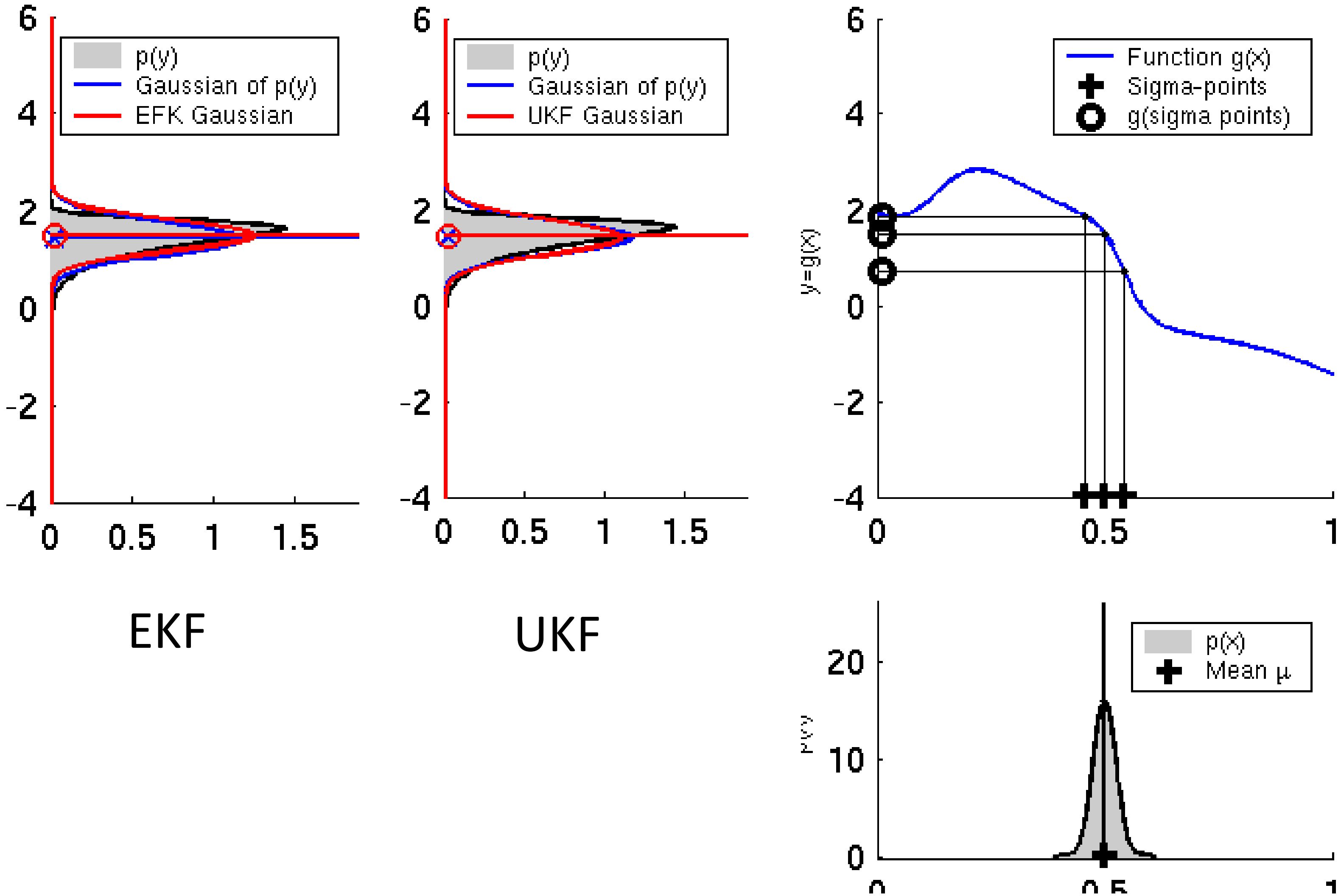


UKF





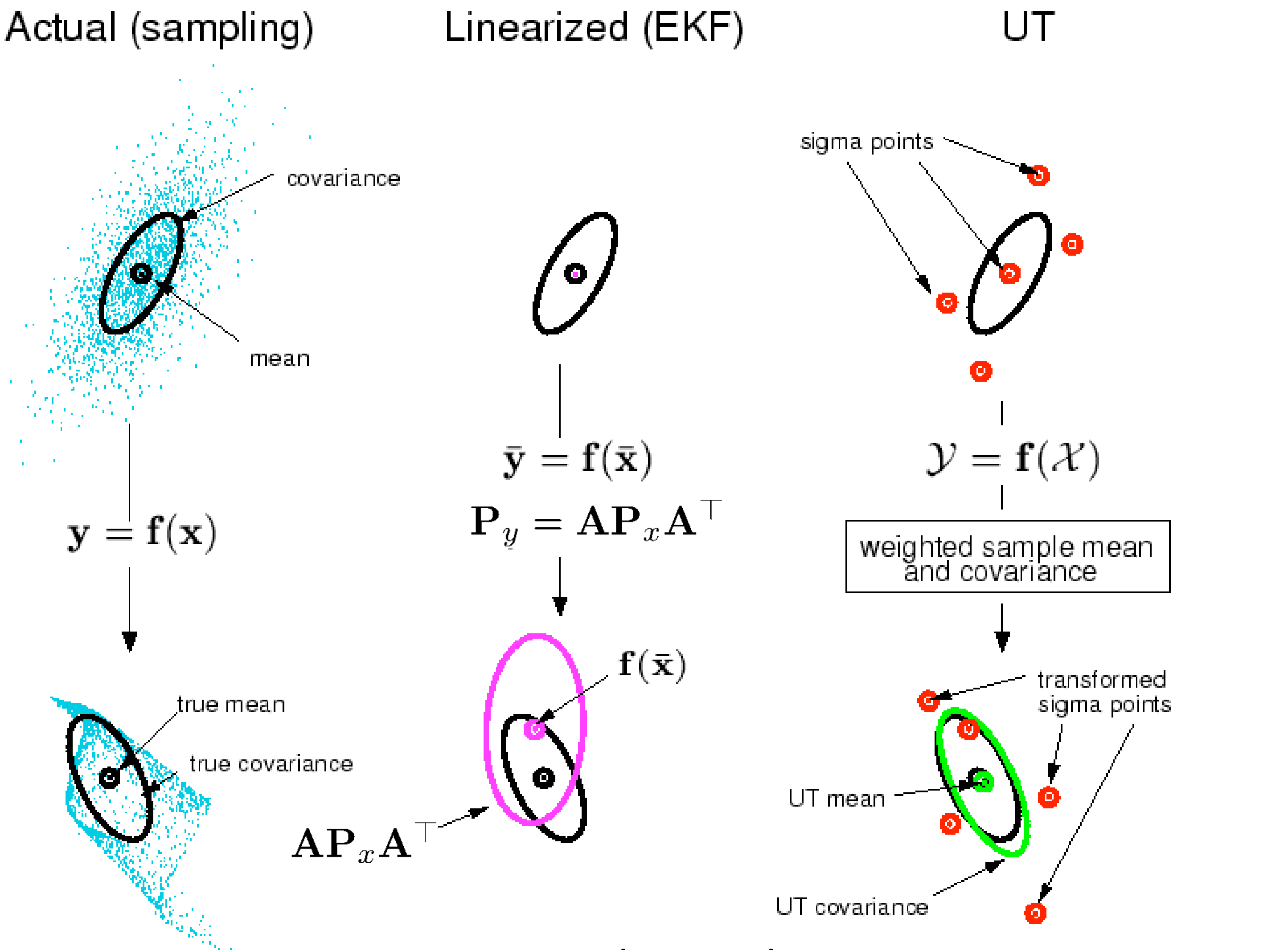
UKF Sigma-Point Estimate: Narrow Variance



EKF

UKF

UKF vs. EKF



Courtesy: E.A. Wan and R. van der Merwe

Unscented Transform Overview

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

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

Courtesy: Cyrill Stachniss

Sigma Points

- How to choose the sigma points?
- How to set the weights?

Courtesy: Cyrill Stachniss

Sigma Points Properties

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

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

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

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

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

Courtesy: Cyrill Stachniss

Sigma Points

- Choosing the sigma points

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

First sigma point is the mean

$$\boldsymbol{\mathcal{X}}^{[i]} = \boldsymbol{\mu} + \left(\sqrt{(n+\lambda)\,\Sigma}\right)_i \quad \text{for } i = 1, \ldots, n$$

Courtesy: Cyrill Stachniss

Sigma Points

- Choosing the sigma points

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

matrix
square root

column vector

dimensionality

scaling parameter

Courtesy: Cyrill Stachniss

Real Symmetric Matrix Square Root

- Defined as $S \text{ with } \Sigma = SS^\top$
- Computed via diagonalization

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

Courtesy: Cyrill Stachniss

Real Symmetric Matrix Square Root

- Thus, we can define

$$S \quad = \underbrace{V \begin{pmatrix} \sqrt{d_{11}} & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \sqrt{d_{nn}} \end{pmatrix}}_{\mathbf{P}_y = \mathbf{A}\mathbf{P}_x\mathbf{A}^\top} V^{-1}$$

- so that

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

- S and Σ have the same Eigenvectors

Courtesy: Cyrill Stachniss

Cholesky Matrix Square Root

- Alternative definition of the matrix square root

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

- Result of the Cholesky decomposition
- Numerically stable solution
- Often used in UKF implementations

- Actually, any such square root factorization is ok, e.g., could use factorization

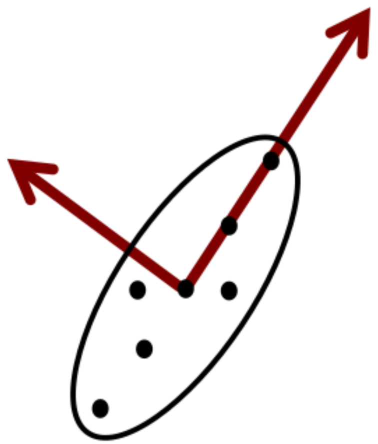
$$\Sigma = AA^\top \qquad \text{where } A = VD^{\frac{1}{2}}$$

Sigma Points and Eigenvectors

- Sigma points **can** but **do not have to** lie on the main axes of Σ

$$\boldsymbol{\mathcal{X}}^{[i]} = \boldsymbol{\mu} + \left(\sqrt{(n+\lambda)\,\Sigma}\right)_i \quad \text{for } i = 1, \ldots, n$$

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



Courtesy: Cyrill Stachniss

Sigma Points Example

Sigma =

1.1335	1.9544
1.9544	5.5336

SigmaL =

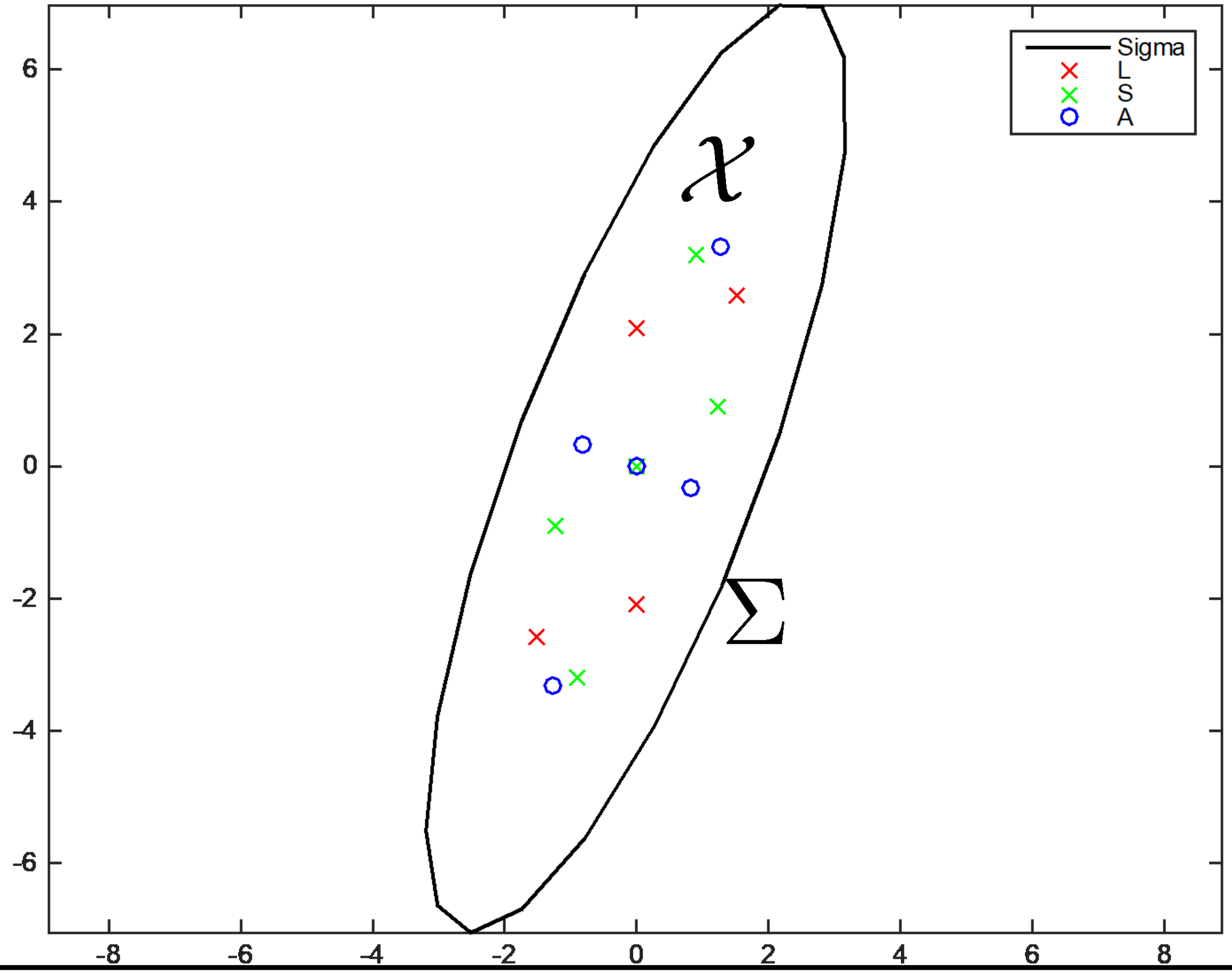
1.1335	1.9544
1.9544	5.5336

SigmaS =

1.1335	1.9544
1.9544	5.5336

SigmaA =

1.1335	1.9544
1.9544	5.5336



Sigma Point 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

Courtesy: Cyrill Stachniss

Recover the Gaussian

- Compute Gaussian from weighted and transformed points

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

$$\Sigma' = \sum_{i=0}^{2n} w_c^{[i]} \, (g(\boldsymbol{\mathcal{X}}^{[i]}) - \boldsymbol{\mu}')(g(\boldsymbol{\mathcal{X}}^{[i]}) - \boldsymbol{\mu}')^\top$$

Courtesy: Cyrill Stachniss

(Scaled) Unscented Transform

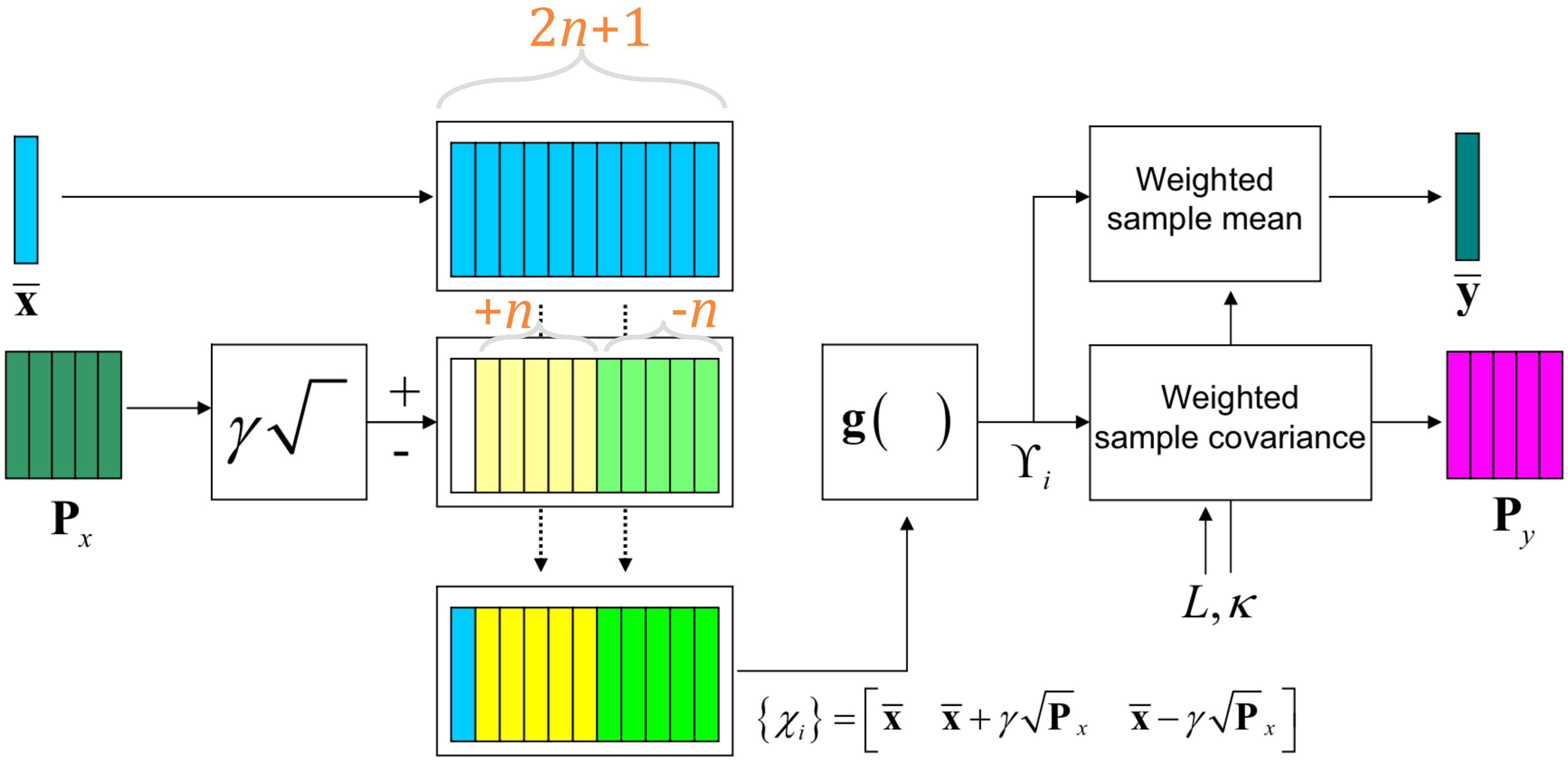


Figure 3.2: Schematic diagram of the unscented transformation.

Source: Van Der Merwe, Thesis

Unscented Transform Summary

- Sigma points

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

- Weights

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

Courtesy: Cyrill Stachniss

SUT Parameters

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

$$\left.\begin{array}{lll} \kappa & \geq & 0 \\ \alpha & \in & (0, 1] \end{array}\right\} \text{Influence how far the sigma points are away from the mean}$$

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

$$\beta = 2 \quad \text{Optimal choice for Gaussians}$$

Courtesy: Cyrill Stachniss

SUT Parameters

- Choose $\kappa \geq 0$
  - to guarantee positive semi-definiteness of the covariance matrix. The specific value of κ is not critical though, so a good default choice is $\kappa = 0$.

- Choose $0 < \alpha \leq 1$
  - to control the "size" of the sigma-point distribution and should be chosen to avoid sampling non-local effects when the nonlinearities are strong; a default choice is $\alpha = 1$.

- Choose $\beta \geq 0$
  - to incorporate knowledge of the higher-order moments of the distribution. For example, for a Gaussian prior the optimal choice is $\beta = 2$.

- The original (un-scaled) UT transform is equivalent to:
  - SUT with $\alpha = 1, \beta = 0$

(Scaled) Unscented Transform

Sigma points

$$\chi^0 = \mu$$

$$\chi^i = \mu \pm \left(\sqrt{(n+\lambda)\Sigma}\right)_i$$

Weights

$$w_m^0 = \frac{\lambda}{n+\lambda} \qquad w_c^0 = \frac{\lambda}{n+\lambda} + (1-\alpha^2+\beta)$$

$$w_m^i = w_c^i = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1,\dots,2n$$

Pass sigma points through nonlinear function

$$\psi^i = g(\chi^i)$$

Recover mean and covariance

$$\mu' = \sum_{i=0}^{2n} w_m^i \psi^i$$

$$\Sigma' = \sum_{i=0}^{2n} w_c^i (\psi^i - \mu')(\psi^i - \mu')^T$$

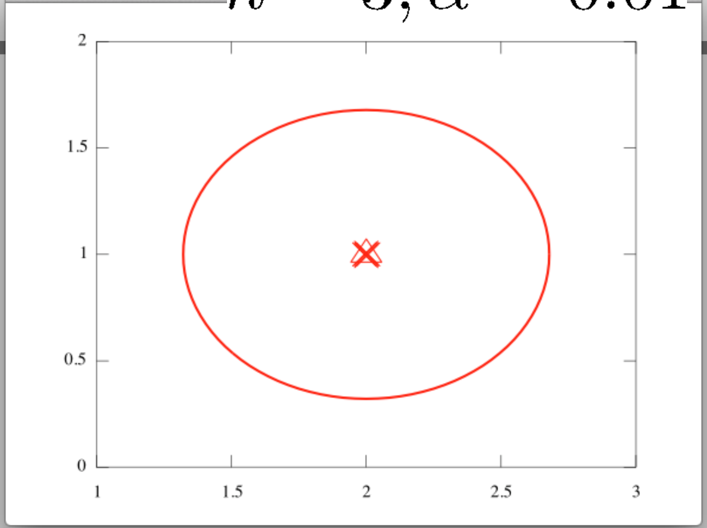
$$\lambda = \alpha^2(n+\kappa) - n$$

$0 < \alpha \leq 1$	Sigma point scaling
$\beta \geq 0$	Higher-order moment matching
$\kappa \geq 0$	Scalar tuning parameter

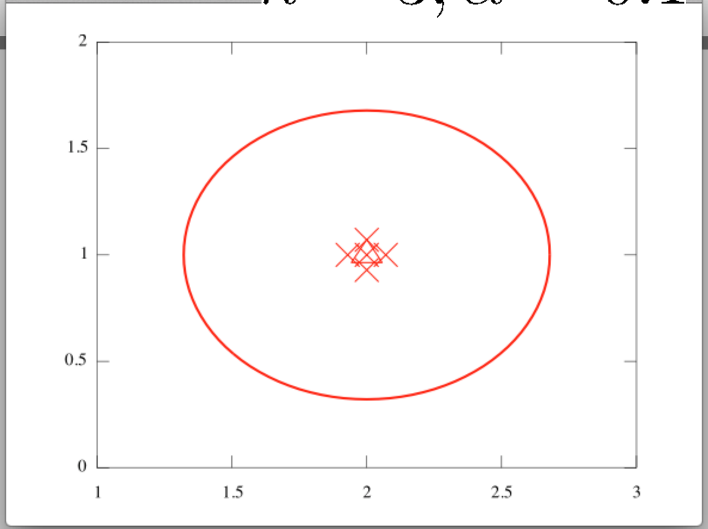
Courtesy: Cyrill Stachniss

Examples

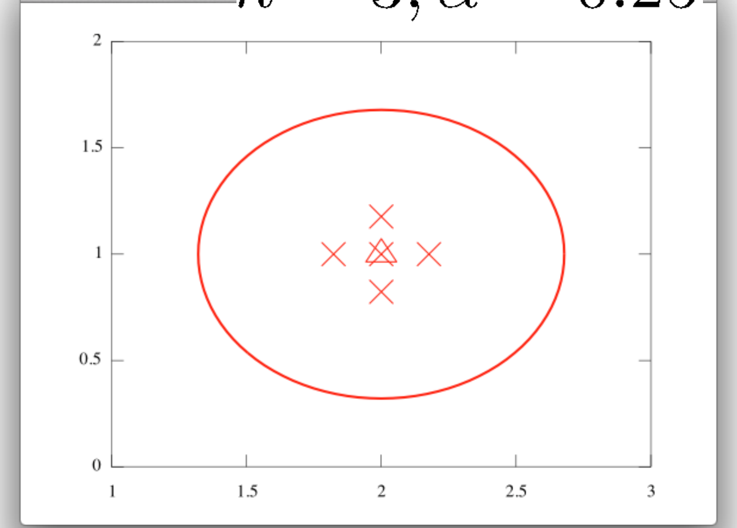
$\kappa = 3, \alpha = 0.01$



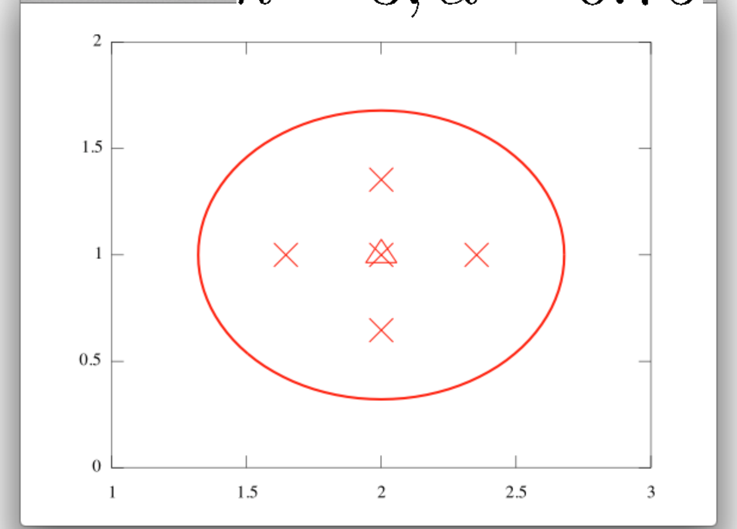
$\kappa = 3, \alpha = 0.1$



$\kappa = 3, \alpha = 0.25$

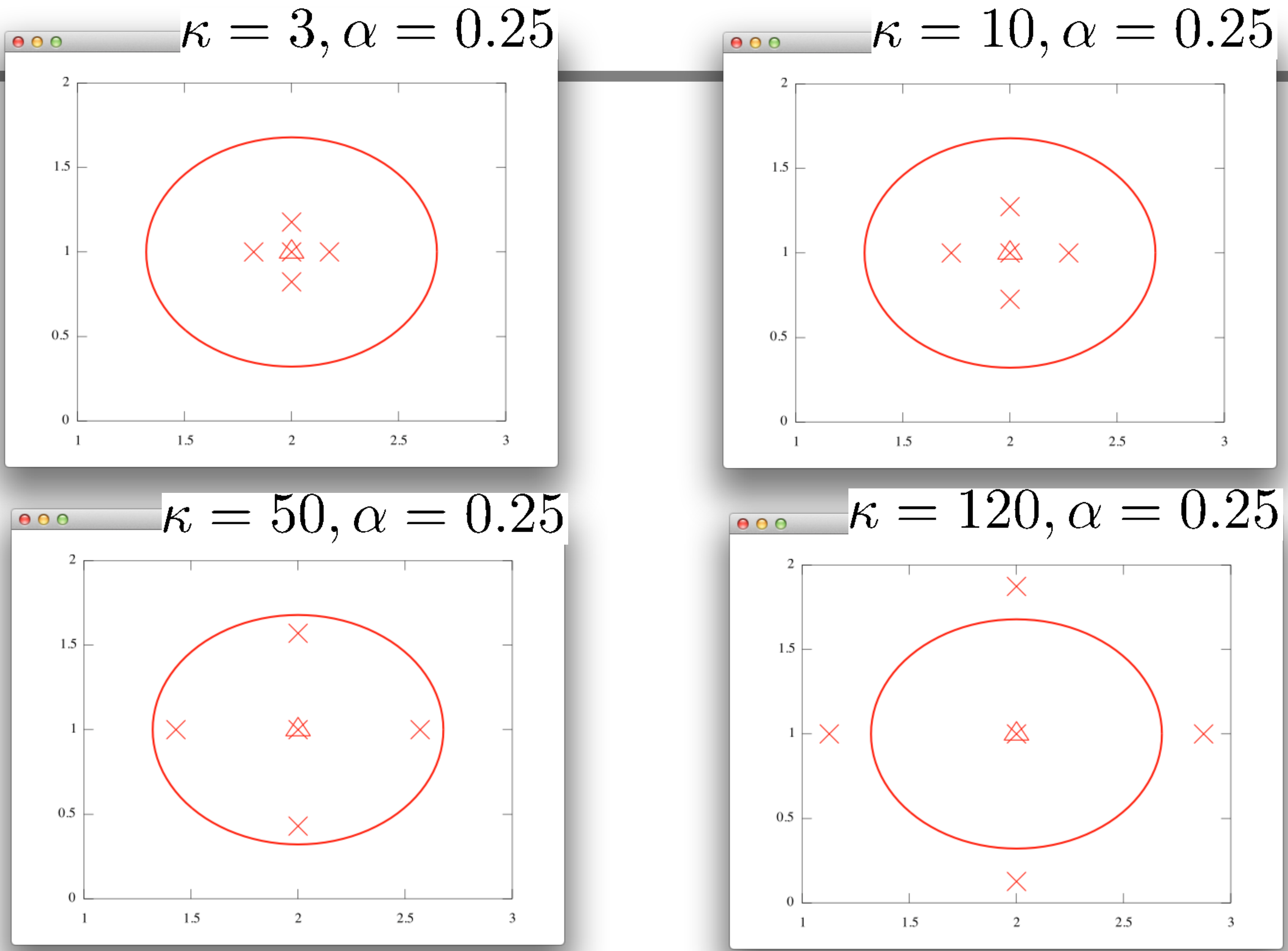


$\kappa = 3, \alpha = 0.75$



Courtesy: Cyrill Stachniss

Examples



- How to apply UT to estimation??

UKF (Unscented Kalman Filter)

UKF Uses the Kalman Update

- KF is the Best Linear Unbiased Estimator (BLUE)
  - i.e., if we restrict our estimator to the class of linear estimators, then the KF is the best *linear* MMSE estimator*
  - What should A and $\mathbf{b}$ be?

$$\hat{\mathbf{x}} = A\mathbf{z} + \mathbf{b} \longleftarrow \text{Affine function of } \mathbf{z}$$

* Note: a nonlinear estimator could do <u>*better*</u>*!!*

To derive, we want our error to be orthogonal to the measurement space

- Estimator

$$\hat{\mathbf{x}} = A\mathbf{z} + \mathbf{b}$$

- Error

$$\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$$

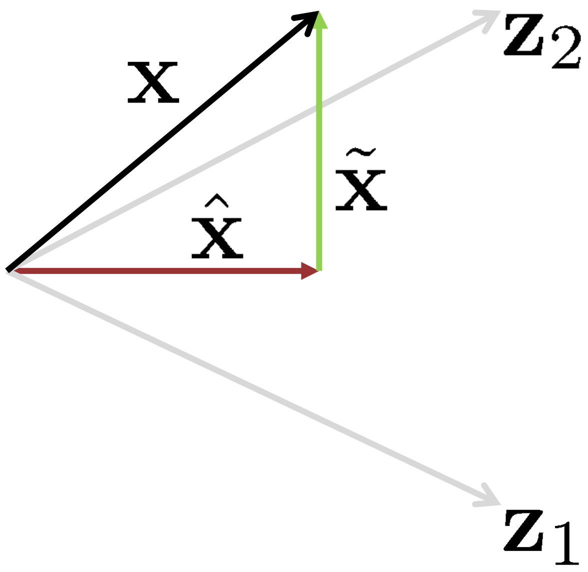
- Unbiased

$$E[\tilde{\mathbf{x}}] = \mathbf{0}$$

- Orthogonal

$$\tilde{\mathbf{x}} \perp \mathbf{z}$$

$$E[\tilde{\mathbf{x}}\mathbf{z}^\top] = 0$$



Best Linear Unbiased Estimator (BLUE)

- Unbiased $\Rightarrow \mathbf{b} = \mu_x - A\mu_z$
- Orthogonal $\Rightarrow A = \Sigma_{\mathbf{xz}}\Sigma_{\mathbf{zz}}^{-1}$
- Estimator
- Matrix MSE

$$\boxed{\begin{aligned}\hat{\mathbf{x}} &= \mu_x + \Sigma_{\mathbf{xz}}\Sigma_{\mathbf{zz}}^{-1}(\mathbf{z} - \mu_z) \\ E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top] &= \Sigma_{\mathbf{xx}} - \Sigma_{\mathbf{xz}}\Sigma_{\mathbf{zz}}^{-1}\Sigma_{\mathbf{zx}}\end{aligned}}$$

- Remarks
  - The ***best estimator*** (in the MMSE sense) for *Gaussian Random variables* is identical to
    - The *best linear unbiased estimator* for *arbitrarily distributed* random variables with the *same first- and second-order moments.*

EKF Algorithm*

1: $\textbf{Extended\_Kalman\_filter}(\boldsymbol{\mu}_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$**:**

2: $\bar{\boldsymbol{\mu}}_t = g(\mathbf{u}_t, \boldsymbol{\mu}_{t-1})$

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

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

5: $\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + K_t(\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t))$

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

7: $return\ \boldsymbol{\mu}_t, \Sigma_t$

* The form shown assumes additive process and observation model noise

Courtesy: Cyrill Stachniss

EKF to UKF – Prediction

1: ~~Extended~~ Unscented Kalman_filter($\boldsymbol{\mu}_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t$):

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

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

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

5: $\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + K_t(\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t))$

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

7: return $\boldsymbol{\mu}_t, \Sigma_t$

Courtesy: Cyrill Stachniss

UKF Algorithm – Prediction*

1: $\textbf{Unscented\_Kalman\_filter}(\boldsymbol{\mu}_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$**:**

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

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

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

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

* The form shown assumes additive process and observation model noise

Courtesy: Cyrill Stachniss

EKF to UKF – Correction

1: ~~Extended~~ Unscented_Kalman_filter($\boldsymbol{\mu}_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t$):

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

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

use sigma point propagation for the expected observation and Kalman gain

5: $\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + K_t(\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t))$

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

7: *return* $\boldsymbol{\mu}_t, \Sigma_t$

Courtesy: Cyrill Stachniss

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

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

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

* The form shown assumes additive process and observation model noise

Courtesy: Cyrill Stachniss

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

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

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

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

(from BLUE)

* The form shown assumes additive process and observation model noise

UKF Algorithm – Correction (2)

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

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

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

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

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

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

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

$$\Sigma_t = \bar{\Sigma}_t - K_t \; S_t \; K_t^\top$$

$$\text{return } \mu_t, \Sigma_t$$

Courtesy: Cyrill Stachniss

UKF

This version of the algorithm implicitly assumes ***additive*** zero-mean process and observation noise

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

$$\mathcal{X}_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}})$$

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

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

Take care with means of circular quantities

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

$$\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \gamma\sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \gamma\sqrt{\bar{\Sigma}_t})$$

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

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

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

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

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

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

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

return μ_t, Σ_t

Means of Circular Quantities

- Trick is to map angles θ_i to the unit circle

- Take arithmetic mean of Cartesian quantities

$$\overline{\cos} = \sum_{i=0}^{2N} \cos(\theta_i) w_m^{[i]} \quad \overline{\sin} = \sum_{i=0}^{2N} \sin(\theta_i) w_m^{[i]}$$

- Map back to corresponding "average" angle*

$$\bar{\theta} = \text{atan2}(\overline{\sin}, \overline{\cos})$$

*Note: poor approx when θ_i is widely distributed

Similarly

- Map angular differences, such as

$$(\mathcal{X}^{[i]} - \mu) \quad \text{to} \quad [-\pi, \pi]$$

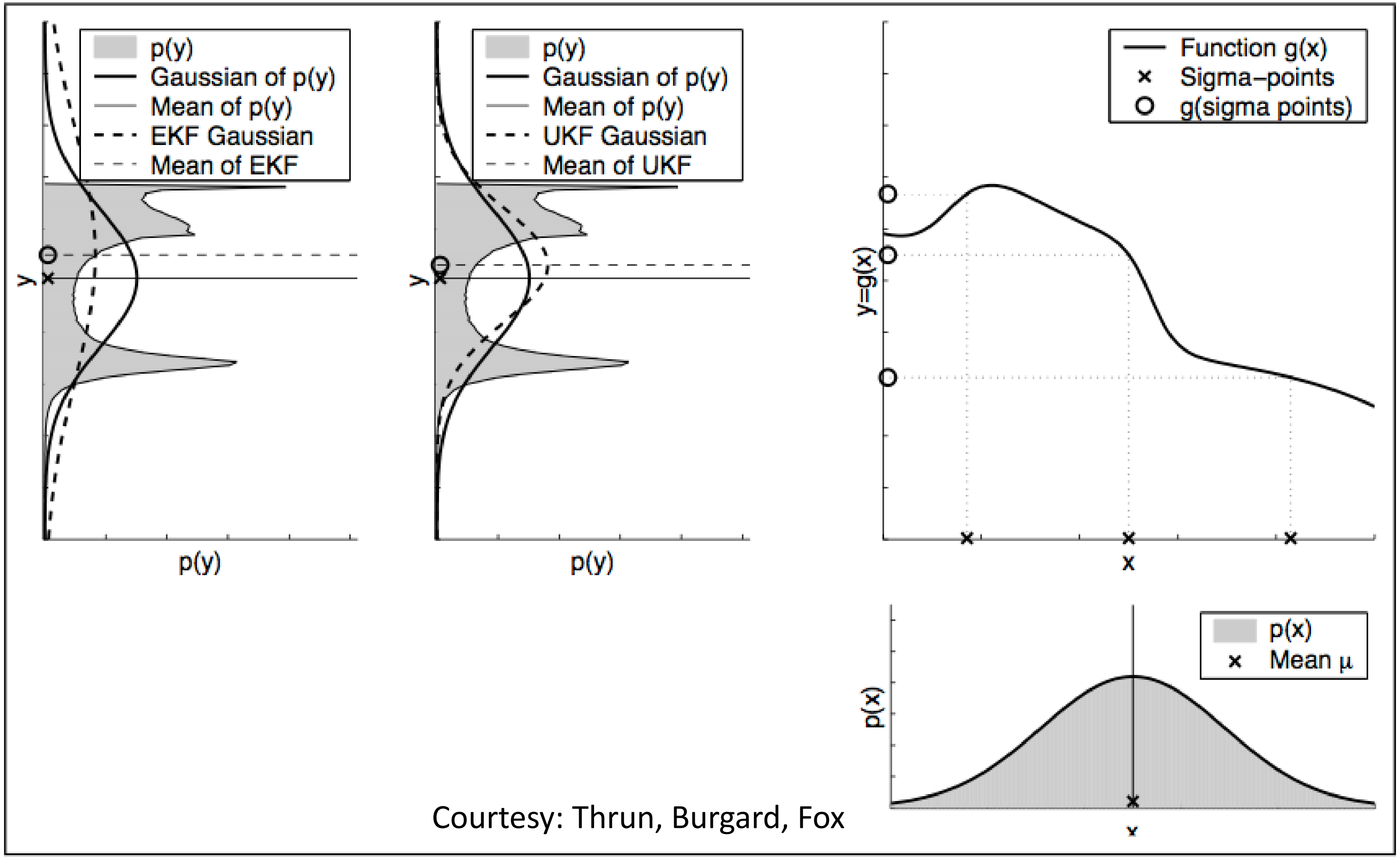
when computing innovation and covariance expressions, e.g.:

$$\Sigma_{xx} = \sum_{i=0}^{2N} w_c^{[i]} (\boldsymbol{\mathcal{X}}^{[i]} - \boldsymbol{\mu}_x)(\boldsymbol{\mathcal{X}}^{[i]} - \boldsymbol{\mu}_x)^\top$$

$$\Sigma_{xz} = \sum_{i=0}^{2N} w_c^{[i]} (\boldsymbol{\mathcal{X}}^{[i]} - \boldsymbol{\mu}_x)(\boldsymbol{\mathcal{Z}}^{[i]} - \boldsymbol{\mu}_z)^\top$$

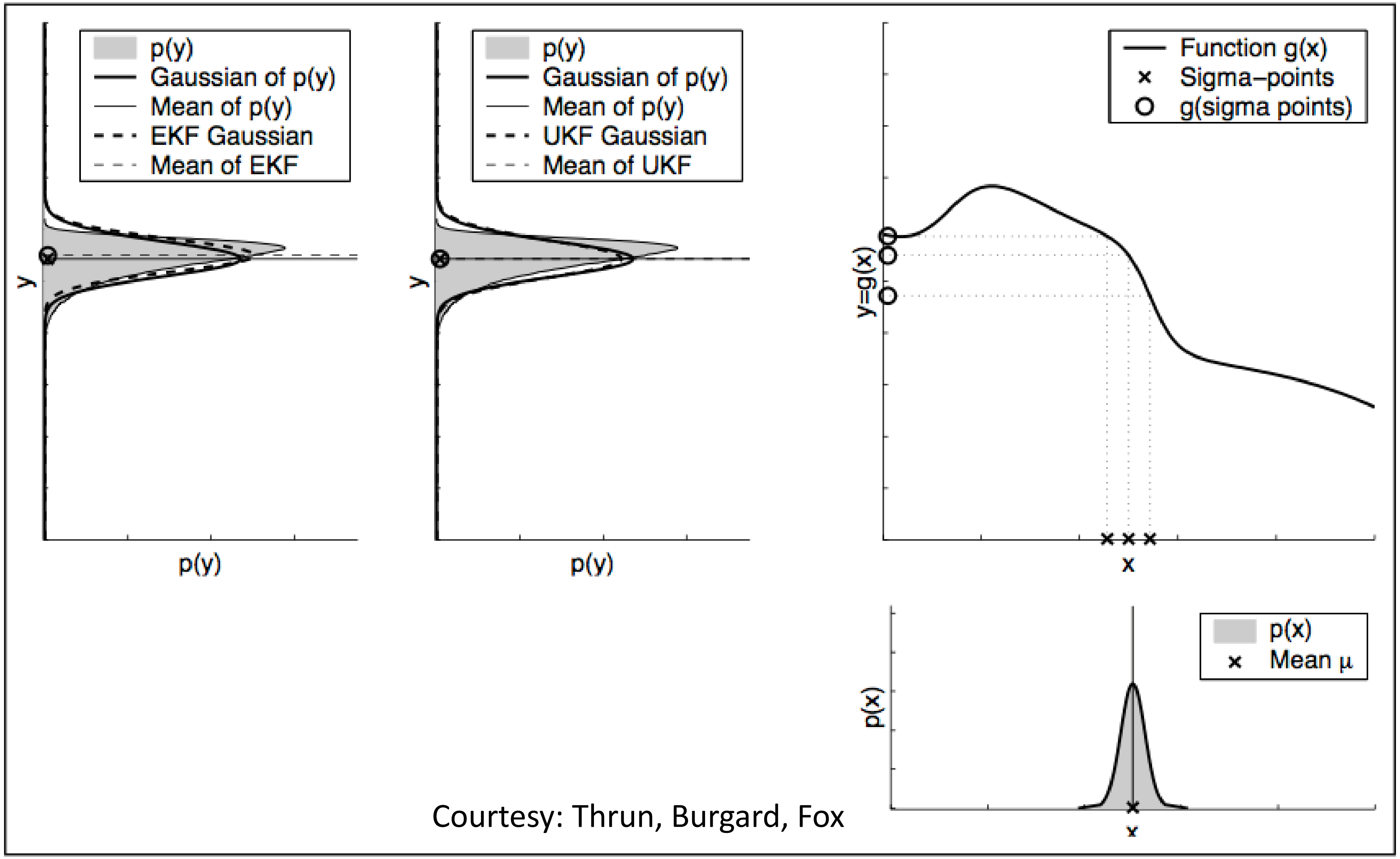
i.e. 2π-0 = 0 !!!

UKF vs. EKF



Courtesy: Thrun, Burgard, Fox

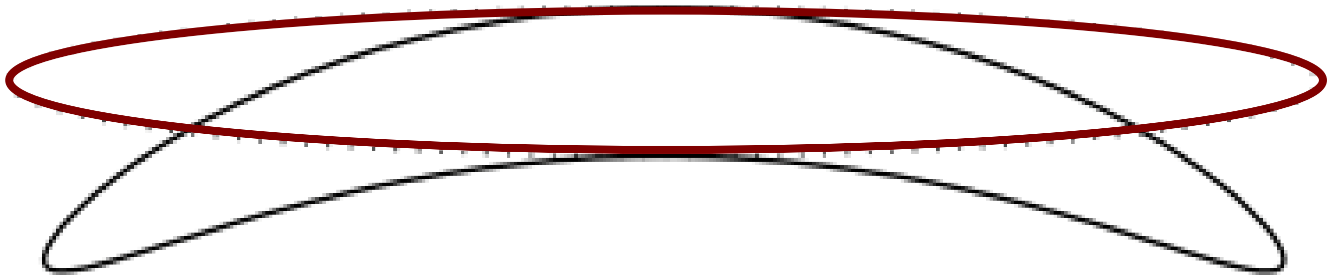
UKF vs. EKF (Small Covariance)



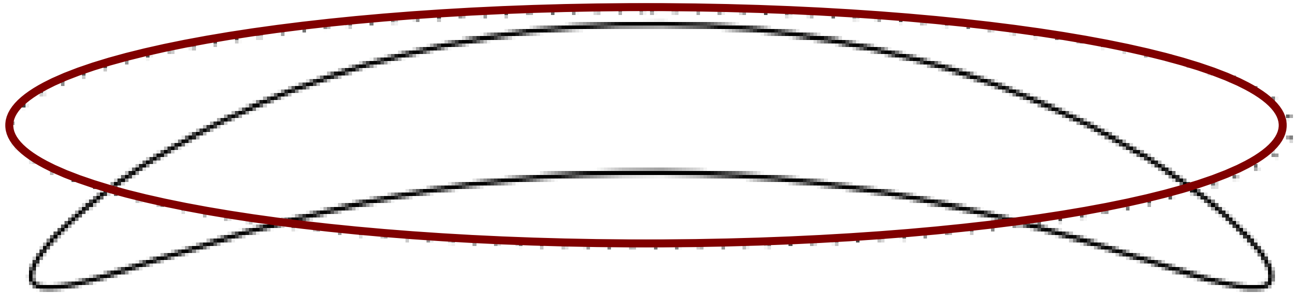
Courtesy: Thrun, Burgard, Fox

UKF vs. EKF – Banana Shape

EKF approximation



UKF approximation



Courtesy: Cyrill Stachniss

UKF Summary

- **Highly efficient**: Same complexity as EKF, with a constant factor slower in typical practical applications
- **Better linearization than EKF**: Accurate in first two derivatives* of Taylor expansion (EKF only first term)
- **Derivative-free**: No Jacobians needed
- **Still not optimal**!

* Accurate in first three derivatives if Gaussian prior

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

Courtesy: Cyrill Stachniss

Literature

Unscented Transform and UKF

- Thrun et al.: “Probabilistic Robotics”, Chapter 3.4
- “A New Extension of the Kalman Filter to Nonlinear Systems” by Julier and Uhlmann, 1995
- “Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models”, PhD Thesis, Rudolph van der Merwe, 2004

Courtesy: Cyrill Stachniss