INTELLIGENT ROBOTS

CHAPTER 10: ERROR PROPAGATION AND FEATURE EXTRACTION

Outline

- Error Propagation
- Feature Extraction
- Split and Merge
- Least Square Estimation

Error Propagation: Motivation

- Probabilistic robotics is
  - **Representation**
  - → **Propagation**
  - **Reduction**

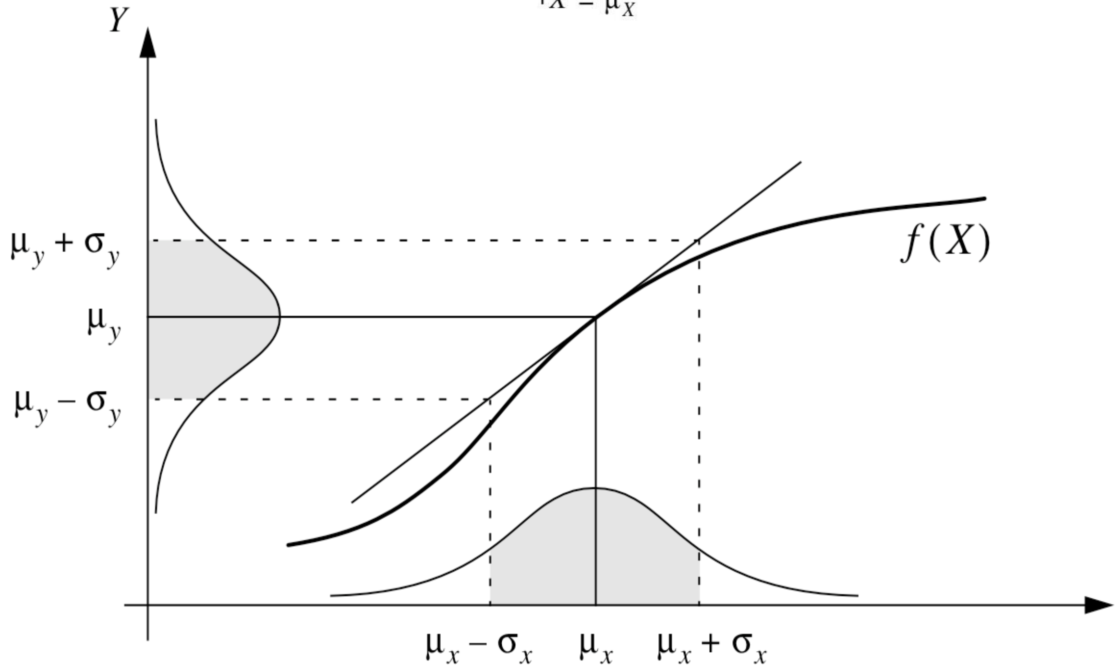
  of uncertainty

- **First-order error propagation** is fundamental for:
  Kalman filter (KF), landmark extraction, KF-based localization and SLAM

First-Order Error Propagation

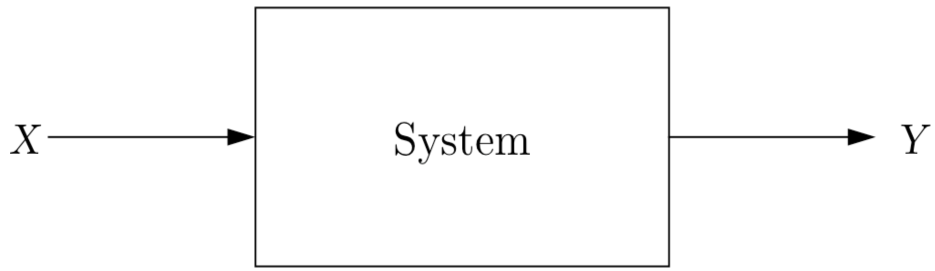
Approximating *f(X)* by a **first-order** Taylor series expansion about the point $X = \mu_X$

$$Y \approx f(\mu_X) + \left.\frac{\partial f}{\partial X}\right|_{X = \mu_X} (X - \mu_X)$$



First-Order Error Propagation

X,Y assumed to be Gaussian $Y = f(X)$



Taylor series expansion

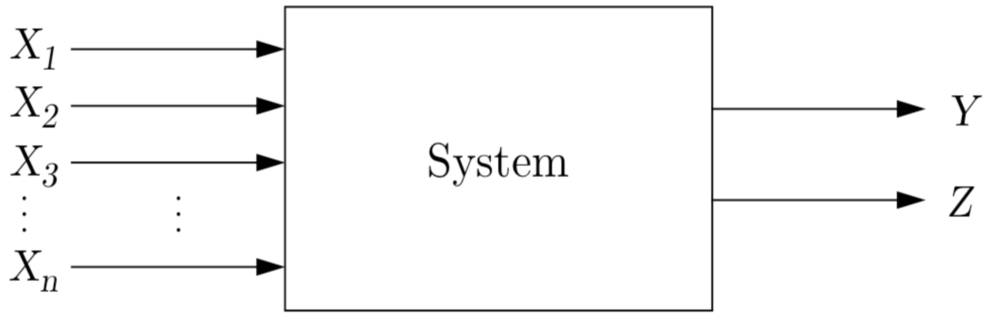
$$Y \approx f(\mu_X) + \left.\frac{\partial f}{\partial X}\right|_{X = \mu_X} (X - \mu_X)$$

Wanted: $\mu_Y \quad \sigma_Y^2$ (Solution on blackboard)

First-Order Error Propagation

$$Y = f(X_1, X_2, \ldots, X_n)$$

$$Z = g(X_1, X_2, \ldots, X_n)$$

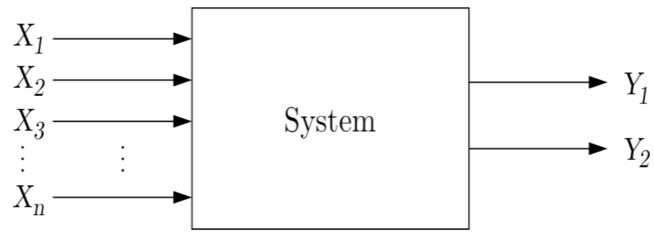


Wanted: σ_{YZ} (Exercise)

First-Order Error Propagation

Putting things together...

$$C_X = \begin{bmatrix} \sigma^2_{X_1} & \sigma_{X_1X_2} & \cdots & \sigma_{X_1X_n} \\ \sigma_{X_2X_1} & \sigma^2_{X_2} & \cdots & \sigma_{X_2X_n} \\ \vdots & \vdots & & \vdots \\ \sigma_{X_nX_1} & \sigma_{X_nX_2} & \cdots & \sigma^2_{X_n} \end{bmatrix}$$



$$C_Y = \begin{bmatrix} \sigma^2_{Y_1} & \sigma_{Y_1Y_2} \\ \sigma_{Y_2Y_1} & \sigma^2_{Y_2} \end{bmatrix}$$

with

$$\sigma^2_Y = \sum_i \left(\frac{\partial f}{\partial X_i}\right)^2 \sigma^2_i + \sum_{i \neq j}\sum \left(\frac{\partial f}{\partial X_i}\right)\left(\frac{\partial f}{\partial X_j}\right)\sigma_{ij}$$

$$\sigma_{YZ} = \sum \left(\frac{\partial f}{\partial X_i}\right)\left(\frac{\partial g}{\partial X_i}\right)\sigma^2_i + \sum_{i \neq j}\sum \left(\frac{\partial f}{\partial X_i}\right)\left(\frac{\partial g}{\partial X_j}\right)\sigma_{ij}$$

→ "Is there a **compact form?...**"

First-Order Error Propagation

- It's a **non-square matrix** $n \times m$ in general
- Suppose you have a vector-valued function $f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix}$
- Let the **gradient operator** be the vector of (first-order) partial derivatives

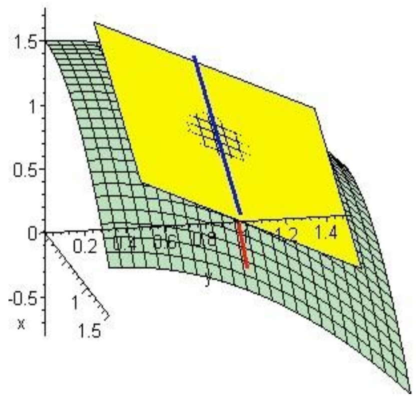
$$\nabla_{\mathbf{x}} = \begin{bmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} & \cdots & \frac{\partial}{\partial x_n} \end{bmatrix}^T$$

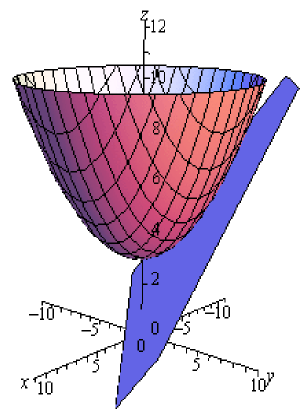
- Then, the **Jacobian matrix** is defined as

$$\mathbf{F}_{\mathbf{x}} = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x_1} & \cdots & \frac{\partial}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_n} \end{bmatrix}$$

Jacob Matrix

- It’s the orientation of the **tangent plane** to the vector- valued function at a given point





- **Generalizes the gradient** of a scalar valued function
- Heavily used for **first-order error propagation…**

First-Order Propagation

...Yes! Given

- Input covariance matrix C_X
- Jacobian matrix F_X

the **Error Propagation Law**

$$\boxed{C_Y = F_X C_X F_X^T}$$

computes the output covariance matrix C_Y

First-Order Propagation

Alternative Derivation:

$$
\begin{aligned}
\mu_x &= E(x) \\
&= E(Au + b) \\
&= AE(u) + b \\
&= A\mu_u + b
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_x &= E((x - E(x))(x - E(x))^T) \\
&= E((Au + b - AE(u) - b)(Au + b - AE(u) - b)^T) \\
&= E((A(u - E(u)))(A(u - E(u)))^T) \\
&= E((A(u - E(u)))((u - E(u))^T A^T)) \\
&= AE((u - E(u))(u - E(u))^T)A^T \\
&= A\Sigma_u A^T
\end{aligned}
$$

Example: Line Extraction

Wanted: Parameter Covariance Matrix

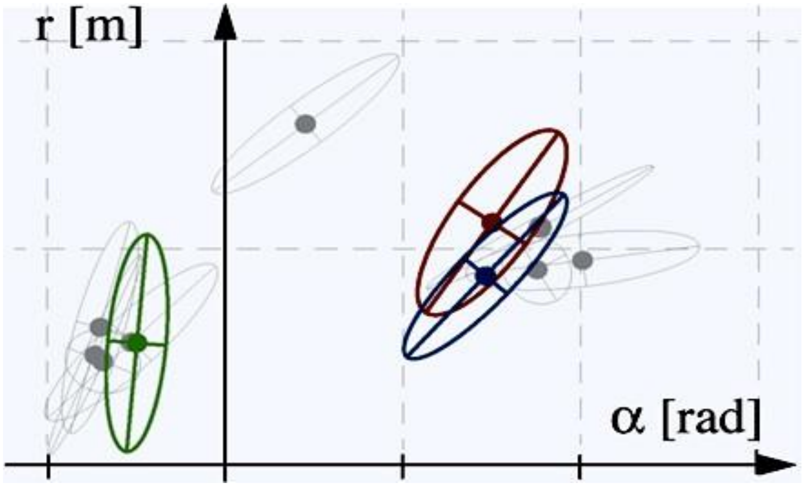
$$C_{AR} = \begin{bmatrix} \sigma_A^2 & \sigma_{AR} \\ \sigma_{AR} & \sigma_R^2 \end{bmatrix}$$

$$C_X = \begin{bmatrix} \sigma_{\rho_1}^2 & 0 & \dots & 0 \\ 0 & \sigma_{\rho_2}^2 & \dots & 0 \\ : & : & & : \\ 0 & 0 & \dots & \sigma_{\rho_n}^2 \end{bmatrix}$$

Simplified sensor model: all independent

$$C_{AR} = F_X C_X F_X^T$$

Result: Gaussians in the parameter space



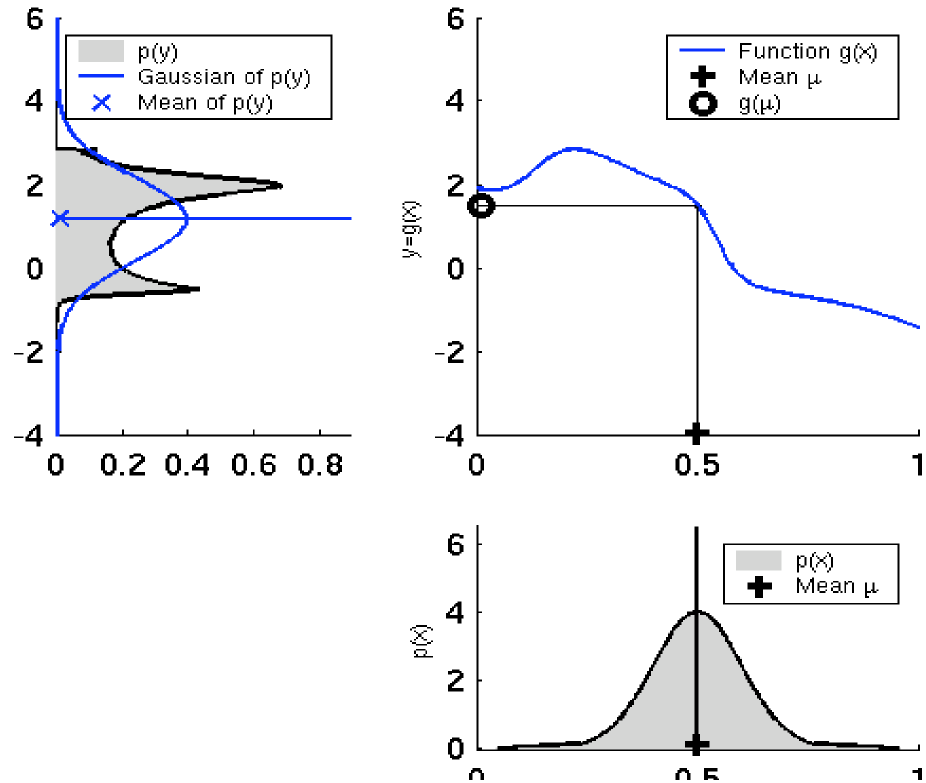
Other Error Propagation Techniques

- **Second-Order Error Propagation**

  Rarely used (complex expressions)

- **Monte-Carlo**

Non-parametric representation of uncertainties



1. Sampling from *p(X)*
2. Propagation of samples
3. Histogramming
4. Normalization

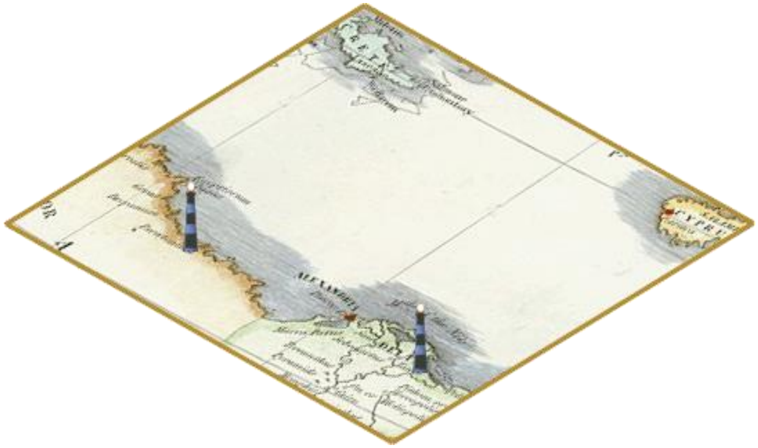
Outline

- Error Propagation
- Feature Extraction
- Split and Merge
- Least Square Estimation

Feature Extraction: Motivation

Landmarks for:

- Localization
- SLAM
- Scene analysis



Examples:

- **Lines, corners, clusters:** good for indoor
- **Circles, rocks, plants:** good for outdoor

Features: Properties

A feature/landmark is a **physical object** which is

- **static**
- **perceptible**
- (at least locally) **unique**

Abstraction from the raw data...

- **type** (range, image, vibration, etc.)
- **amount** (sparse or dense)
- **origin** (different sensors, map)

\+ Compact, efficient, accurate, scales well, semantics

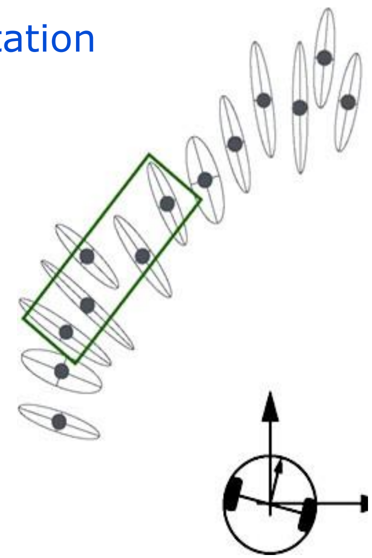
– Not general

Feature Extraction

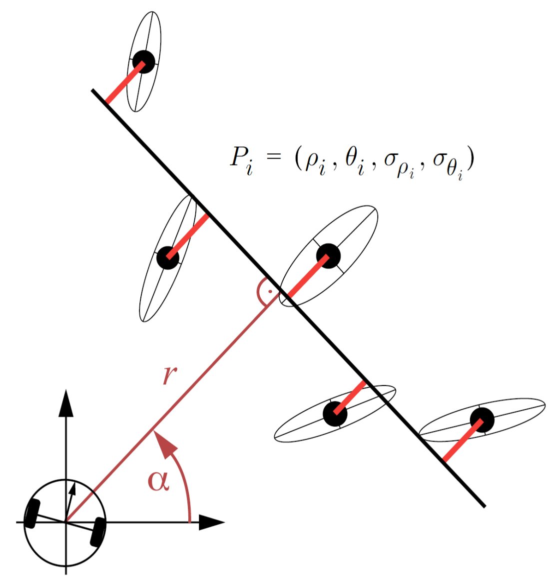
Can be subdivided into two subproblems:

- **Segmentation:** *Which* points contribute?
- **Fitting:** *How* do the points contribute?

Segmentation

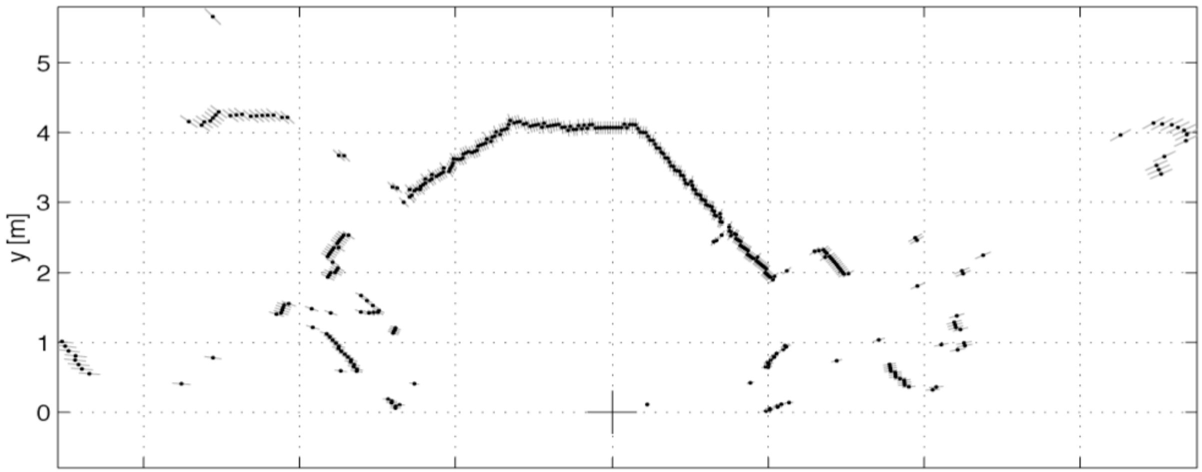


Fitting

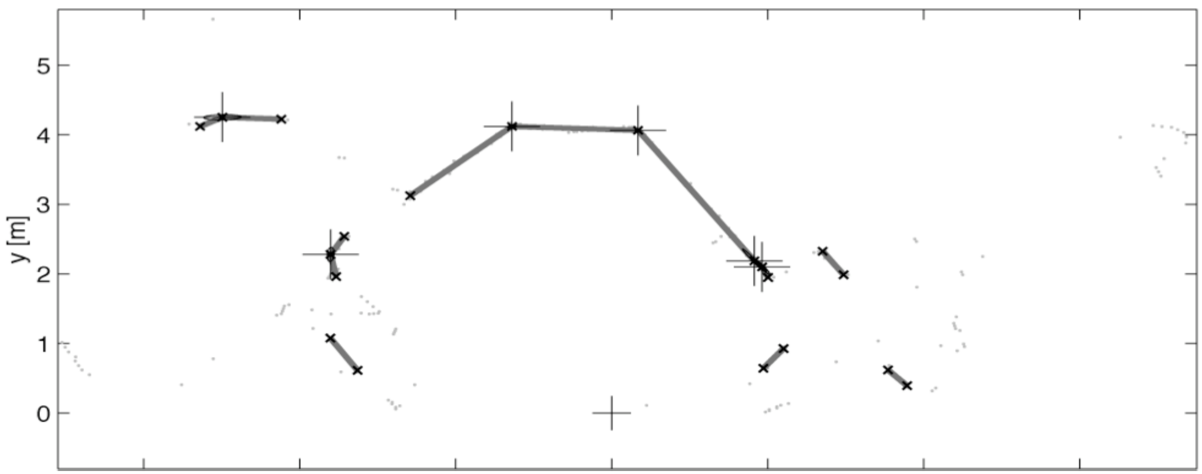


Feature Extraction

Raw
range data



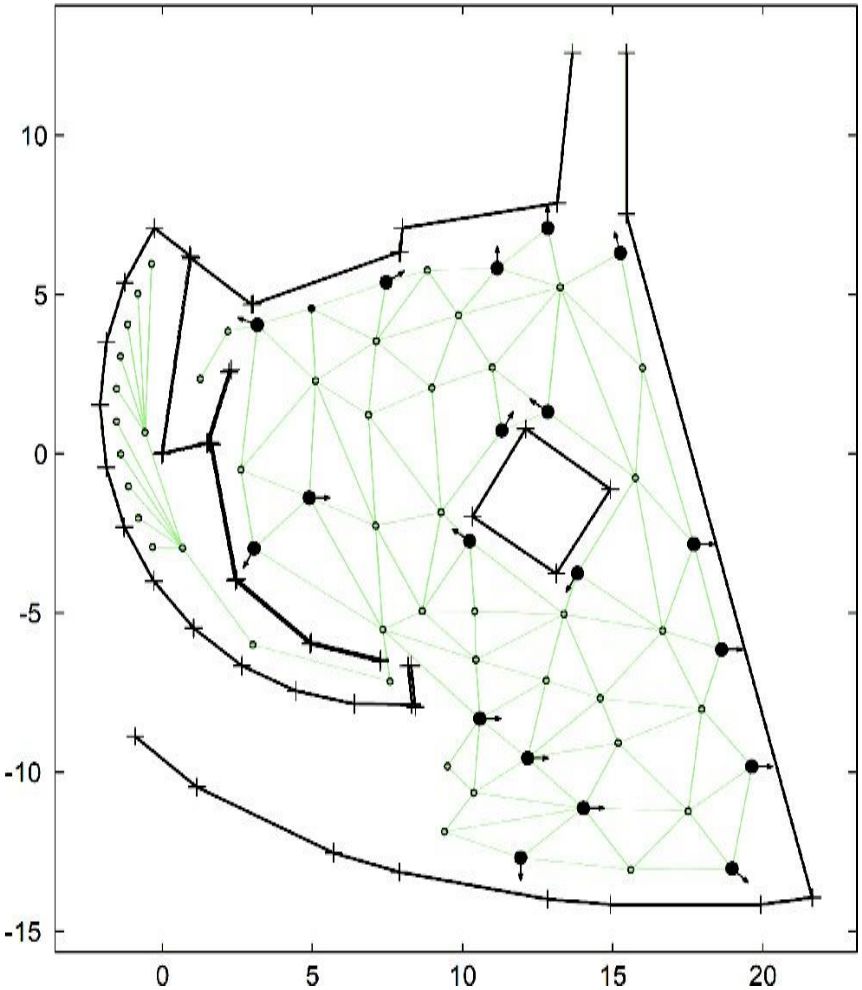
Line
segments



Example: Global Map with Lines

Expo.02 map

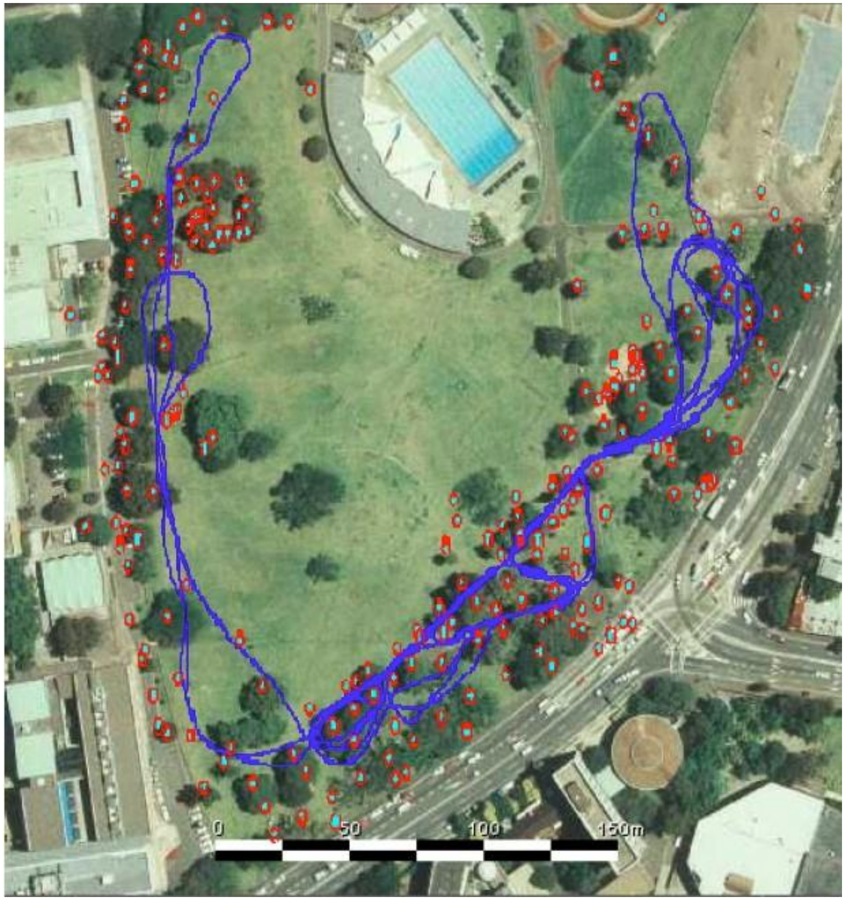
- 315 m²
- 44 Segments
- 8 kbytes
- 26 bytes / m²
- Localization accuracy ~1cm



Example: Global Feature with Circles

Victoria Park, Sydney

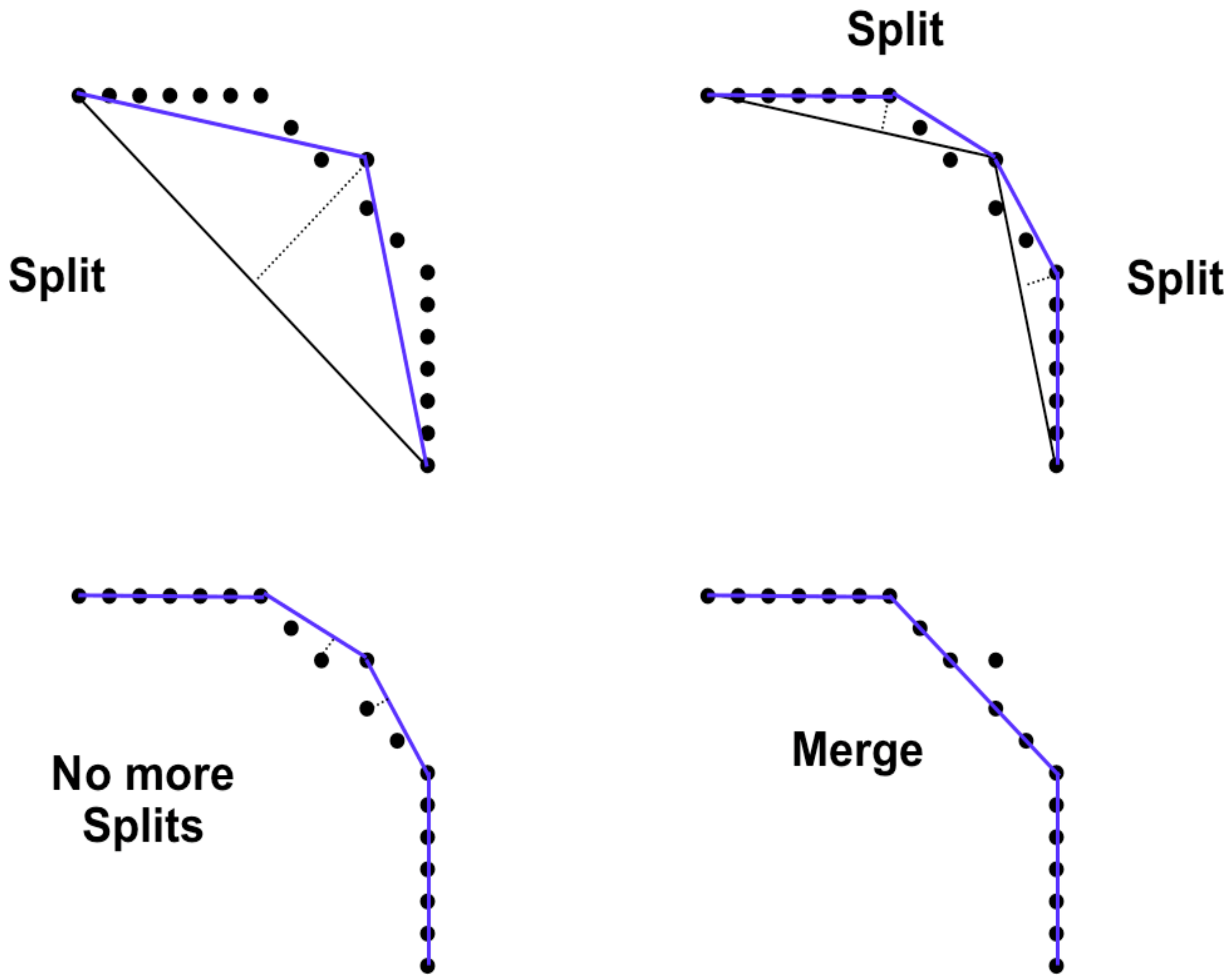
- Trees



Outline

- Error Propagation
- Feature Extraction
- Split and Merge
- Least Square Estimation

Split and Merge



Split and Merge: Algorithm

Split

- Obtain the line passing by the two extreme points
- Find the most distant point to the line
- If distance > threshold, split and repeat with the left and right point sets

Merge

- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance <= threshold, merge both segments

Split and Merge: Improvements

- Residual analysis before split

$$\sum_{i=P_S}^{P_E} d_i^2 > \sum_{i=P_S}^{P_B} d_i^2 + \sum_{i=P_B}^{P_E} d_i^2$$

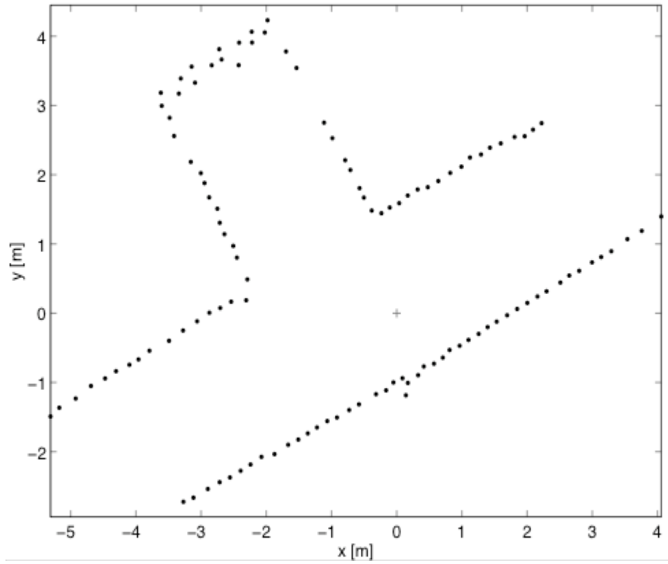
P_S, P_E, P_B
start-, end-, break-point

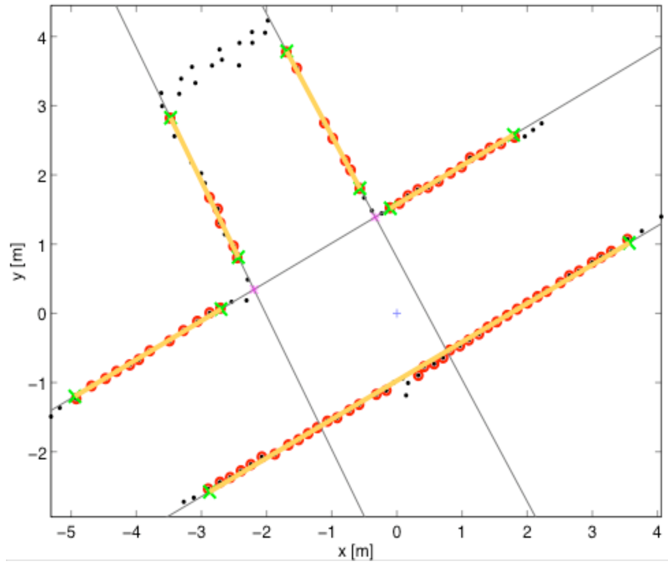
Split only if the break point provides a "better interpretation" in terms of the error sum

[Castellanos 1998]

Split and Merge: Improvements

- Merge **non-consecutive** segments as a post-processing step

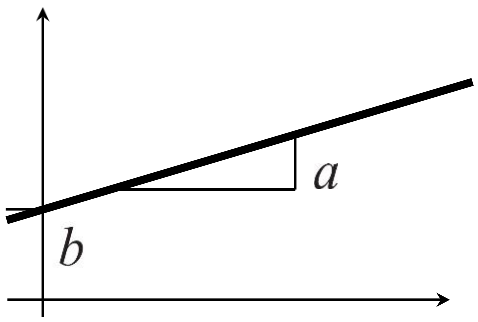




Line Representation

Choice of the line representation matters!

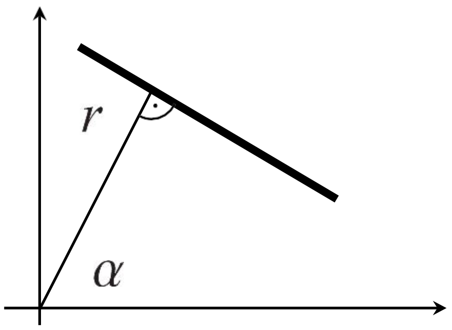
Intercept-Slope



$$y = ax + b$$

$$C = \begin{bmatrix} \sigma_a^2 & \sigma_{ab} \\ \sigma_{ba} & \sigma_b^2 \end{bmatrix}$$

Hessian model



$$x \cos\alpha + y \sin\alpha - r = 0$$

$$C = \begin{bmatrix} \sigma_\alpha^2 & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_r^2 \end{bmatrix}$$

Each model has advantages and drawbacks

Fit Expressions

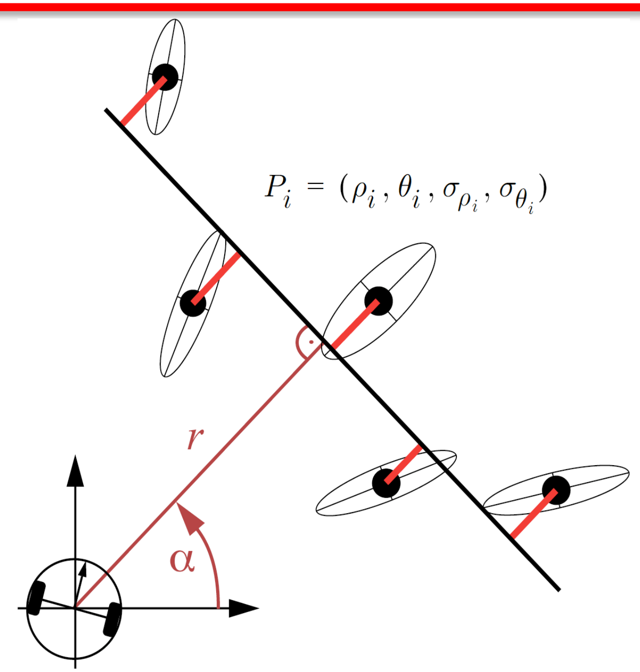
Given:

A set of *n* points in polar coordinates

Wanted:

Line parameters

α, r



$$\tan 2\alpha = \frac{\frac{2}{\sum w_i}\sum_i \sum_{<j} w_i w_j \rho_i \rho_j \sin(\theta_i + \theta_j) + \frac{1}{\sum w_i}\sum (w_i - \sum w_j) w_i \rho_i^2 \sin 2\theta_i}{\frac{2}{\sum w_i}\sum_i \sum_{<j} w_i w_j \rho_i \rho_j \cos(\theta_i + \theta_j) + \frac{1}{\sum w_i}\sum (w_i - \sum w_j) w_i \rho_i^2 \cos 2\theta_i}$$

$$r = \frac{\sum w_i \rho_i \cos(\theta_i - \alpha)}{\sum w_i}$$

[Arras 1997]

Outline

- Error Propagation
- Feature Extraction
- Split and Merge
- Least Square Estimation

LSQ Estimation

Regression, Least Squares-Fitting

$$\epsilon_i = x_i \cos\alpha + y_i \sin\alpha - r$$

$$S = \sum_{i=1}^{n} \epsilon_i^{\,2}$$

Solve the non-linear equation system

$$\frac{\partial S}{\partial \alpha} = 0 \qquad \frac{\partial S}{\partial r} = 0$$

Solution (for points in Cartesian coordinates):

$\rightarrow$ Solution on blackboard

Circle Extraction

Can be formulated as a **linear** regression problem

Given n points $\mathcal{P} = \{P_i\}_{i=1}^{n}$ with $P_i = (x_i \;\; y_i)^T$

Circle equation: $(x_i - x_c)^2 + (y_i - y_c)^2 = r_c^2$

Develop circle equation

$$x_i^2 - 2x_i x_c + x_c^2 + y_i^2 - 2y_i y_c + y_c^2 = r_c^2$$

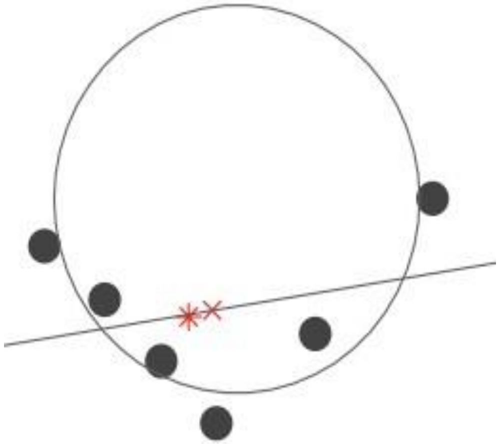
$$(-2x_i \quad -2y_i \;\; 1)\begin{pmatrix} x_c \\ y_c \\ x_c^2 + y_c^2 - r_c^2 \end{pmatrix} = (-x_i^2 - y_i^2)$$

Circle Extraction

Leads to **overdetermined** equation system $A \cdot x = b$

$$A = \begin{pmatrix} -2x_1 & -2y_1 & 1 \\ -2x_2 & -2y_2 & 1 \\ \vdots & \vdots & \vdots \\ -2x_n & -2y_n & 1 \end{pmatrix} \quad b = \begin{pmatrix} -x_1^2 - y_1^2 \\ -x_2^2 - y_2^2 \\ \vdots \\ -x_n^2 - y_n^2 \end{pmatrix}$$

with vector of unknowns



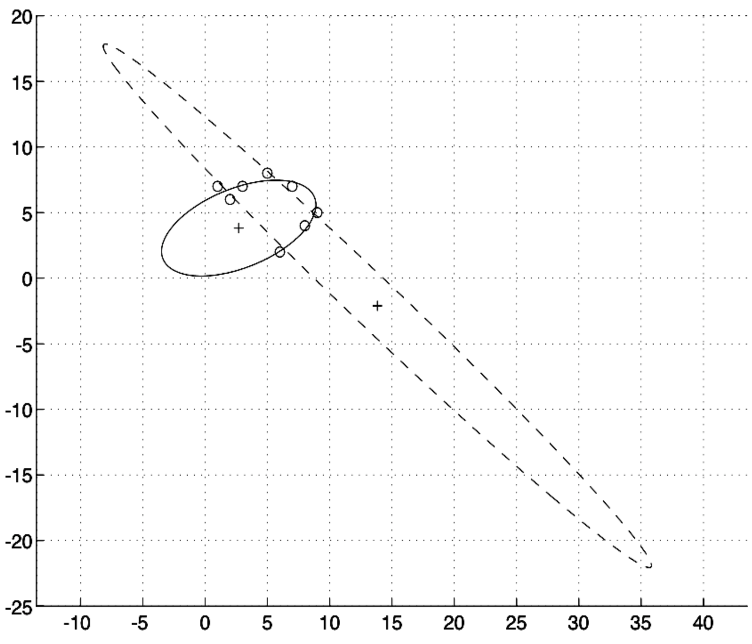
Solution via **Pseudo-Inverse**

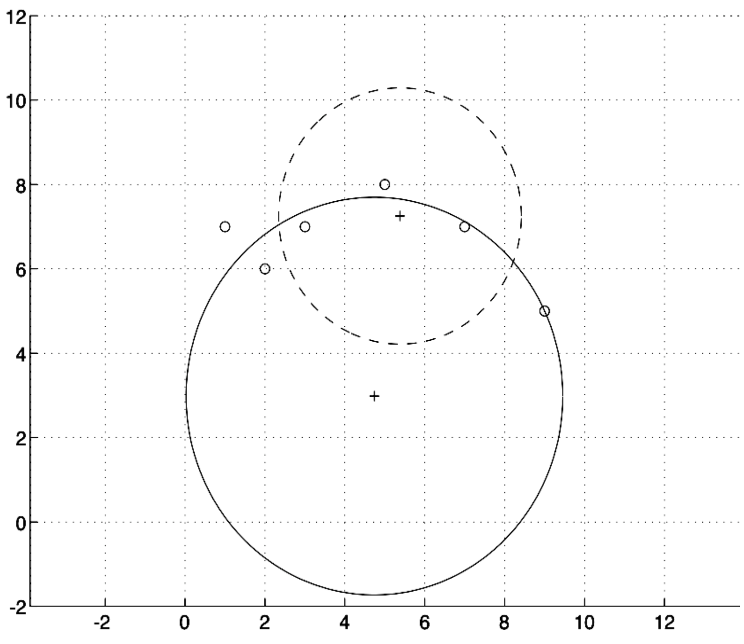
$$x = (A^T A)^{-1} A^T \cdot b$$

(assuming that *A* has full rank)

Fitting Curves to Points

Attention: Always know the errors that you minimize!





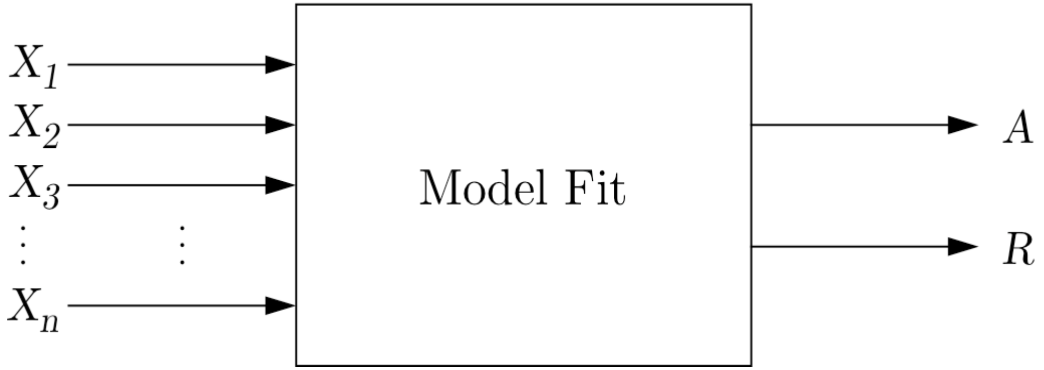
Algebraic versus **geometric** fit solutions [Gander 1994]

LSQ Estimation: Uncertainties

How does the **input uncertainty** propagate over the fit expressions to the **output**?

$X_1, \ldots, X_n$: Gaussian input random variables

A, R : Gaussian output random variables



Example: Line Extraction

Wanted: Parameter Covariance Matrix

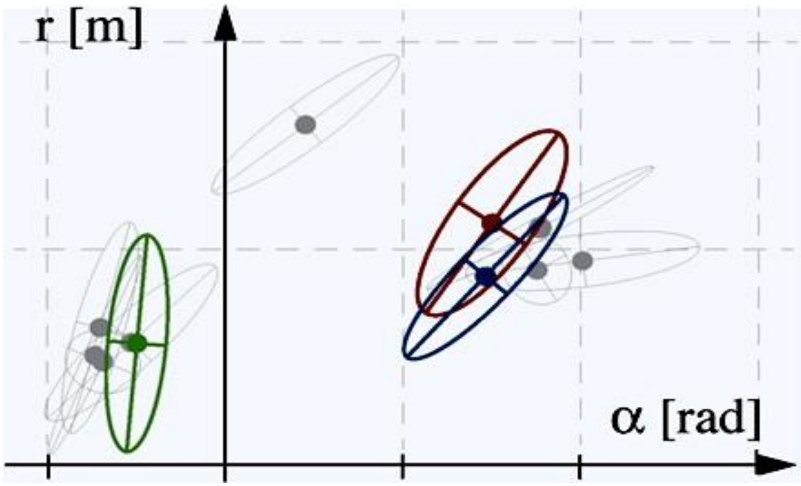
$$C_{AR} = \begin{bmatrix} \sigma_A^2 & \sigma_{AR} \\ \sigma_{AR} & \sigma_R^2 \end{bmatrix}$$

$$C_X = \begin{bmatrix} \sigma_{\rho_1}^2 & 0 & \dots & 0 \\ 0 & \sigma_{\rho_2}^2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \sigma_{\rho_n}^2 \end{bmatrix}$$

Simplified sensor model: all independent

$$C_{AR} = F_X C_X F_X^T$$

Result: Gaussians in the parameter space



Line Extraction in Real Time

- Robot *Pygmalion* EPFL, Lausanne
- CPU: PowerPC 604e at 300 MHz Sensor: 2 SICK LMS
- Line Extraction Times: ***~ 25 ms***