Spatial Awareness

Uncertainty and State Estimation

Teresa Vidal-Calleja

LOCALISATION

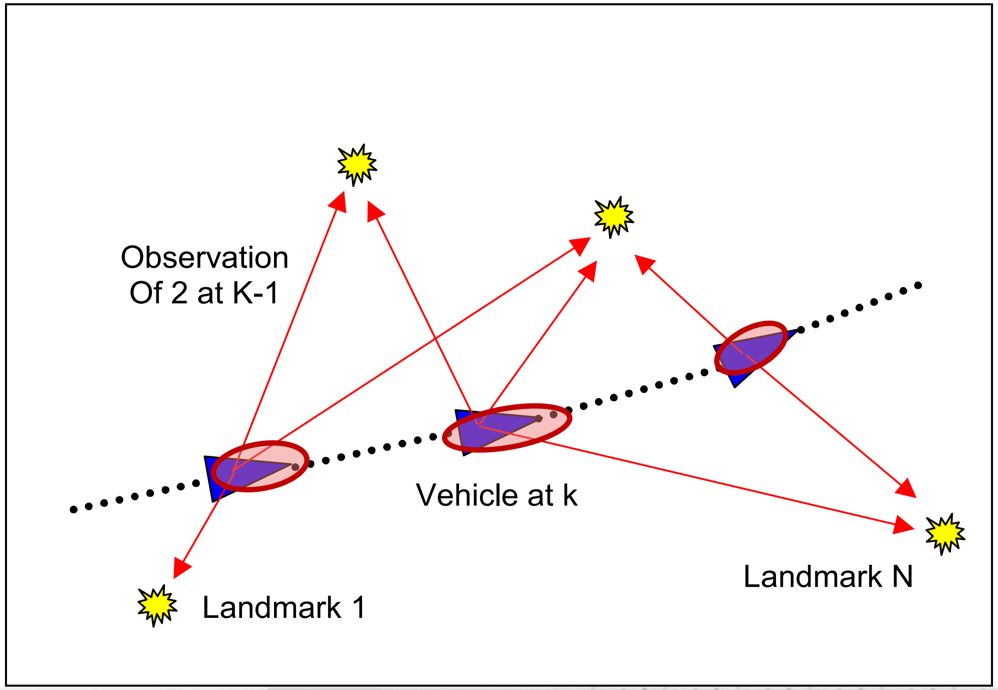
Localisation Problem: How to estimate the robot pose from noisy sensor information?

The standard method is based on probability theory to combine (FUSE) information from different noisy sensors

Why probability?

- true location is unknow
- many possible locations
- which one is the most likely one?

Map is known!



MAPPING

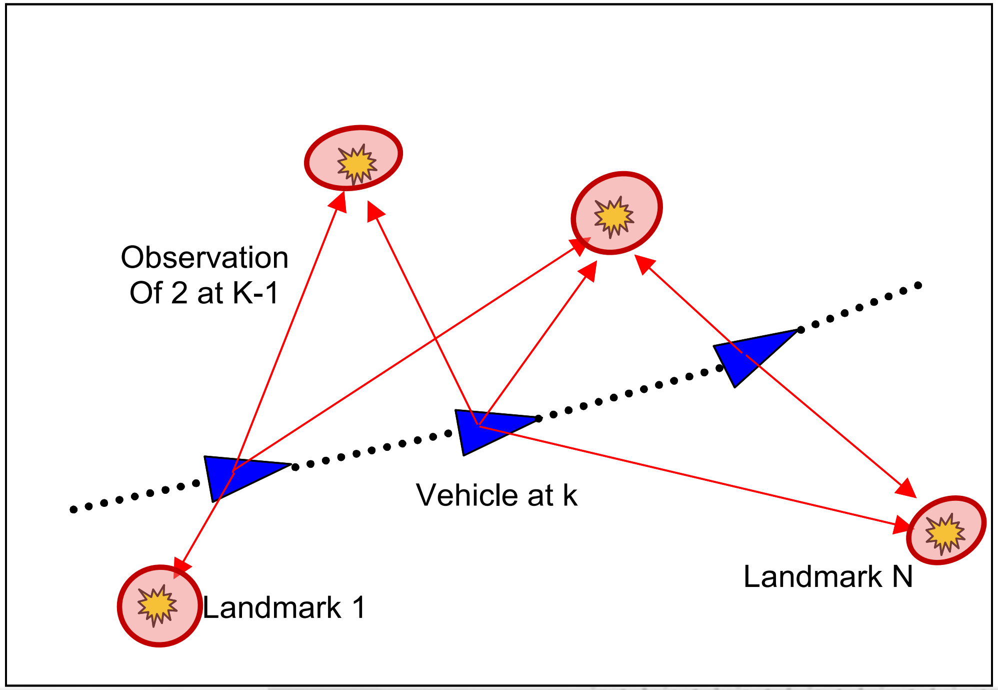
Mapping Problem: How to estimate the spatial model of robot's environment from noisy sensors?

How do robots understand the world?

Data must be fused over time

Localisation is known!

The spatial representation is static

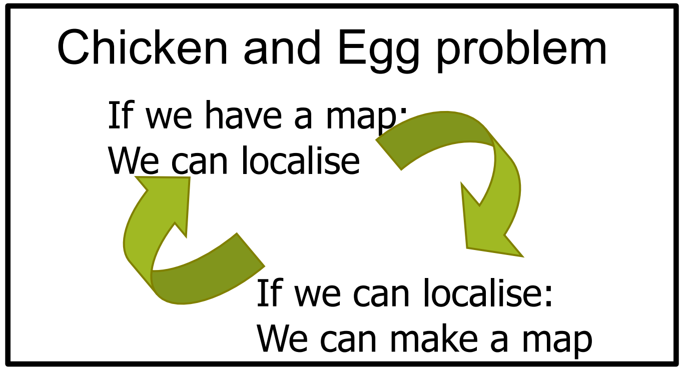


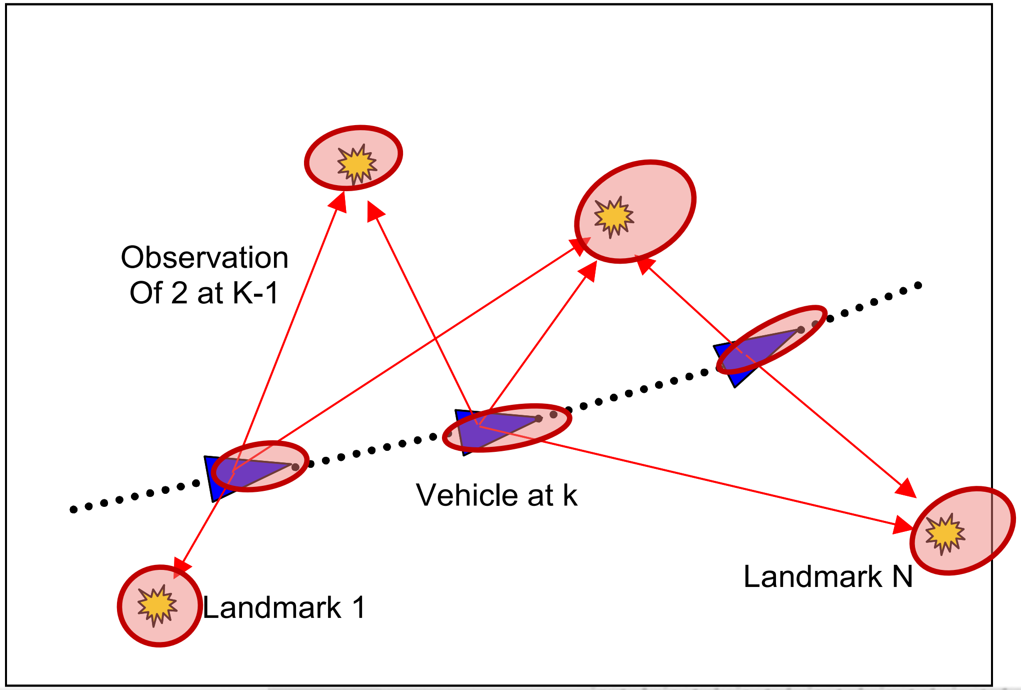
SLAM

SLAM Problem: How to estimate the robot pose and at the same time the map of the environment from noisy sensor information?

The standard method is based on probability theory to combine (FUSE) information from different noisy sensors

- True location unknown
- Map is also unknown





DISCRETE RANDOM VARIABLES

- X denotes a random variable
- X can take on a countable number of values in $\{x_1, x_2, \ldots, x_n\}$
- $P(X=x_i)$ or $P(x_i)$ is the probability that the random variable X takes on value x_i

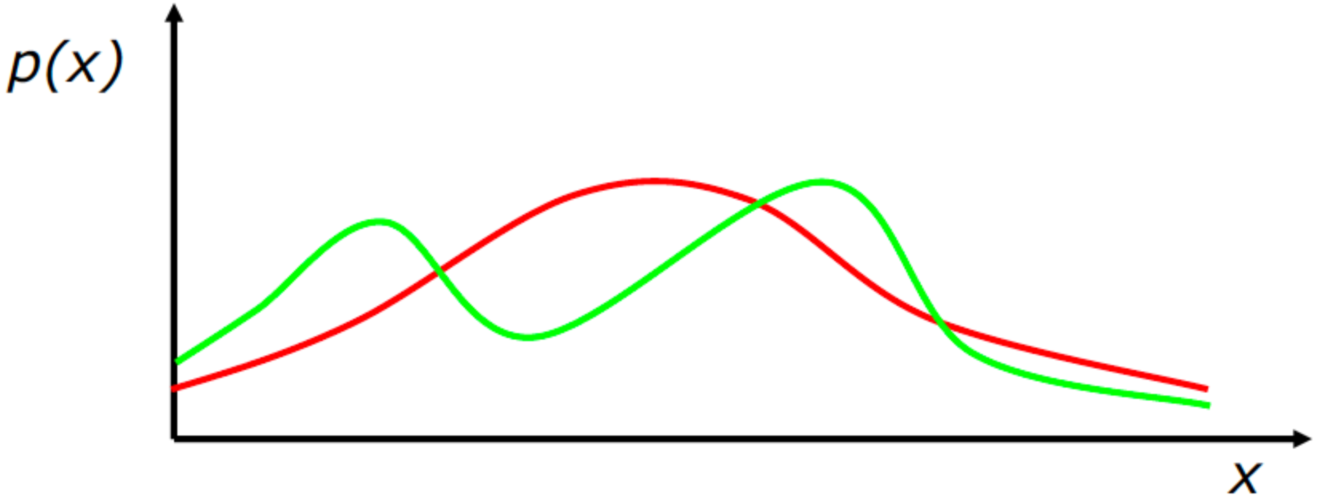
- $P(.)$ is called probability mass function

$P(room)$ = $(0.7, 0.2, 0.08, 0.02)$

CONTINUOUS RANDOM VARIABLES

- X takes on values in the continuum
- $p(X=x)$ or $p(x)$ is a probability density function

$$\Pr(x \in (a,b)) = \int_a^b p(x)\,dx$$



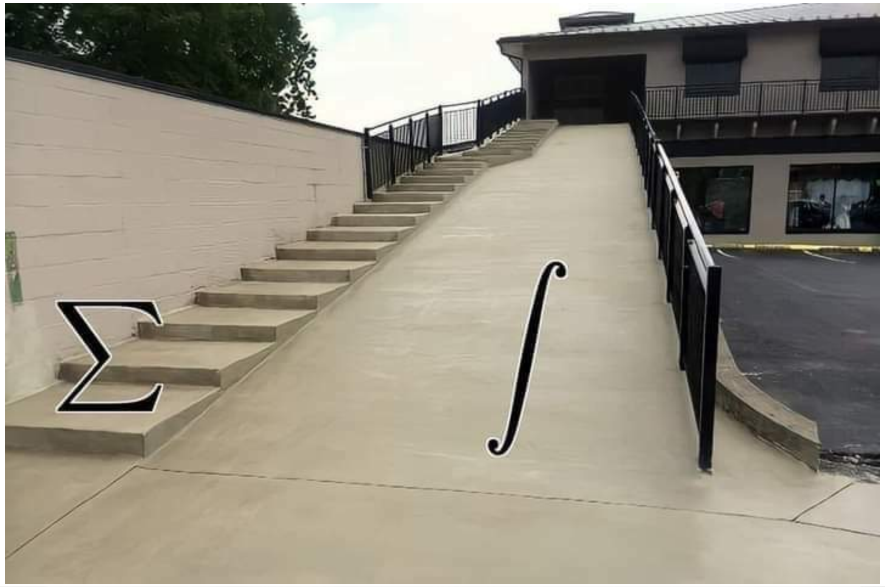
PROBABILITY SUMS UP TO 1

Discrete case

$$\sum_x P(x) = 1$$

Continuous case

$$\int p(x)\, dx = 1$$



LAW OF TOTAL PROBABILITY

Discrete case

$$P(x) = \sum_y P(x \mid y) P(y)$$

Continuous case

$$p(x) = \int p(x \mid y)\, p(y)\, dy$$

JOINT AND CONDITIONAL PROBABILITIES

- $P(X{=}x \text{ and } Y{=}y) = P(x,y)$
- If X and Y are independent then

  $$P(x,y) = P(x)\, P(y)$$

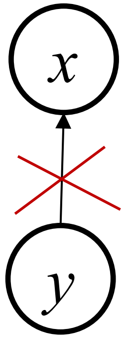
- $P(x \mid y)$ is the probability of x given y

  $$P(x \mid y) = P(x,y) \,/\, P(y)$$

  $$P(x,y) = P(x \mid y)\, P(y)$$

- If X and Y are independent then

  $$P(x \mid y) = P(x)$$



MARGINALISATION

Discrete case

$$P(x)=\sum_y P(x,y)$$

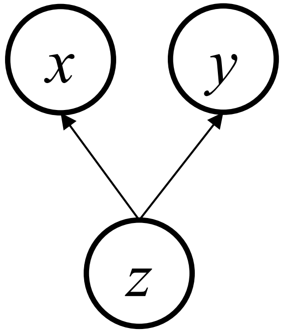
Continuous case

$$p(x)=\int p(x,y)\,dy$$

CONDITIONAL INDEPENDENCE

$$P(x, y, z) = P(x \mid z) P(y \mid z)$$

- Equivalent to $P(y, z) = P(y \mid z, x)$
  and $P(x, z) = P(x \mid z, y)$



- But this does not necessarily mean

  $$P(x, y) = P(x) P(y)$$

  (real independence)

BAYES RULE

$$P(x,y) = P(x \mid y)P(y) = P(y \mid x)P(x)$$

$$\Rightarrow$$

posterior

$$\boxed{P(x \mid y) = \frac{P(y \mid x)\ P(x)}{P(y)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}}$$

GAUSSIAN – MARGINALIZATION AND CONDITIONING

- Given

$$x = \begin{pmatrix} x_a \\ x_b \end{pmatrix} \qquad p(x) = \mathcal{N}$$

- The marginals are Gaussians

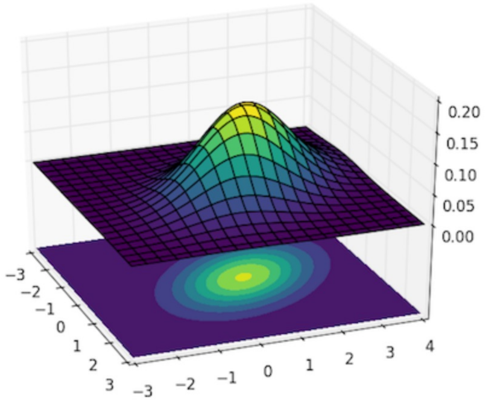
$$p(x_a) = \mathcal{N} \qquad p(x_b) = \mathcal{N}$$

- and conditionals are Gaussians

$$p(x_a \mid x_b) = \mathcal{N} \qquad p(x_b \mid x_a) = \mathcal{N}$$

$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}):$$

$$\mu = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} \quad \Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$$



MARGINALISATION

- Given

$$p(x) = p(x_a, x_b) = \mathcal{N}(\mu, \Sigma)$$

with

$$\mu = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} \quad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

- The marginal distribution is

$$p(x_a) = \int p(x_a, x_b) \, dx_b = \mathcal{N}(\mu, \Sigma)$$

with

$$\mu = \mu_a \qquad \Sigma = \Sigma_{aa}$$

CONDITIONING

- Given $$p(x) = p(x_a, x_b) = \mathcal{N}(\mu, \Sigma)$$

  with $$\mu = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} \quad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

- The conditional distribution is

$$p(x_a \mid x_b) = \frac{p(x_a, x_b)}{p(x_b)} = \mathcal{N}(\mu, \Sigma)$$

  with

$$\mu = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(b - \mu_b)$$

$$\Sigma = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$$

GAUSSIAN – MARGINALIZATION AND CONDITIONING

$$p\left(\left(\begin{array}{c} x_a \\ x_b \end{array}\right)\right) = \mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\left(\begin{array}{c} \mu_a \\ \mu_b \end{array}\right), \left(\begin{array}{cc} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{array}\right)\right)$$

marginalization

$$p(x_a) = \mathcal{N}(\mu, \Sigma)$$

$$\mu = \mu_a$$

$$\Sigma = \Sigma_{aa}$$

conditioning

$$p(x_a \mid x_b) = \mathcal{N}(\mu, \Sigma)$$

$$\mu = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(b - \mu_b)$$

$$\Sigma = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$$

INFORMATION FORM

$$p(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_\alpha \\ \boldsymbol{\mu}_\beta \end{bmatrix}, \begin{bmatrix} \Sigma_{\alpha\alpha} & \Sigma_{\alpha\beta} \\ \Sigma_{\beta\alpha} & \Sigma_{\beta\beta} \end{bmatrix}\right) = \mathcal{N}^{-1}\left(\begin{bmatrix} \boldsymbol{\eta}_\alpha \\ \boldsymbol{\eta}_\beta \end{bmatrix}, \begin{bmatrix} \Lambda_{\alpha\alpha} & \Lambda_{\alpha\beta} \\ \Lambda_{\beta\alpha} & \Lambda_{\beta\beta} \end{bmatrix}\right)$$

	MARGINALIZATION $p(\boldsymbol{\alpha}) = \int p(\boldsymbol{\alpha}, \boldsymbol{\beta})\, d\boldsymbol{\beta}$	CONDITIONING $p(\boldsymbol{\alpha} \mid \boldsymbol{\beta}) = p(\boldsymbol{\alpha}, \boldsymbol{\beta}) / p(\boldsymbol{\beta})$
COVARIANCE FORM	$\boldsymbol{\mu} = \boldsymbol{\mu}_\alpha$ $\Sigma = \Sigma_{\alpha\alpha}$	$\boldsymbol{\mu}' = \boldsymbol{\mu}_\alpha + \Sigma_{\alpha\beta}\Sigma_{\beta\beta}^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}_\beta)$ $\Sigma' = \Sigma_{\alpha\alpha} - \Sigma_{\alpha\beta}\Sigma_{\beta\beta}^{-1}\Sigma_{\beta\alpha}$
INFORMATION FORM	$\boldsymbol{\eta} = \boldsymbol{\eta}_\alpha - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\boldsymbol{\eta}_\beta$ $\Lambda = \Lambda_{\alpha\alpha} - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\Lambda_{\beta\alpha}$	$\boldsymbol{\eta}' = \boldsymbol{\eta}_\alpha - \Lambda_{\alpha\beta}\boldsymbol{\beta}$ $\Lambda' = \Lambda_{\alpha\alpha}$

Walter et al, Exactly Sparse Extended Information Filters for Feature-Based SLAM, IJRR 2007

LOCALISATION

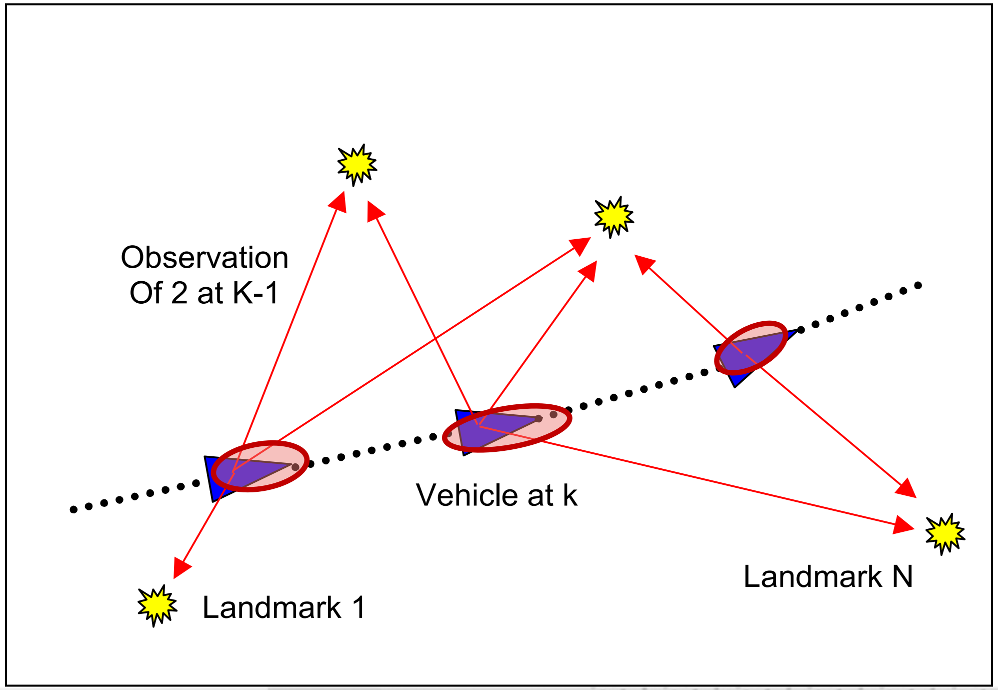
Localisation Problem: How to estimate the robot pose from noisy sensor information?

The standard method is based on probability theory to combine (FUSE) information from different noisy sensors

Why probability?

- true location is unknow
- many possible locations
- which one is the most likely one?

Map is known!



KALMAN FILTER

(Rudolf E. Kálmán, 1930-2016
Image from Wikipedia)

- Provides a linear-least squares estimator in a recursive format

- Applied in a huge range of applications: Signal processing, control, navigation and tracking etc

- It is a Bayes filter
- Optimal solution for linear models and Gaussian distributions

LINEAR MOTION AND SENSOR MODELS

Linear discrete time dynamic system (motion model)

State

Control input

Process noise with covariance Q

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

State transition function

Control input function

Linear measurement equation (sensor model)

Sensor reading

State

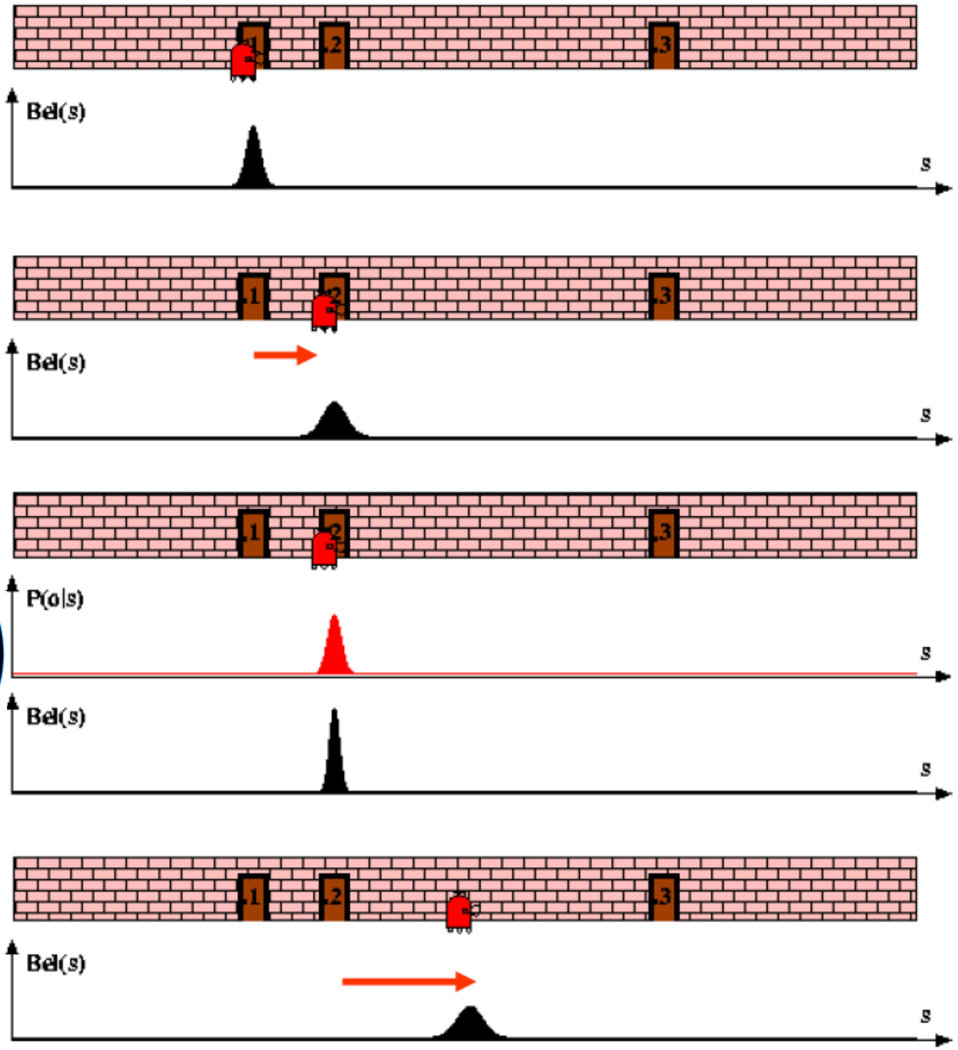
Sensor noise with covariance R

$$z_t = C_t x_t + \delta_t$$

Sensor function

RECURSIVE BAYESIAN ESTIMATION

Kalman Filter is a Bayesian filter -> with Gaussian pdf



Prior $p(\boldsymbol{x})$

Prediction

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = \frac{1}{\sqrt{|2\pi \boldsymbol{Q}|}} \exp\left(-\frac{1}{2} \| x_t - A_t x_{t-1} + B_t u_t \|^2_{\boldsymbol{Q}}\right)$$

Observation -> Update

$$p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\ell}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\ell}), \boldsymbol{R}) = \frac{1}{\sqrt{|2\pi \boldsymbol{R}|}} \exp\left(-\frac{1}{2} \| z_t - C_t x_t \|^2_{\boldsymbol{R}}\right)$$

Posterior -> Prior

$p(\boldsymbol{x}|\boldsymbol{z})$

Prediction

Thrun et al, Probabilistic Robotics 2005, MIT press

KALMAN FILTER ALGORITHM

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

2. Prediction:
3. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t$

5. Correction:
6. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

9. Return μ_t, Σ_t

KF Properties

- Product of two Gaussians is a Gaussian
- Gaussians stays Gaussians under linear transformations
- Marginal and conditional distribution of a Gaussian stays a Gaussian
- The key is computing mean and covariance of the marginal and conditional of a Gaussian

NON-LINEAR DYNAMIC AND MEASUREMENT MODELS

In Robotics our models are mainly NON-LINEAR!

$$\cancel{x_t = A_t x_{t-1} + B_t u_t + \epsilon_t} \qquad \cancel{z_t = C_t x_t + \delta_t}$$

$$x_t = g(u_t, x_{t-1}) + \epsilon_t \qquad z_t = h(x_t) + \delta_t$$

NONLINEAR DYNAMIC AND MEASUREMENT MODELS

Nonlinear discrete time dynamic system (motion model)

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

- g – nonlinear function for motion model
- x – state (vector)
- u – control input (vector)
- w – process noise (vector) with covariance Q

Nonlinear measurement equation (sensor model)

Sensor reading State Sensor noise with covariance R

$$z_t = h(x_t) + \delta_t$$

Nonlinear function for sensor model

RECURSIVE BAYESIAN ESTIMATION

Kalman Filter is a Bayesian filter -> with Gaussian pdf

Prior $p(\boldsymbol{x})$

Prediction

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = \frac{1}{\sqrt{|2\pi \boldsymbol{Q}|}} \exp\left(-\frac{1}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{u}_t)\|^2_{\boldsymbol{Q}}\right)$$

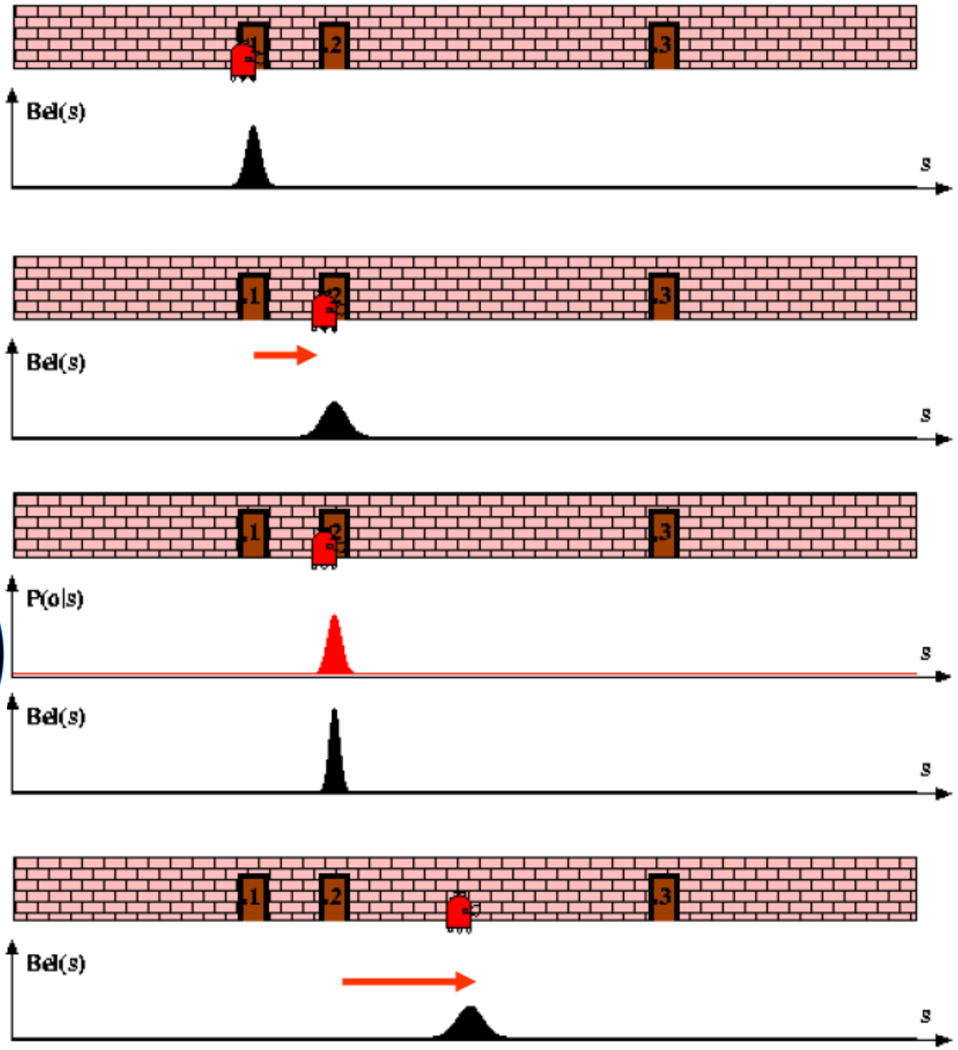
Observation -> Update

$$p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\ell}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\ell}), \boldsymbol{R}) = \frac{1}{\sqrt{|2\pi \boldsymbol{R}|}} \exp\left(-\frac{1}{2}\|\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\ell})\|^2_{\boldsymbol{R}}\right)$$

Posterior -> Prior

$p(\boldsymbol{x}|\boldsymbol{z})$

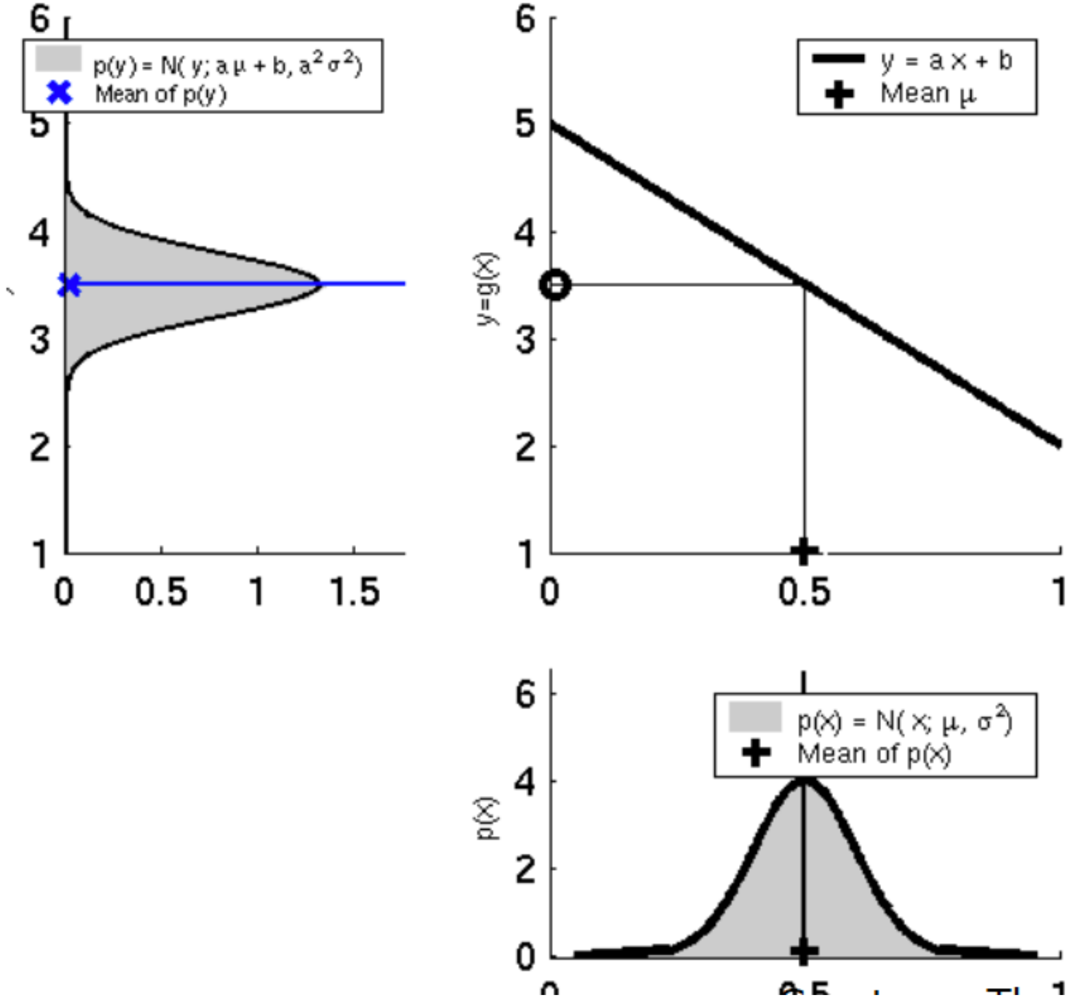
Prediction



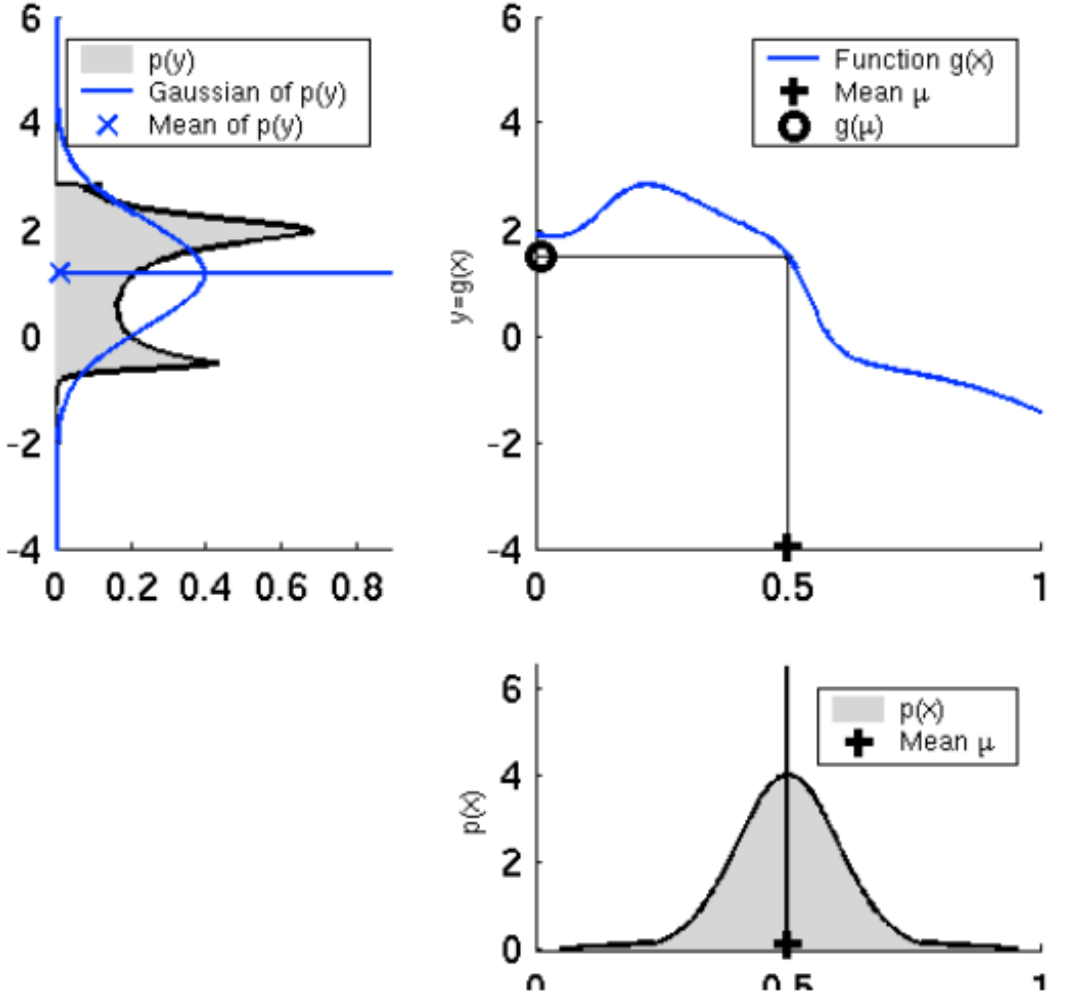
Robotics Institute

Thrun et al, Probabilistic Robotics 2005, MIT press

LINEARITY ASSUMPTION

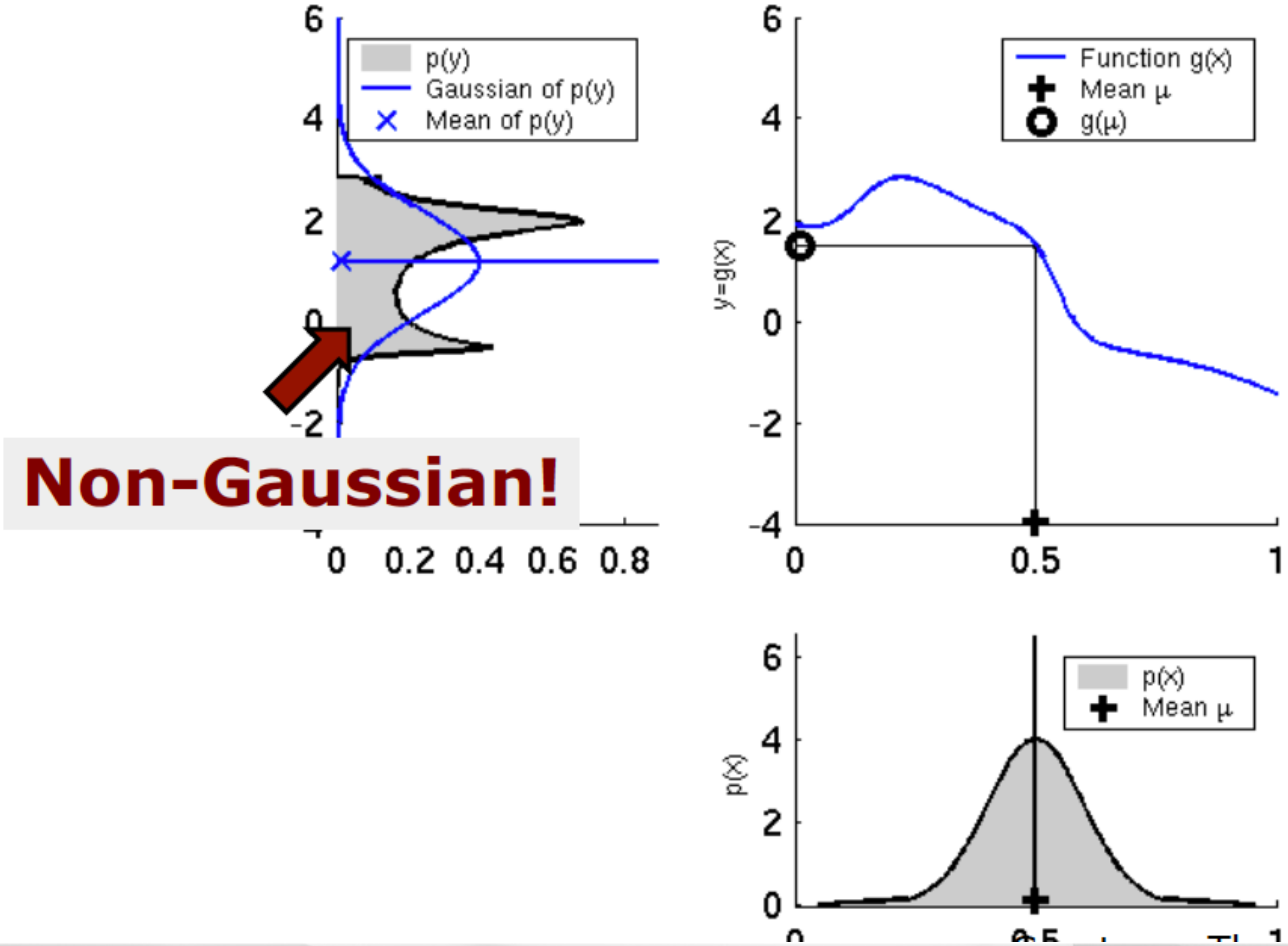


LINEARITY ASSUMPTION



Credit C. Stachniss

LINEARITY ASSUMPTION



NON-GAUSSIAN DISTRIBUTION

- The non-linear functions lead to non-Gaussian distributions
- Kalman filter is not applicable anymore!
- What can be done to resolve this?

  Local linearization!
- **EXTENDED KALMAN FILTER (EKF)**

EKF LINEARISATION: FIRST ORDER TAYLOR EXPANSION

Prediction:

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

Update:

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

Jacobian matrices

JACOBIAN MATRIX

- It is a non-square matrix $m\ x\ n$ 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 by:

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

EKF LINEARISATION: FIRST ORDER TAYLOR EXPANSION

Prediction:

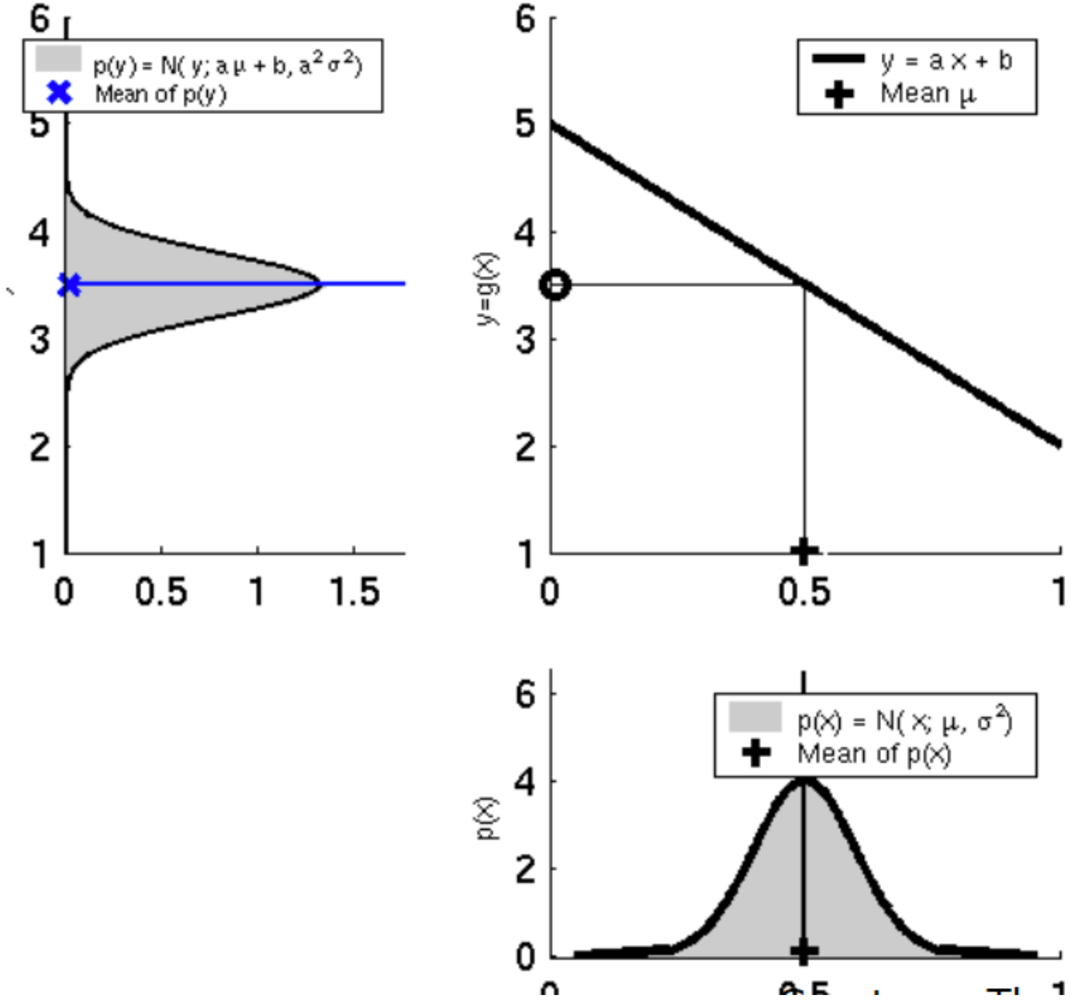
$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{\frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}}_{=:\, G_t} (x_{t-1} - \mu_{t-1})$$

Update:

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{\frac{\partial h(\bar{\mu}_t)}{\partial x_t}}_{=:\, H_t} (x_t - \bar{\mu}_t)$$

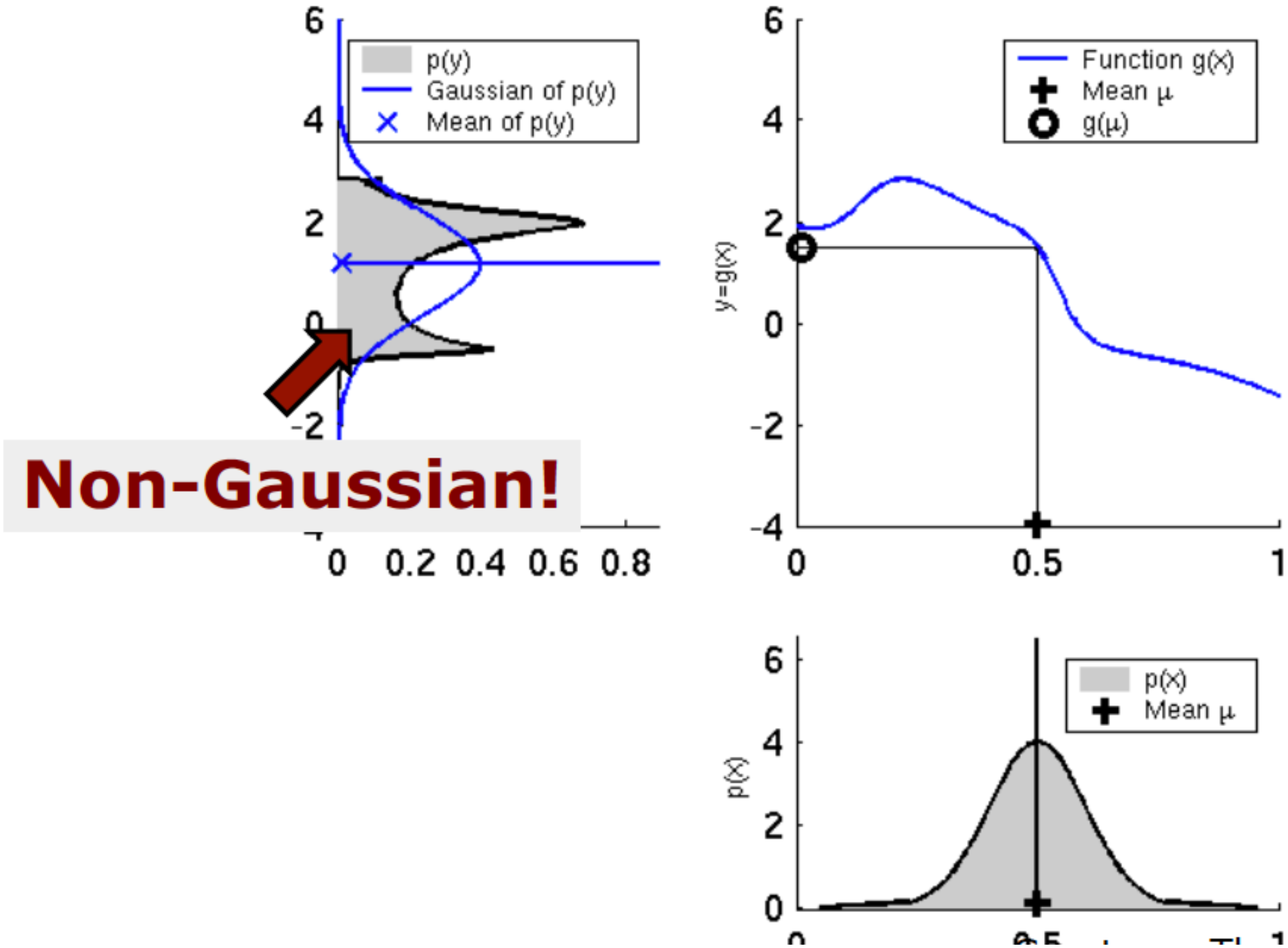
Linear functions!

LINEARITY ASSUMPTION

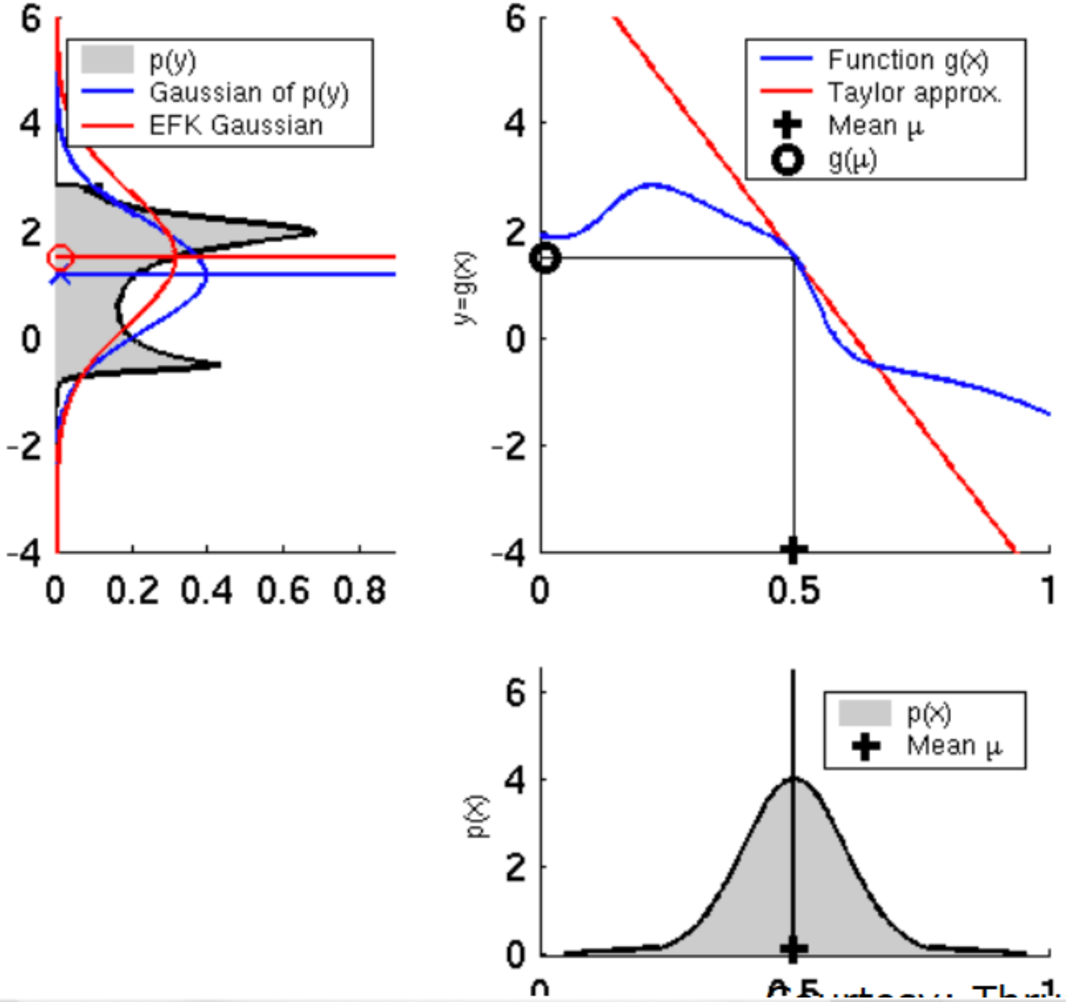


Credit C. Stachniss

LINEARITY ASSUMPTION

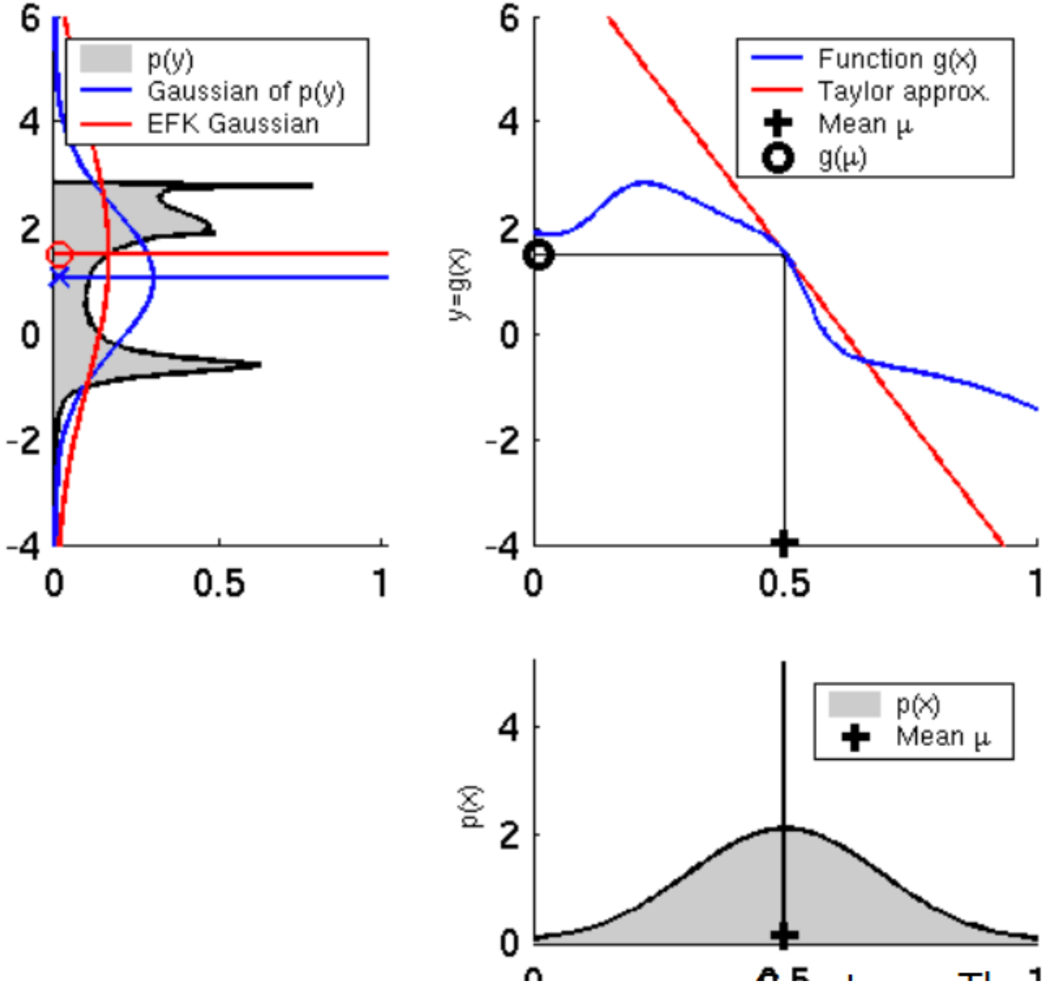


EKF LINEARISATION

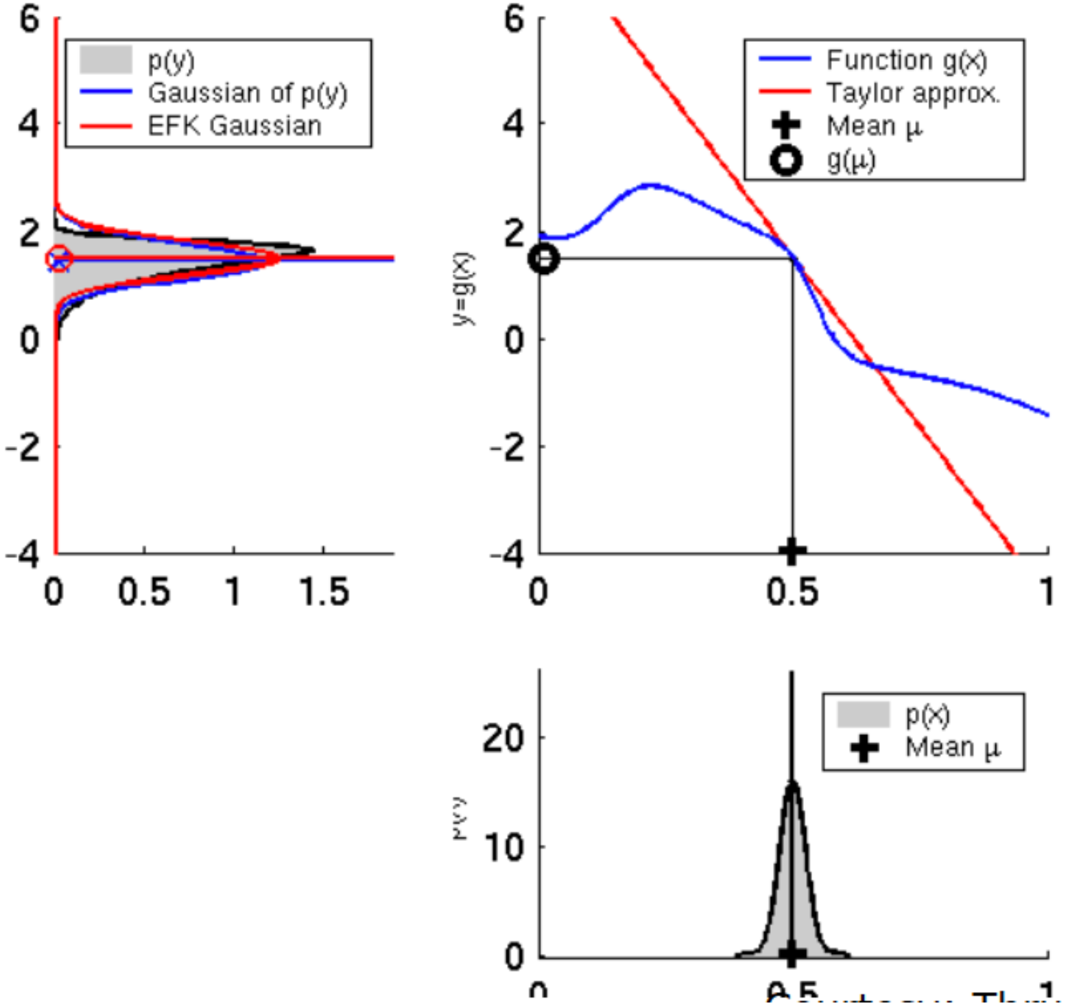


Credit C. Stachniss

EKF LINEARISATION



EKF LINEARISATION



Credit C. Stachniss

EKF ALGORITHM

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

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

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

9. Return μ_t, Σ_t

KF

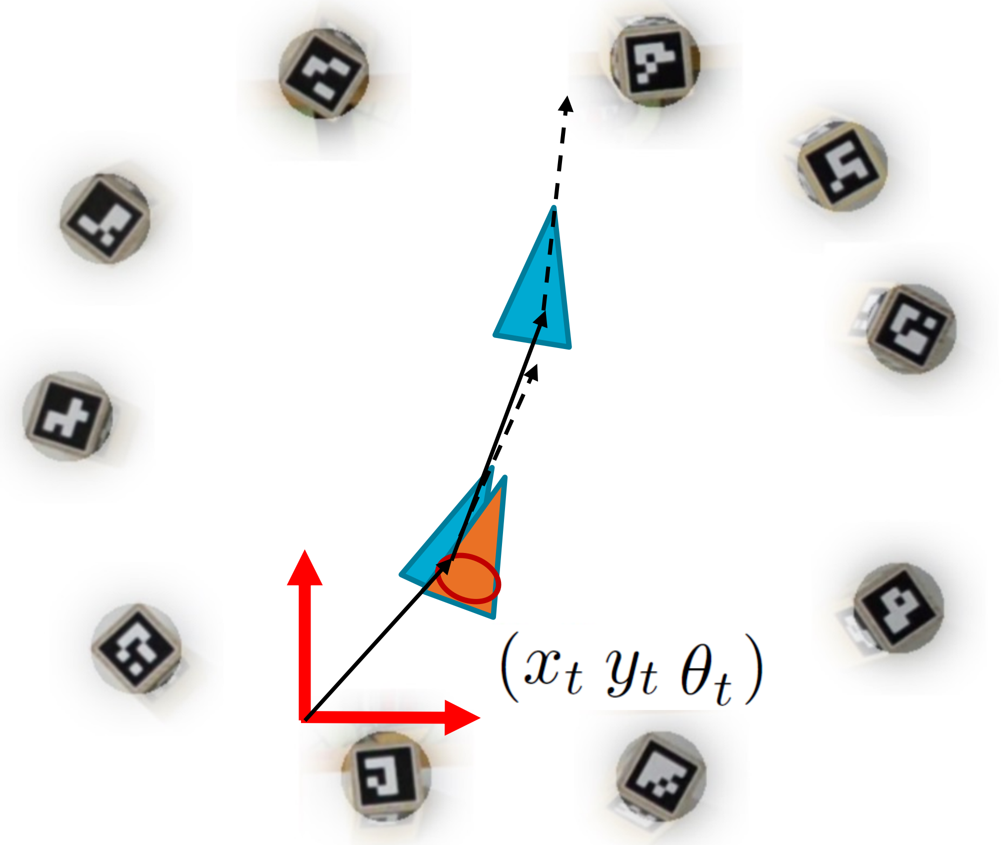
$$\longleftarrow \quad \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$
$$\longleftarrow \quad \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t$$
$$\longleftarrow \quad K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1}$$
$$\longleftarrow \quad \mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$$
$$\longleftarrow \quad \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$$

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t} \qquad G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}$$

2D LOCALISATION EKF

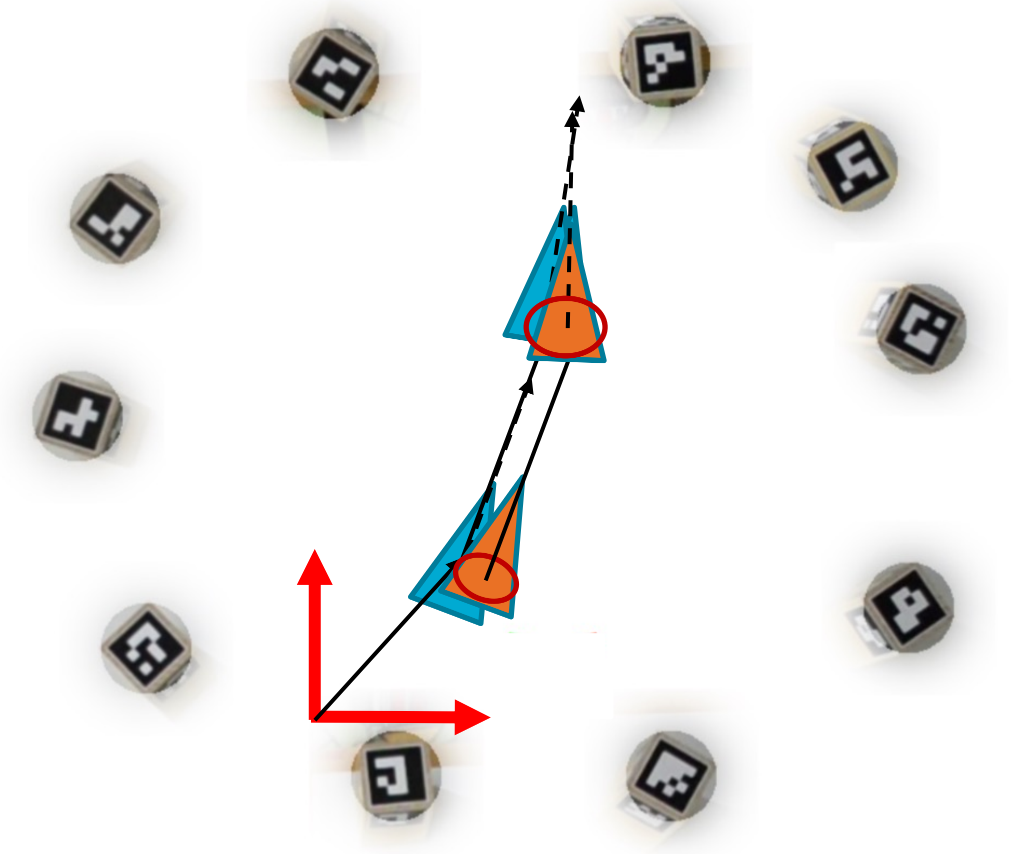
Initialisation

$$N(x_0; \mu_0, \Sigma_0)$$



2D LOCALISATION EKF

Prediction



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

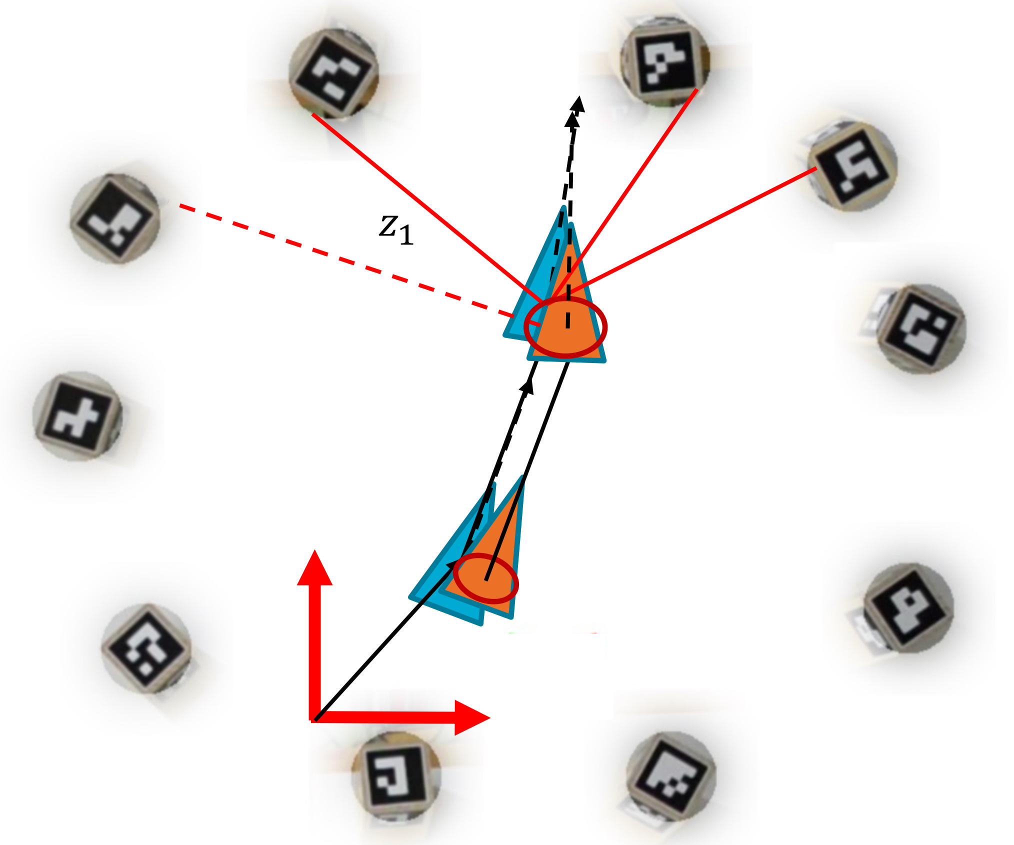
$$\theta_{t+1} = \theta_t + \omega dt$$
$$x_{t+1} = x_t + v/\omega(\sin\theta_{t+1} - \sin\theta_{t+1})$$
$$y_{t+1} = y_t + v/\omega(-\cos\theta_{t+1} + \cos\theta_{t+1})$$

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

$$\Sigma_t = G_t \Sigma_{t-1} G_t^T + Q_t$$

2D LOCALISATION EKF

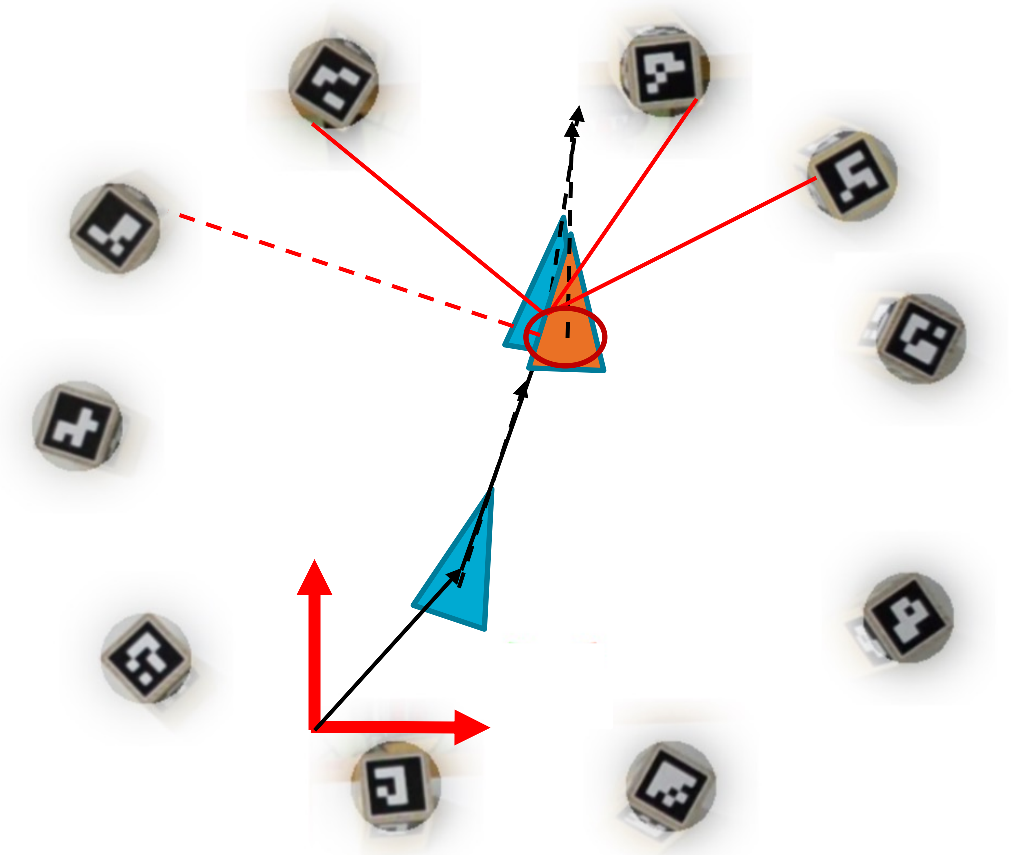
Observe



$$z = \{z_1, z_2, \ldots z_M\}$$

2D LOCALISATION EKF

Compute likelihood (expected measurements)



$$h(\bar{\mu}_t) = \{\hat{z}_1, \hat{z}_2, \ \ldots \ \hat{z}_M\}_i$$

$$\rho = \sqrt{(m_x - x)^2 + (m_y - y)^2}$$

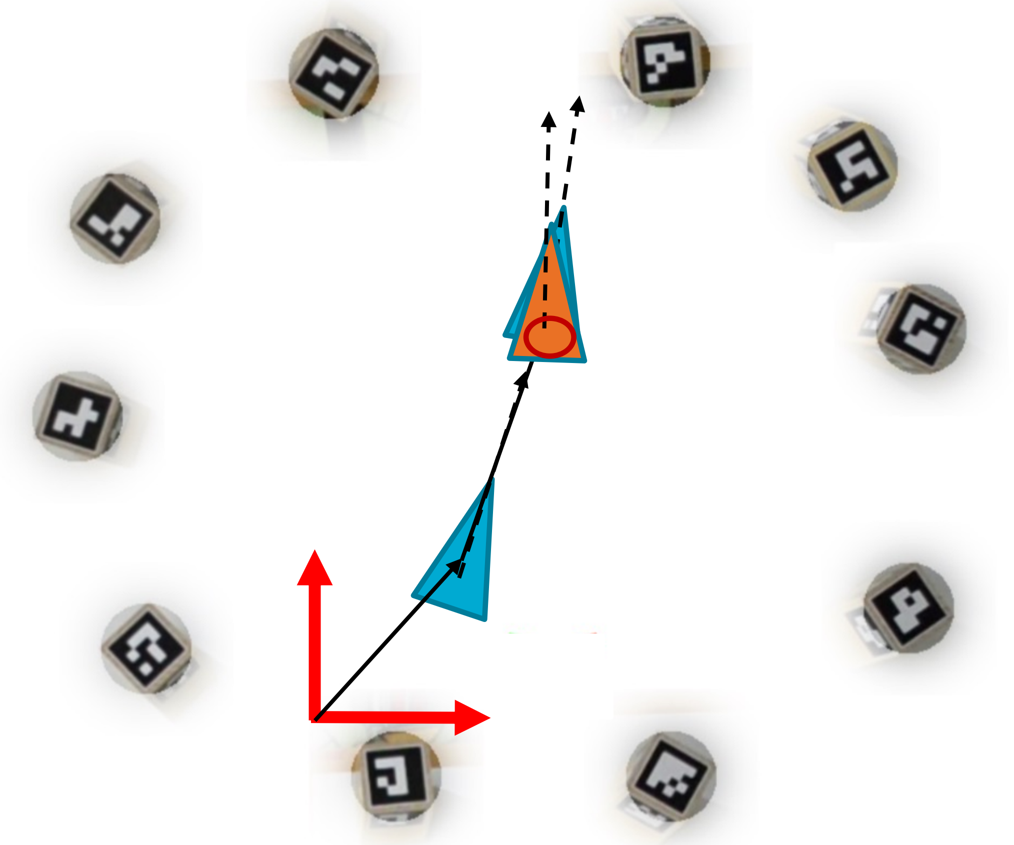
$$\alpha = atan2\left(m_y - y, m_x - x\right) - \theta$$

$$(z_t - h(\bar{\mu}_t))$$

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t}$$

2D LOCALISTION EKF

Update



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

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

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

SLAM

SLAM Problem: How to estimate the robot pose and at the same time the map of the environment from noisy sensor information?

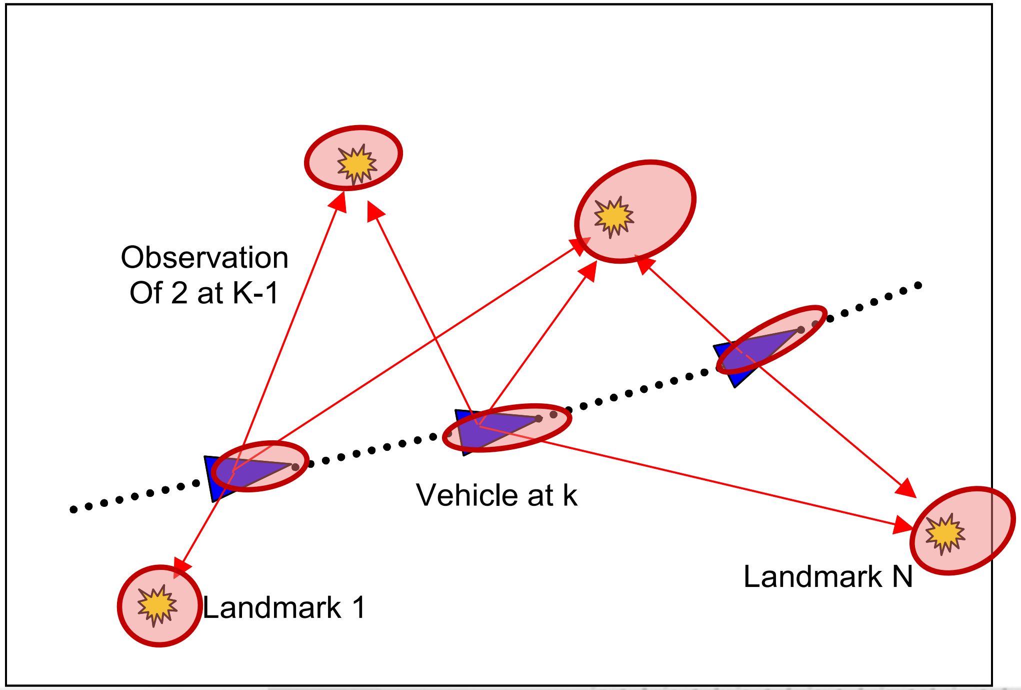
The standard method is based on probability theory to combine (FUSE) information from different noisy sensors

- True location unknown
- Map is also unknown

Chicken and Egg problem

If we have a map:
We can localise

If we can localise:
We can make a map



SLAM SOLUTIONS

- Filter– treats SLAM problem as a recursive state estimation problem of a dynamic system (estimate the current robot pose and position of landmarks observed so far) – EKF

Recursive

- Smoothing – treats SLAM problem as an optimisation problem – Maximum a Posteriori estimation (find the best configuration: all the robot poses and landmark positions) - NLLS

Batch

DEFINITION OF THE SLAM PROBLEM

Given

- The robot's controls

$$u_{1:T} = \{u_1, u_2, u_3, \ldots, u_T\}$$

- Observations

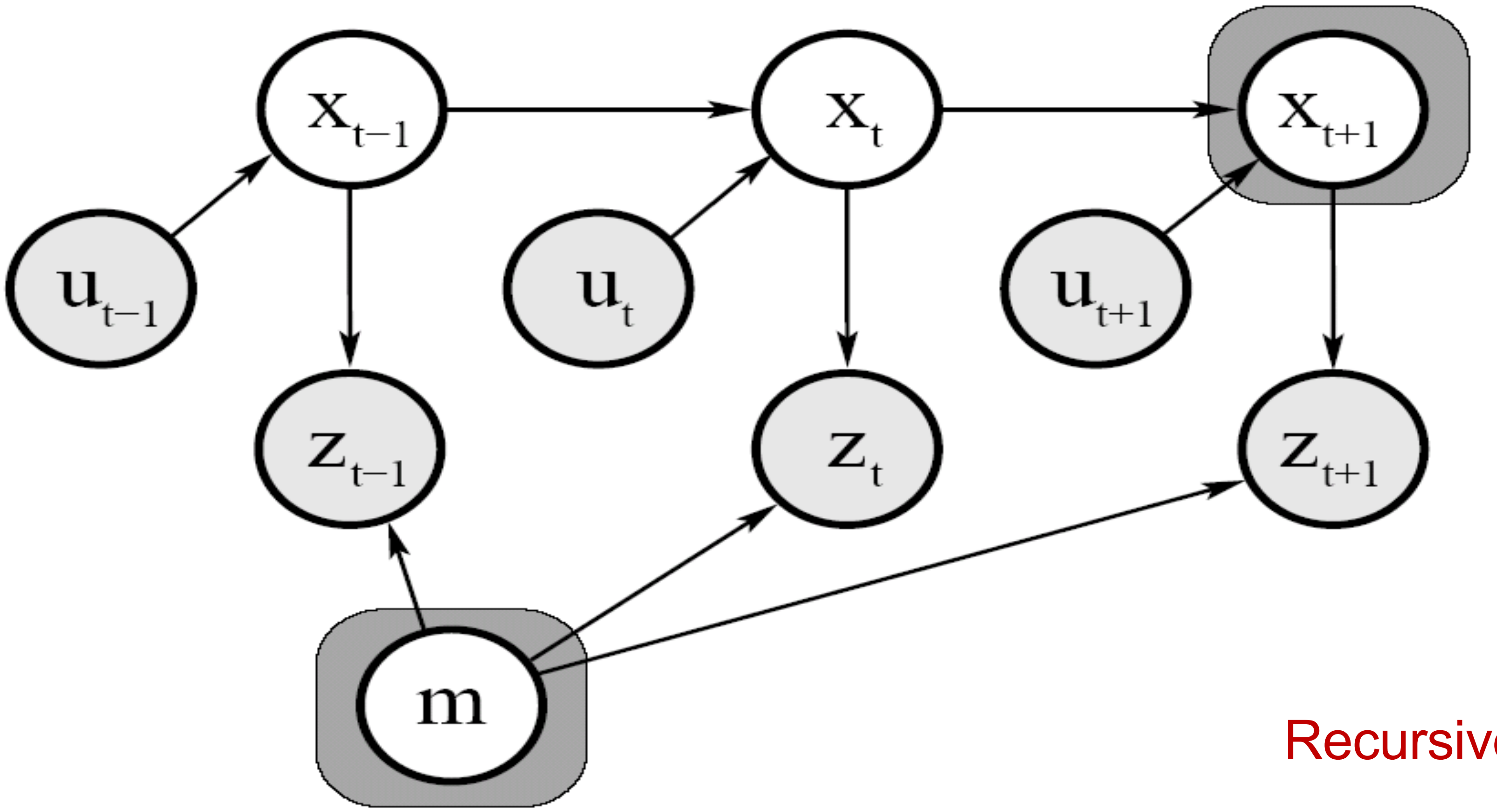
$$z_{1:T} = \{z_1, z_2, z_3, \ldots, z_T\}$$

Wanted

- Map of the environment m
- Path of the robot

$$x_{0:T} = \{x_0, x_1, x_2, \ldots, x_T\}$$

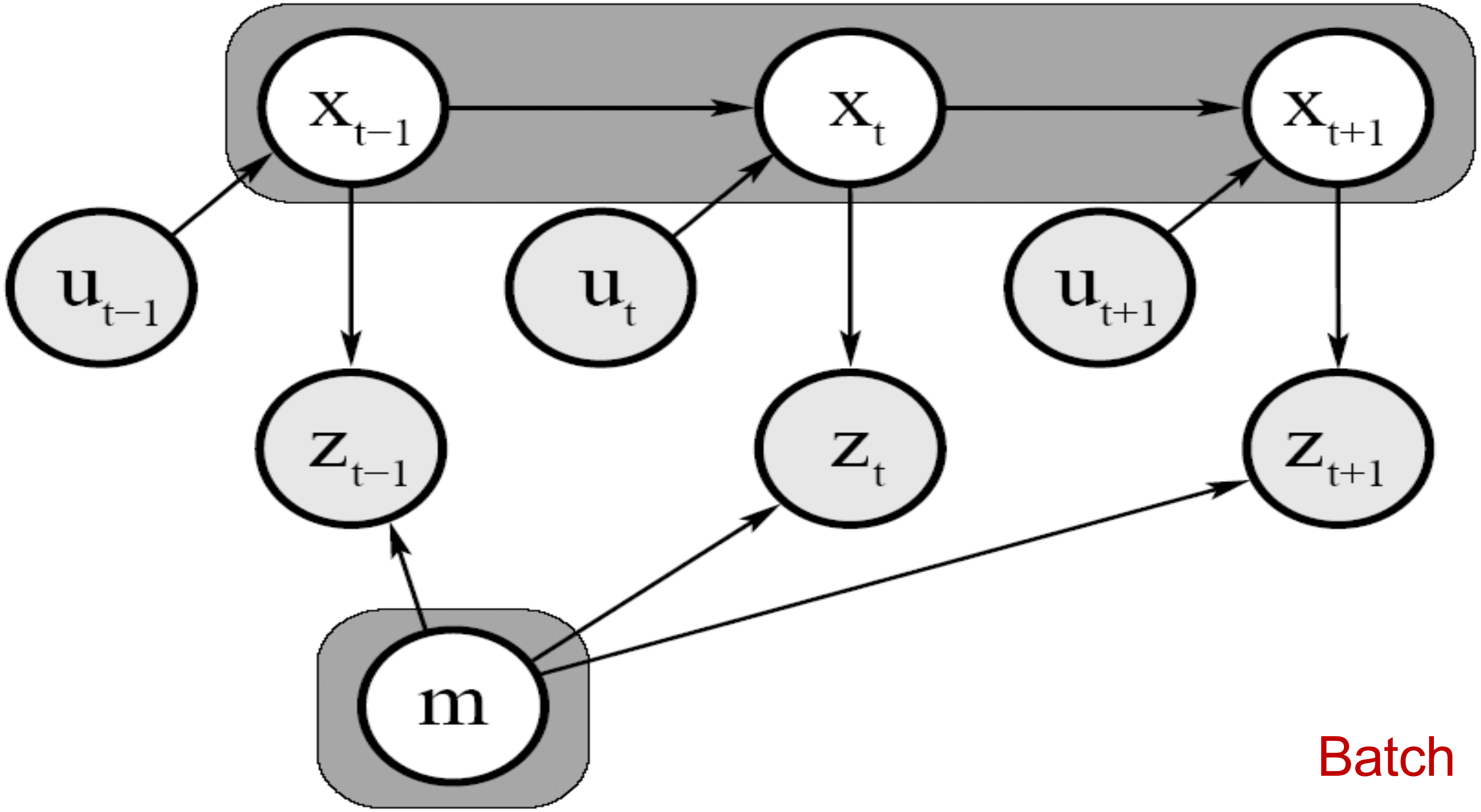
GRAPHICAL MODEL OF EKF SLAM



$$p(x_t, m \mid z_{1:t}, u_{1:t})$$

Recursive

GRAPHICAL MODEL OF NLLS-SLAM



Batch

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$

EKF-SLAM

Application of the EKF to SLAM

- Estimate robot's pose and locations of landmarks in the environment
- Assumption: known correspondences
- State space (for the 2D plane) is

$$x_t = (\underbrace{x, y, \theta}_{\text{robot's pose}}, \underbrace{m_{1,x}, m_{1,y}}_{\text{landmark 1}}, \ldots, \underbrace{m_{n,x}, m_{n,y}}_{\text{landmark n}})^T$$

EKF-SLAM: STATE REPRESENTATION

Map with n landmarks: (3+2n)- dimensional Gaussian

- Belief is represented by

$$\underbrace{\begin{pmatrix} x \\ y \\ \theta \\ m_{1,x} \\ m_{1,y} \\ \vdots \\ m_{n,x} \\ m_{n,y} \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \cdots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} & \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \cdots & \sigma_{m_{n,x}} & \sigma_{m_{n,y}} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} & \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \cdots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\ \sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{\theta} & \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \cdots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\ \sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{\theta} & \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \cdots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{\theta} & \sigma_{m_{n,x}m_{1,x}} & \sigma_{m_{n,x}m_{1,y}} & \cdots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\ \sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{\theta} & \sigma_{m_{n,y}m_{1,x}} & \sigma_{m_{n,y}m_{1,y}} & \cdots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}} \end{pmatrix}}_{\Sigma}$$

$$x_t = (\underbrace{x, y, \theta}_{\text{robot's pose}}, \underbrace{m_{1,x}, m_{1,y}}_{\text{landmark 1}}, \ldots, \underbrace{m_{n,x}, m_{n,y}}_{\text{landmark n}})^T$$

EKF-SLAM: STATE REPRESENTATION

More compactly

$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n} \end{pmatrix}}_{\Sigma}$$

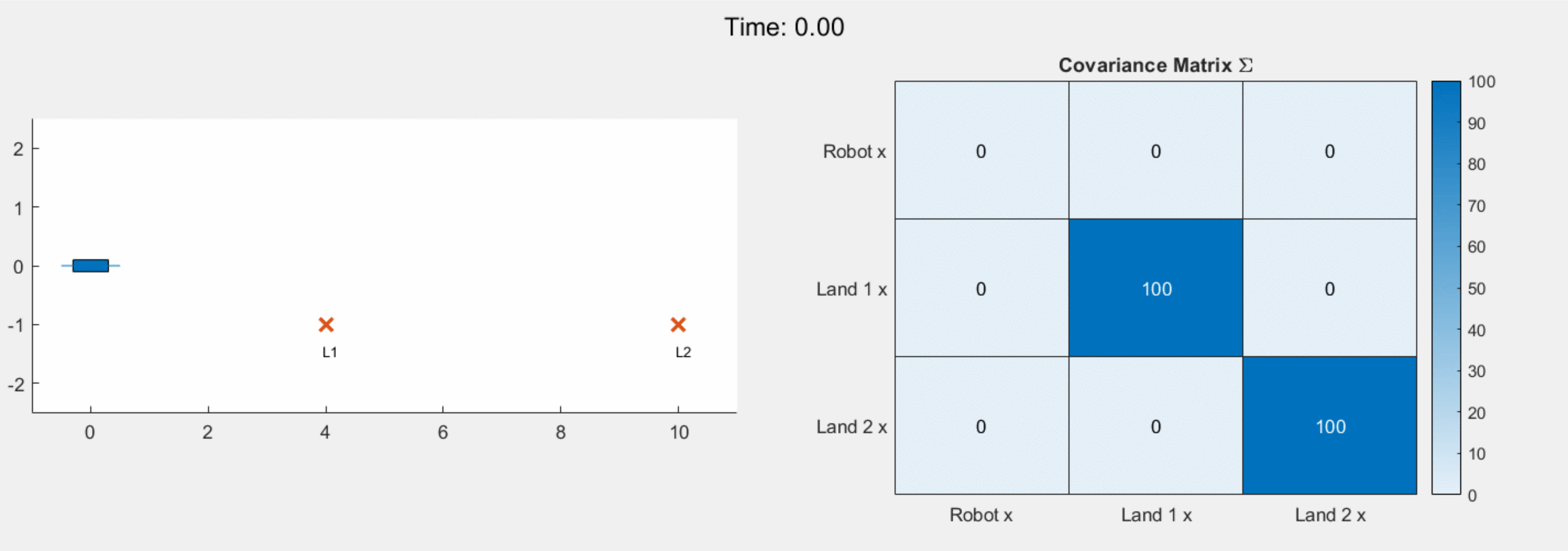
EKF-SLAM: STATE REPRESENTATION

Even more compactly

$$x_R \rightarrow x$$

$$\underbrace{\begin{pmatrix} x \\ m \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{pmatrix}}_{\Sigma}$$

EKF-SLAM: CORRELATION VISUALISATION



Credit Jacob Higgings

EKF-SLAM: INITIALISATION

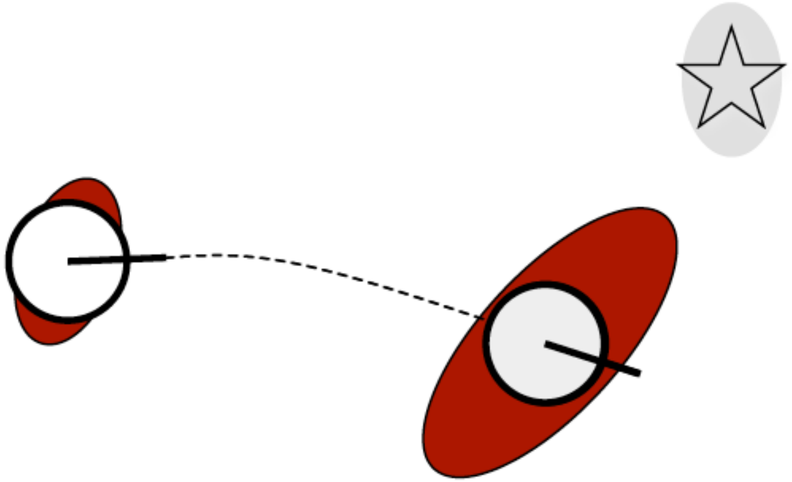
Robot starts in its own reference frame (all landmarks unknown)

- 2N+3 dimensions

$$\mu_0 = \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 \end{pmatrix}^T$$

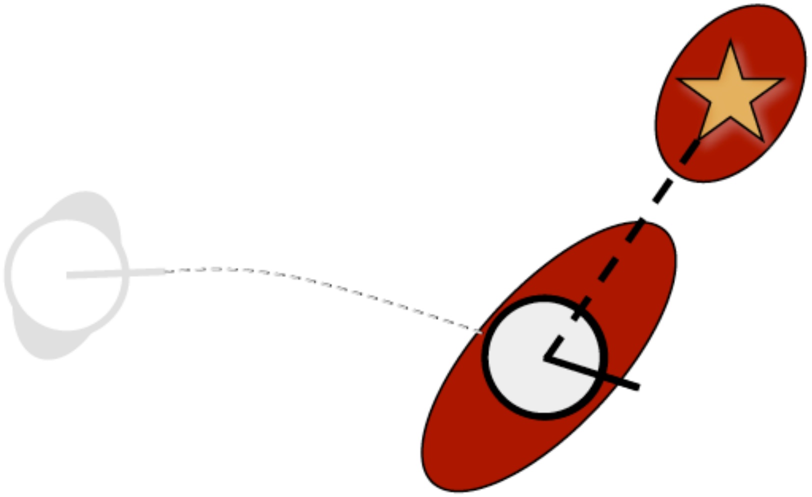
$$\Sigma_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \infty & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \infty \end{pmatrix}$$

EKF-SLAM: STATE PREDICTION



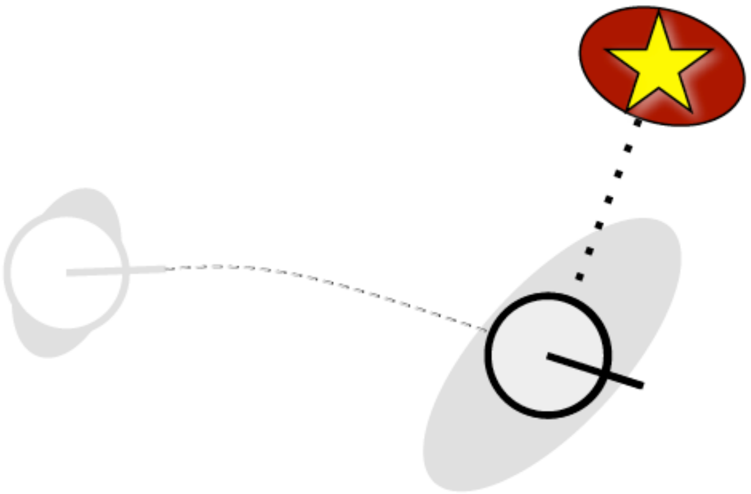
$$
\underbrace{\begin{pmatrix} \boxed{x_R} \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu}
\quad
\underbrace{\begin{pmatrix}
\Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\
\Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\
\vdots & \vdots & \ddots & \vdots \\
\Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n}
\end{pmatrix}}_{\Sigma}
$$

EKF-SLAM: MEASUREMENT PREDICTION



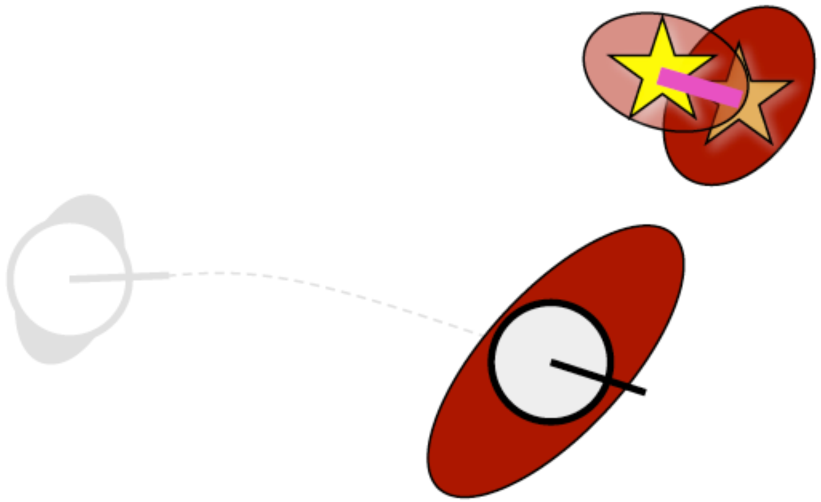
$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n} \end{pmatrix}}_{\Sigma}$$

EKF-SLAM: OBTAINED MEASUREMENT



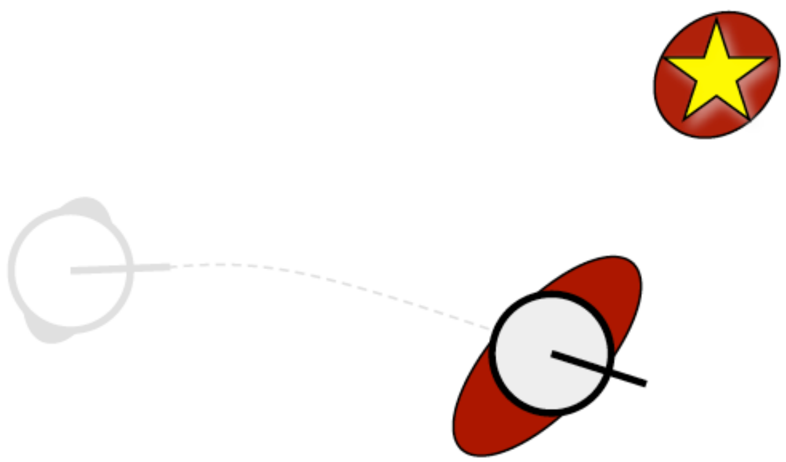
$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n} \end{pmatrix}}_{\Sigma}$$

DATA ASSOCIATION - INNOVATION



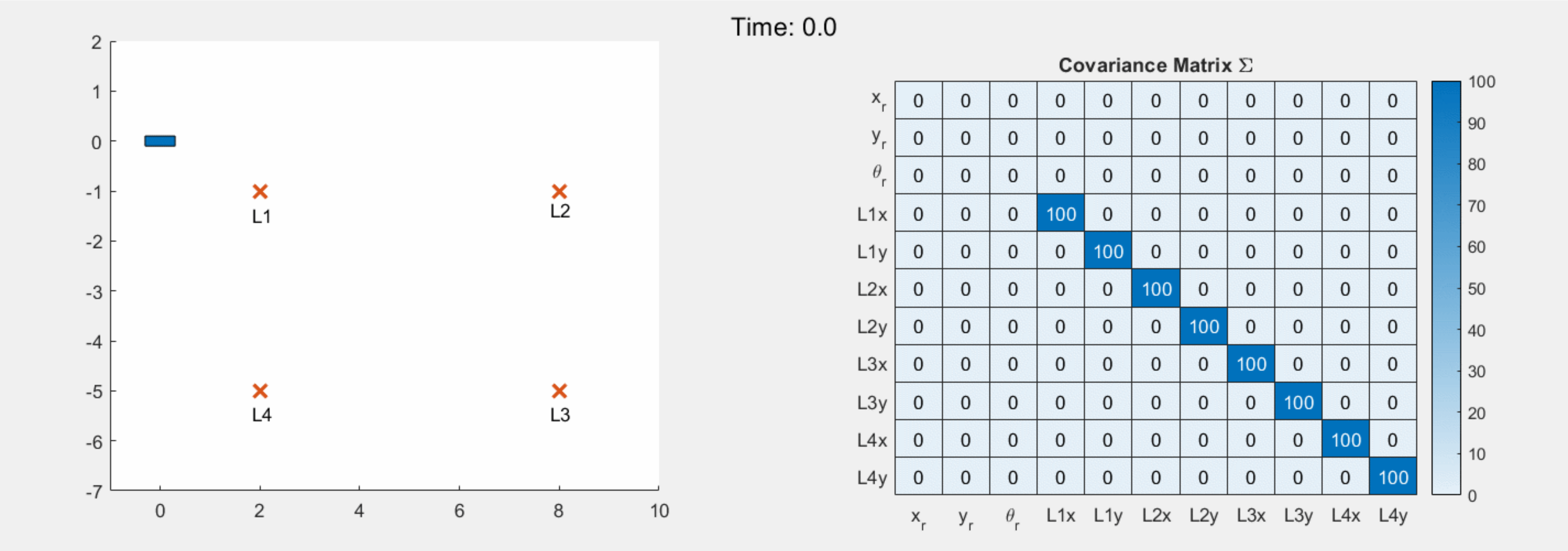
$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n} \end{pmatrix}}_{\Sigma}$$

EKF-SLAM: UPDATE STEP



$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n} \end{pmatrix}}_{\Sigma}$$

EKF-SLAM 2D – CORRELATION VISUALISATION



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

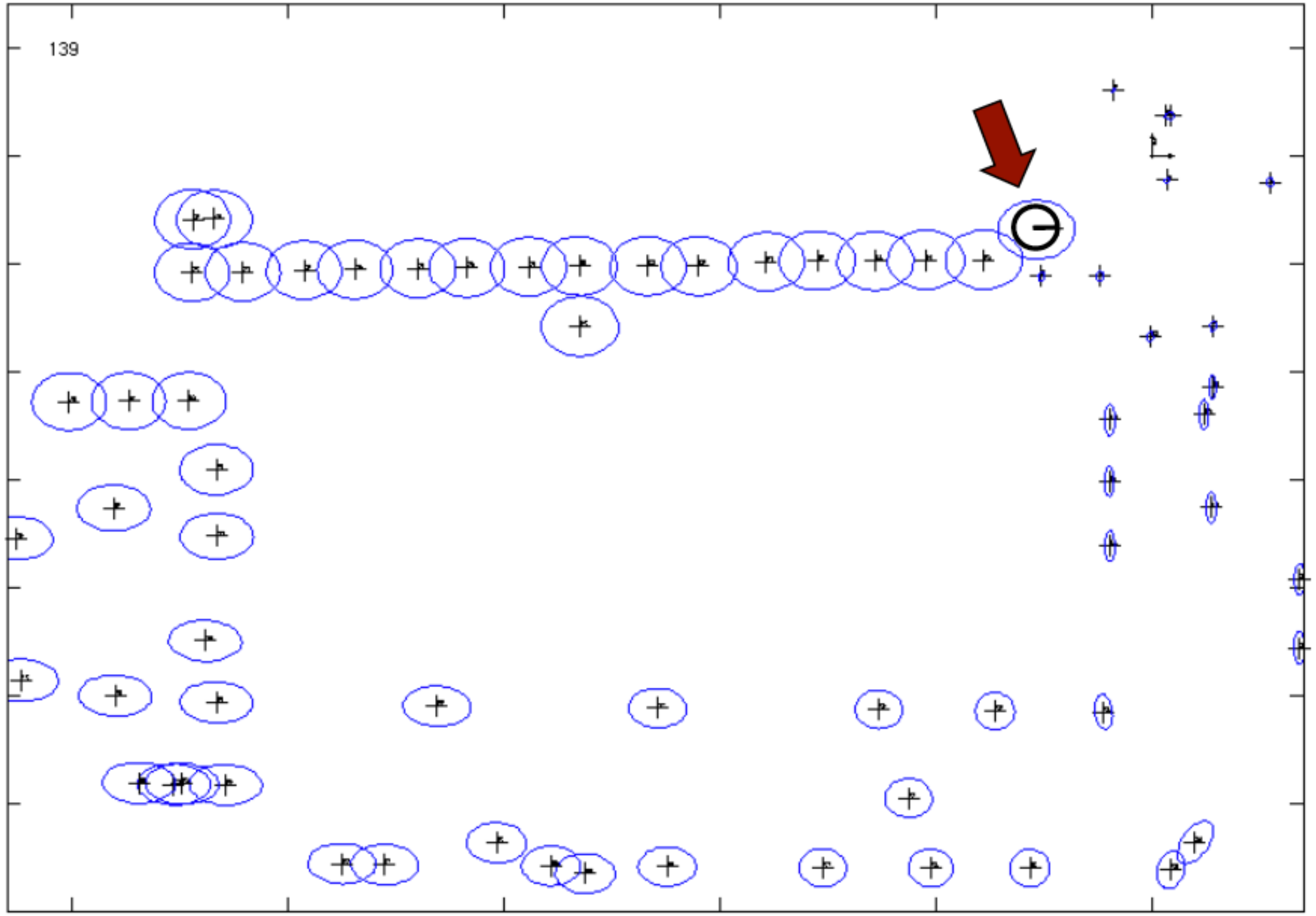
7: *return* μ_t, Σ_t

LOOP CLOSURE

Loop closing means recognizing an already mapped area

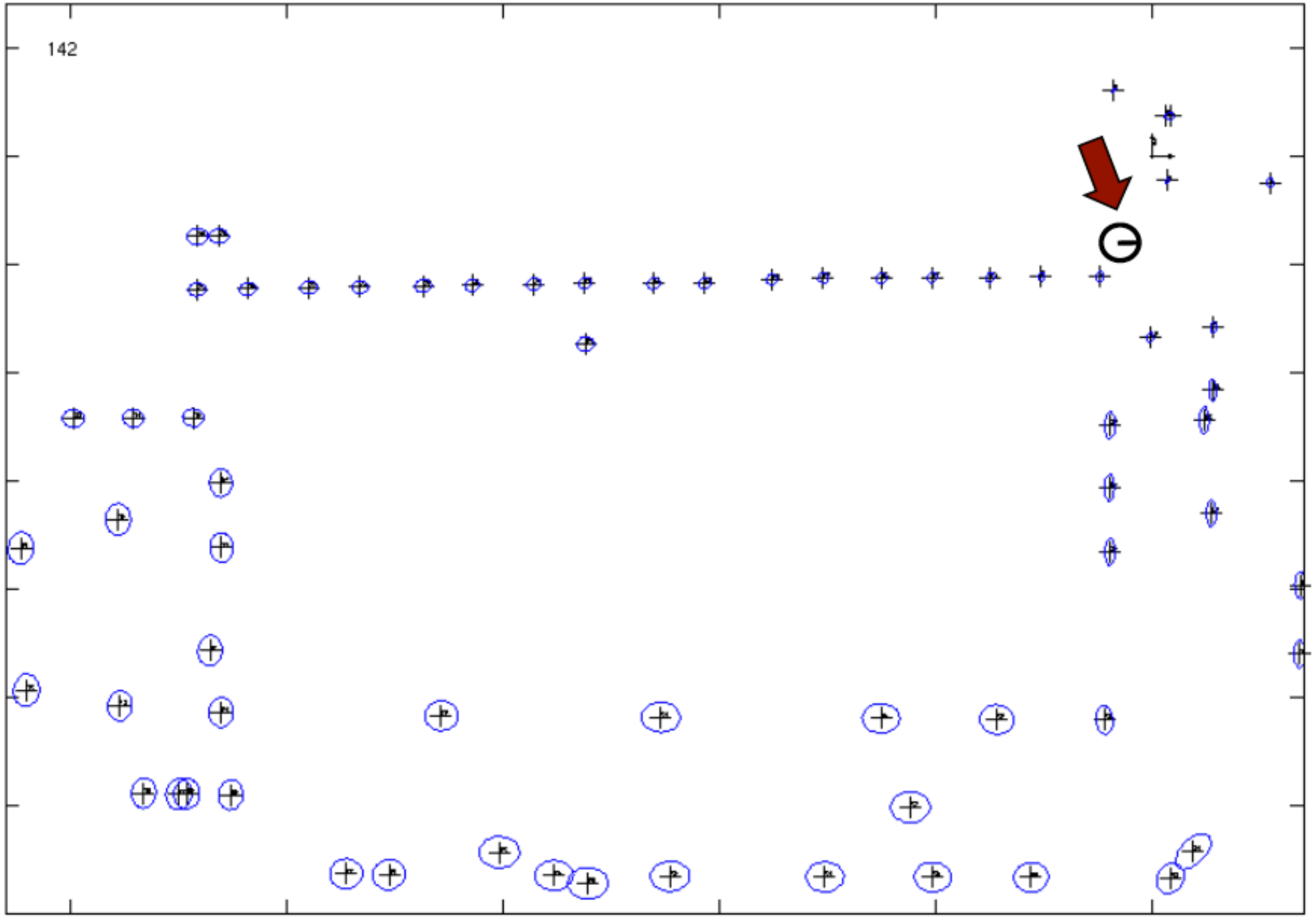
- Data association under
- high ambiguity
- possible environment symmetries
- Uncertainties collapse after a loop closure (whether the closure was correct or not)

LOOP CLOSURE



Courtesy of K. Arras

LOOP CLOSURE



Courtesy of K. Arras

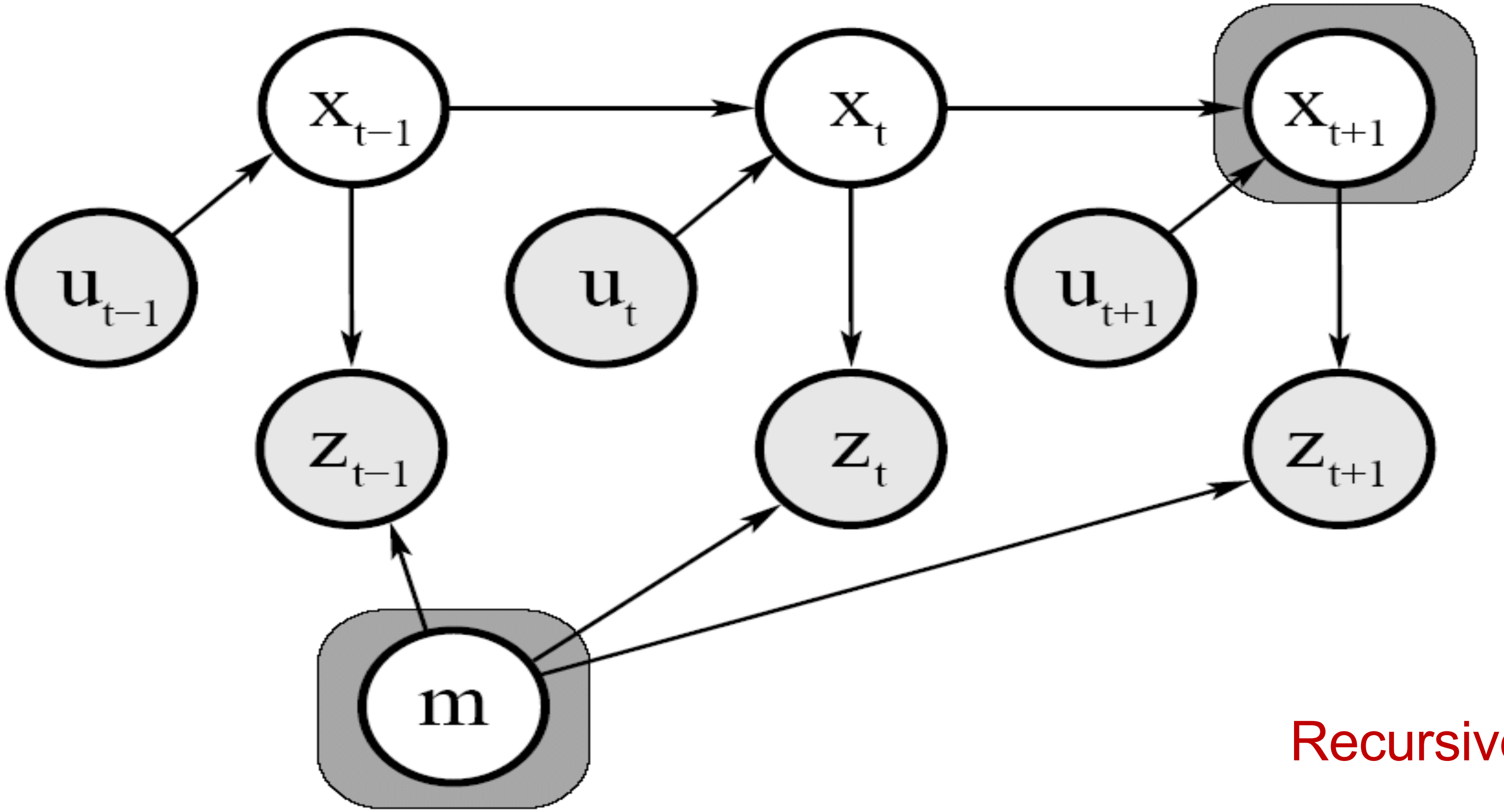
LOOP CLOSURE IN SLAM

Loop closing reduces the uncertainty in robot and landmark estimates

- This can be exploited when exploring an environment for the sake of better (e.g. more accurate) maps

- Wrong loop closures lead to filter divergence

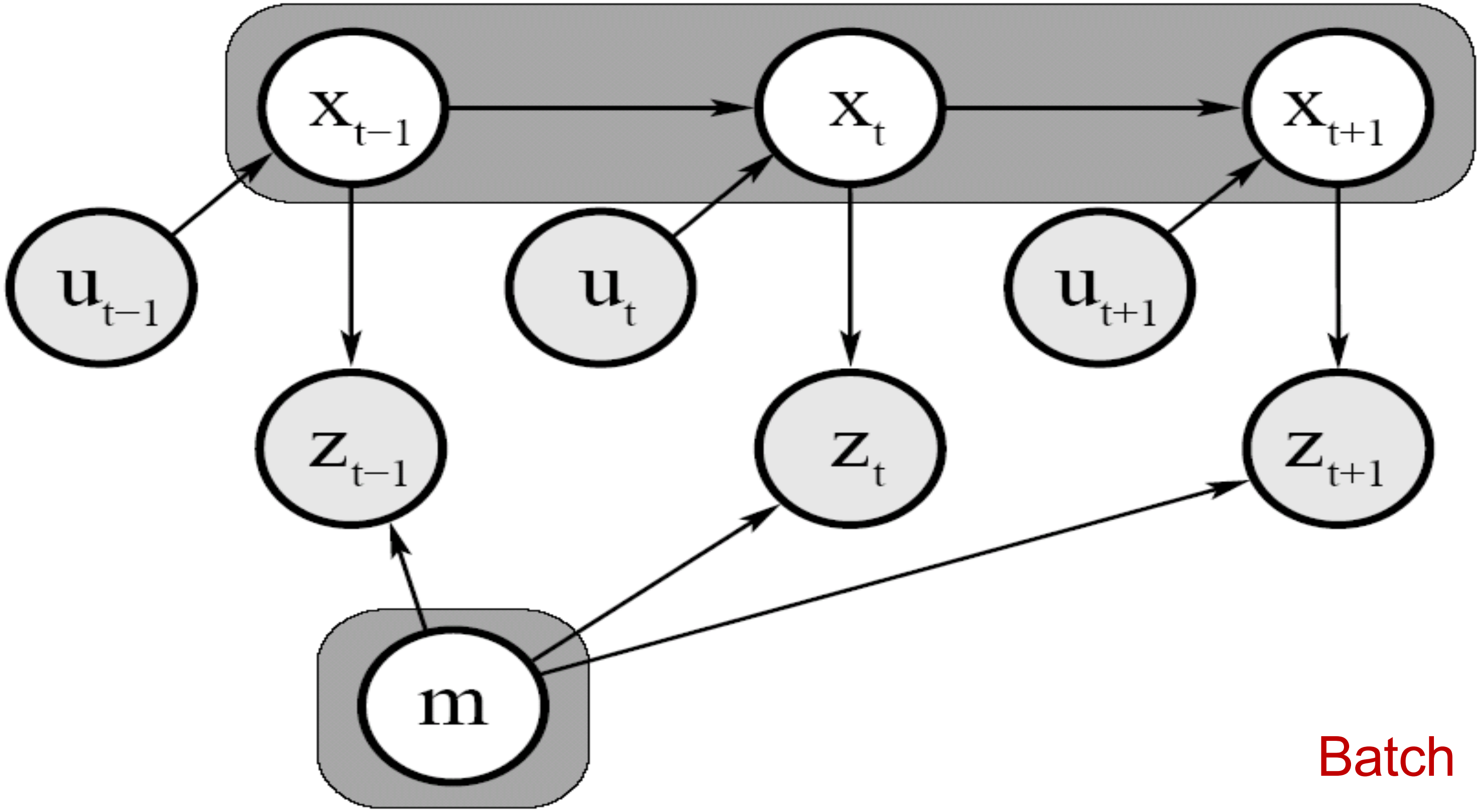
GRAPHICAL MODEL OF EKF SLAM



$$p(x_t, m \mid z_{1:t}, u_{1:t})$$

Recursive

GRAPHICAL MODEL OF NLLS-SLAM



Batch

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$

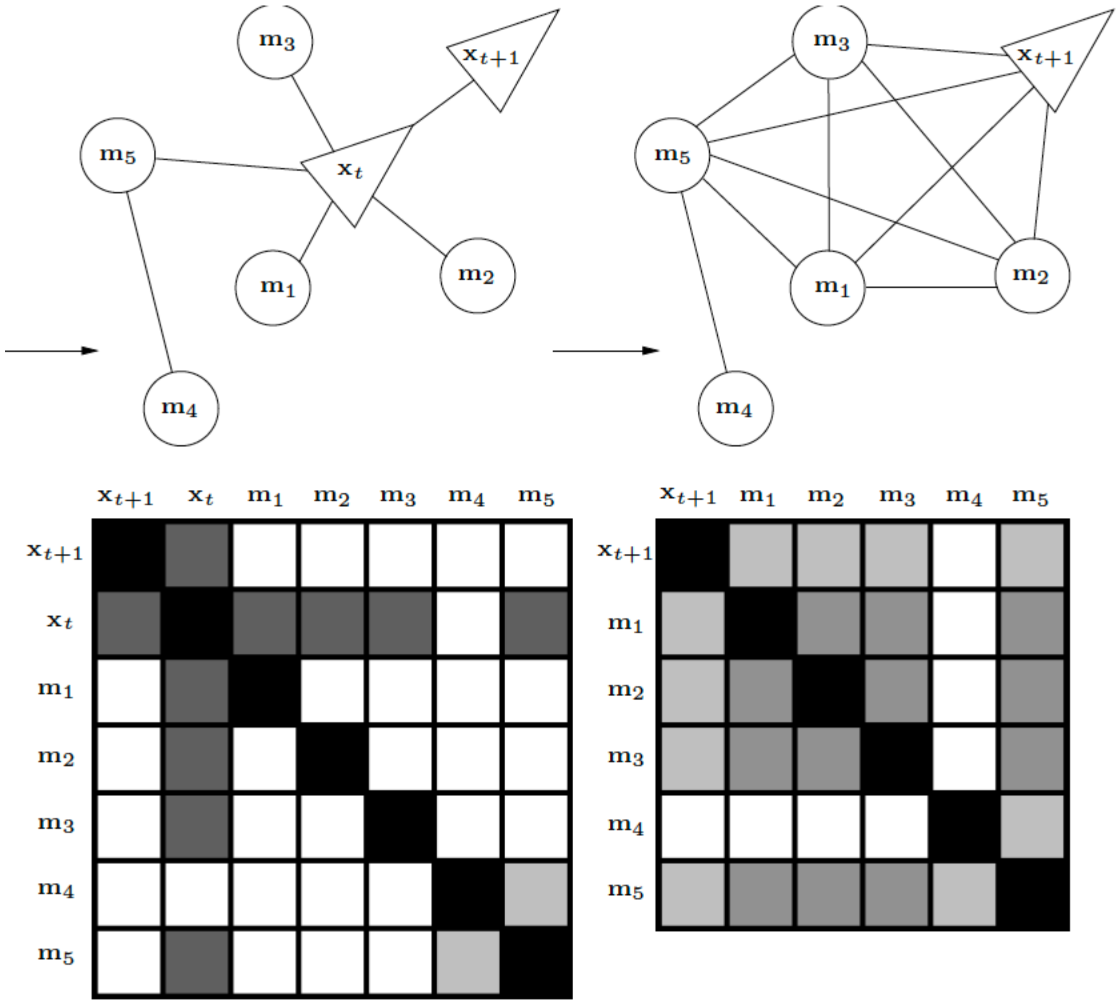
FILTERING VS SMOOTHING

MAP or NLLS or Smoothing (estimate entire trajectory and map)

- Many variables
- Information matrix is sparse

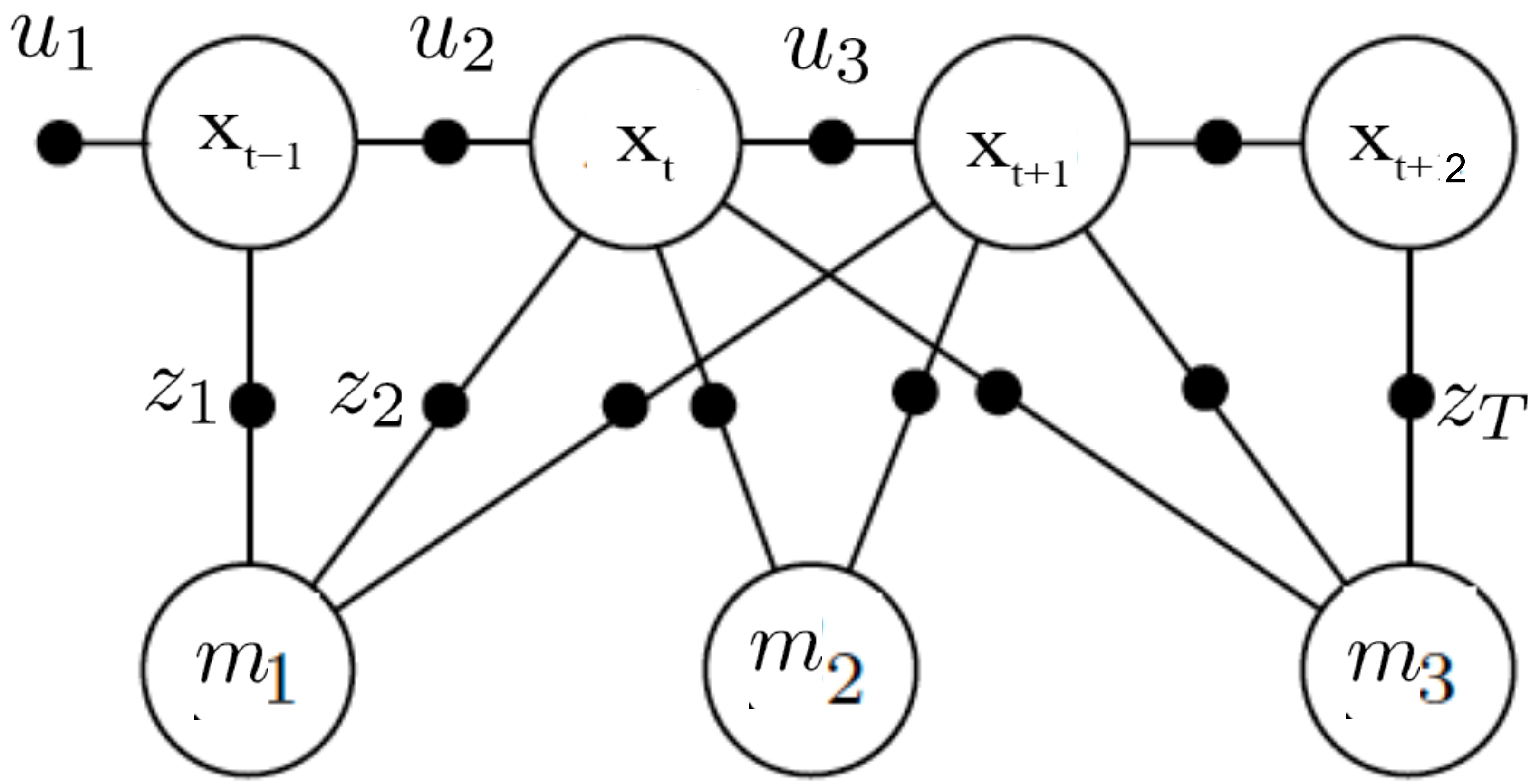
Filtering (estimate only current pose and landmarks)

- Marginalise out ALL old pose states (hence few variables)
- Covariance matrix after Schur complement is typically dense

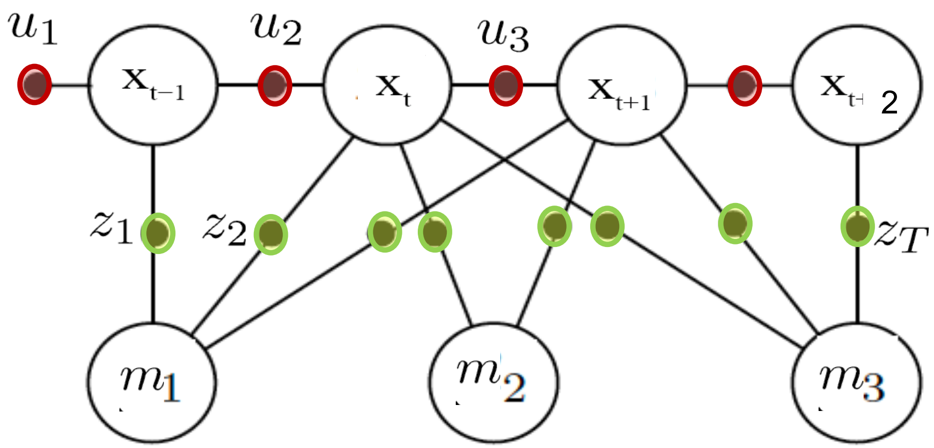


Walter et al, 2007

FACTOR GRAPHS IN SLAM



FACTOR GRAPHS IN SLAM - NLLS



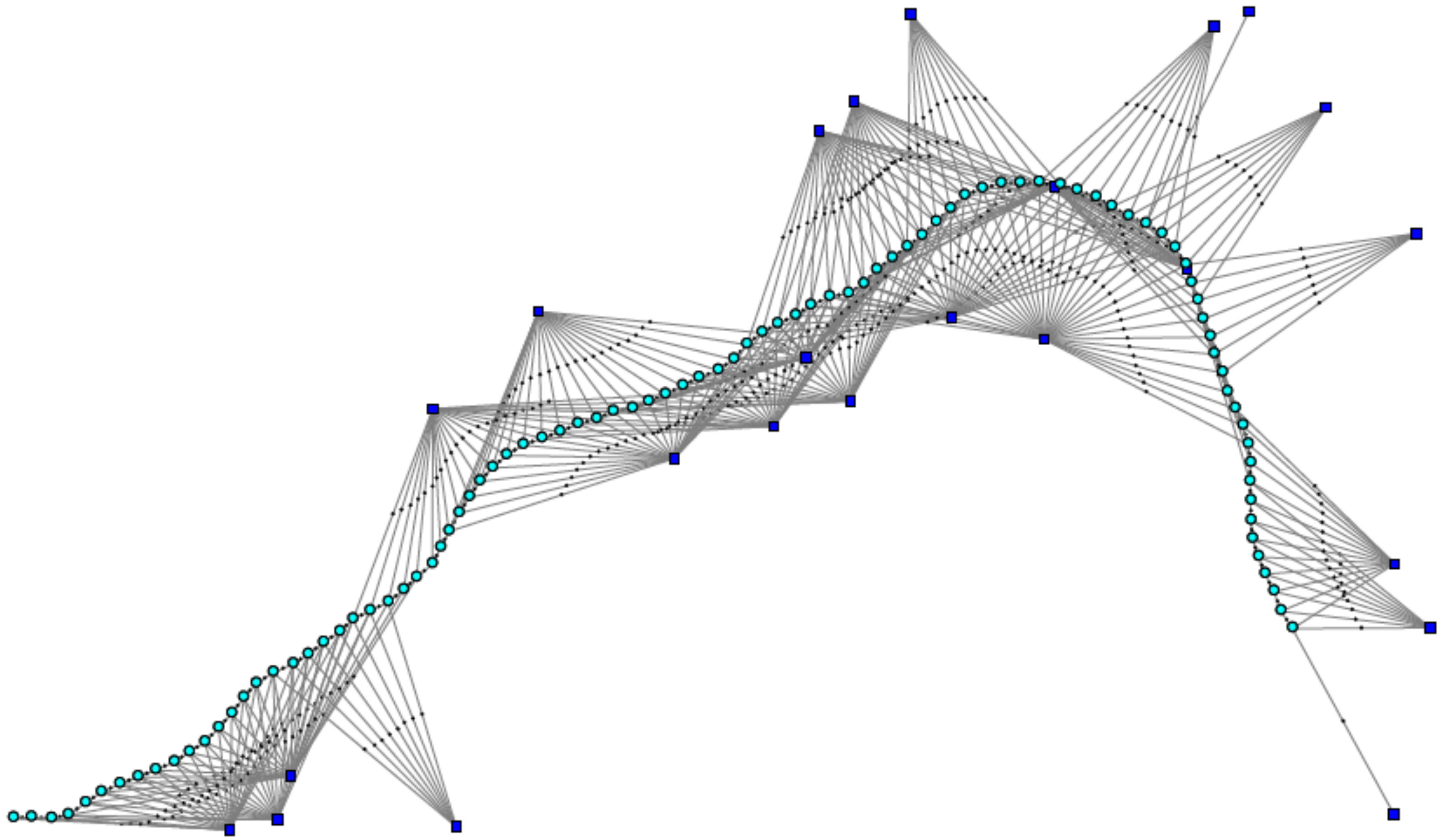
Maximum A-Posteriori Estimation (MAP)

$$\mathcal{S}^* = \underset{\mathcal{S}}{\operatorname{argmin}} \; -\log(p(\boldsymbol{x}|\boldsymbol{z})) = \underset{\mathcal{S}}{\operatorname{argmin}} \; J(\boldsymbol{x})$$

$$J(\boldsymbol{x}) \triangleq \sum_i \|\boldsymbol{x}_{t+1} - \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{u}_t)\|^2_{\boldsymbol{Q}} + \sum_i \|\boldsymbol{z}_i - \boldsymbol{h}_i(\boldsymbol{x}_i)\|^2_{\boldsymbol{\Sigma}_i}$$

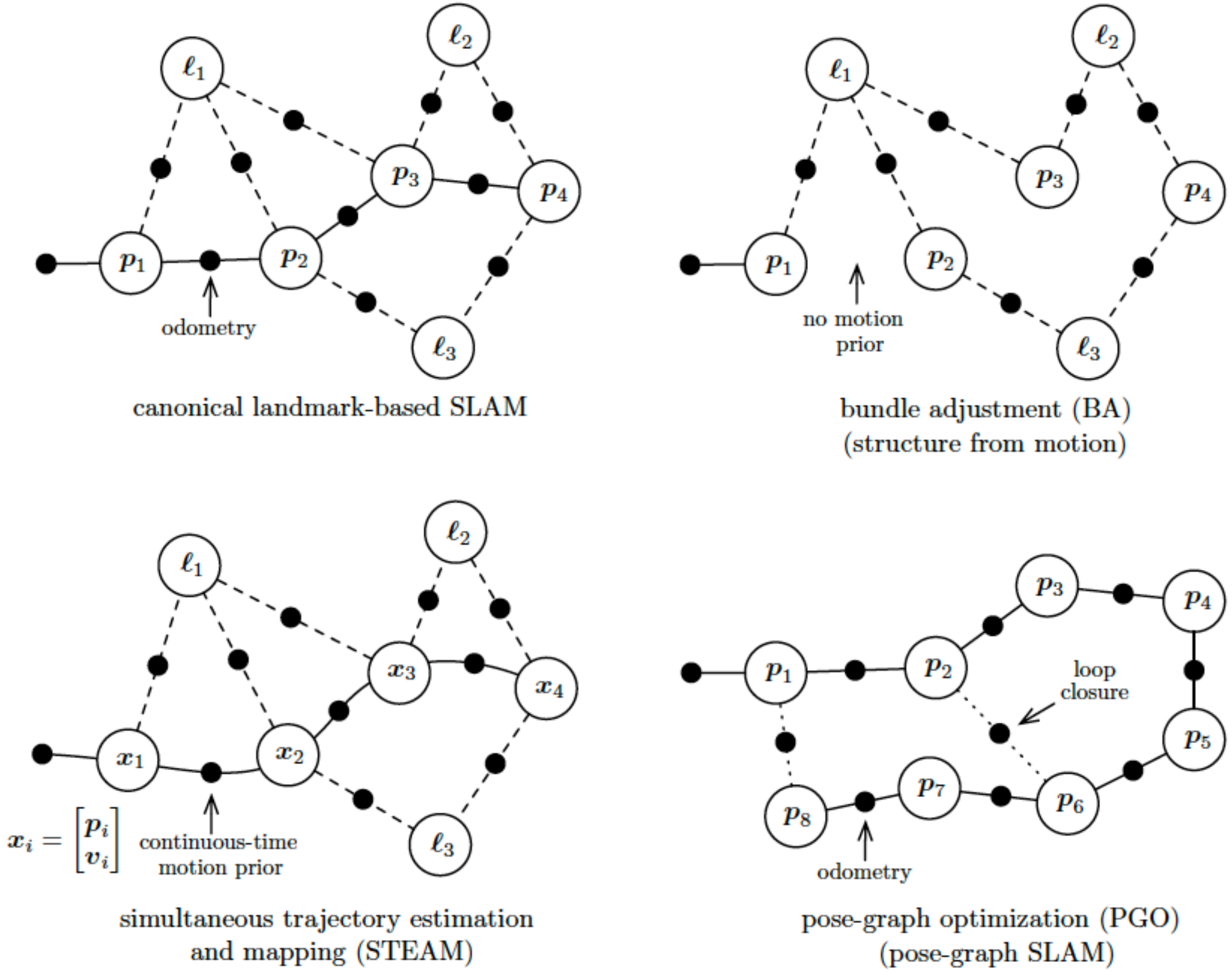
Maximum Likelihood Estimation (MLE)

FACTOR GRAPHS IN SLAM



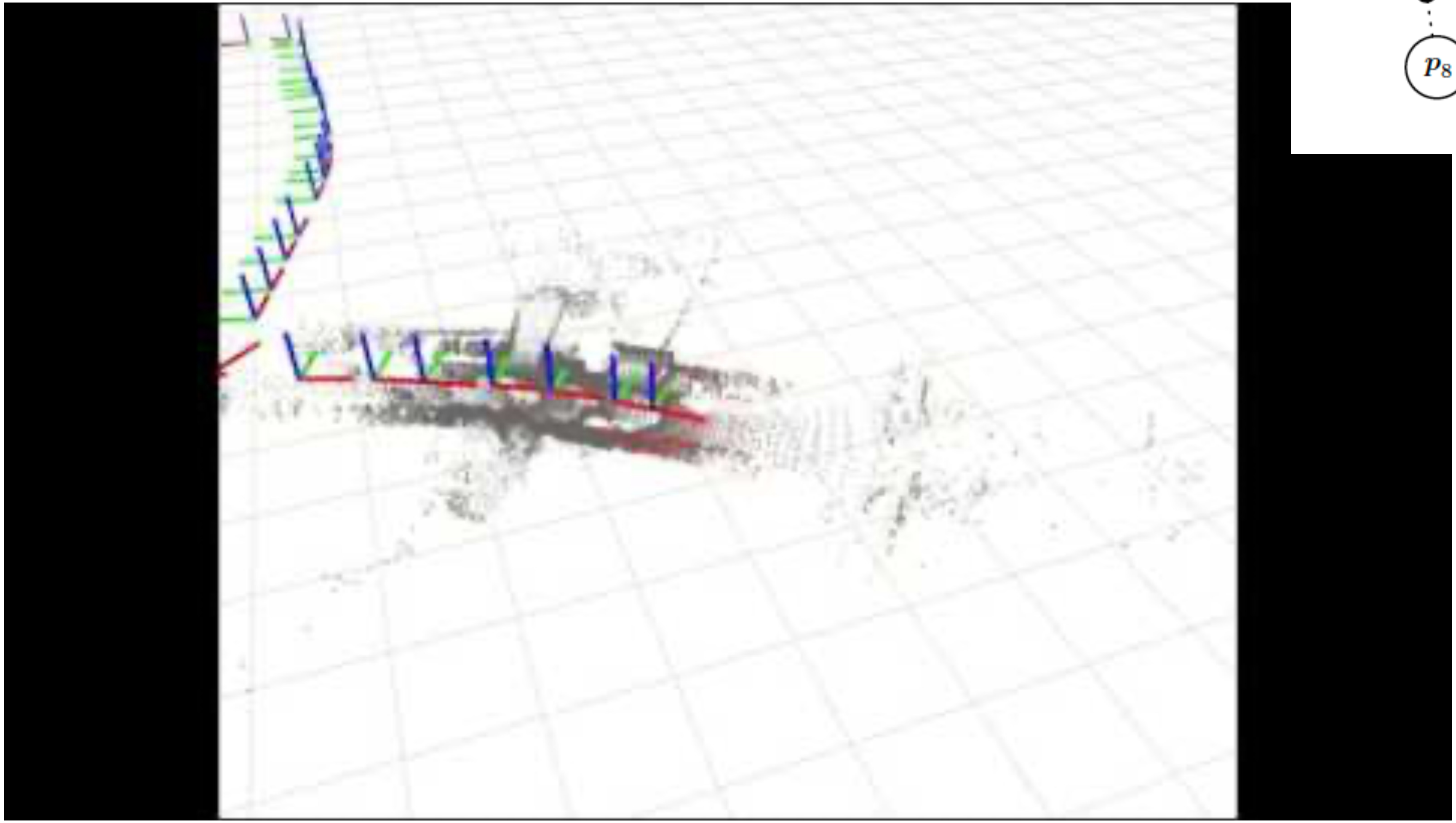
Barfoot et al 2023

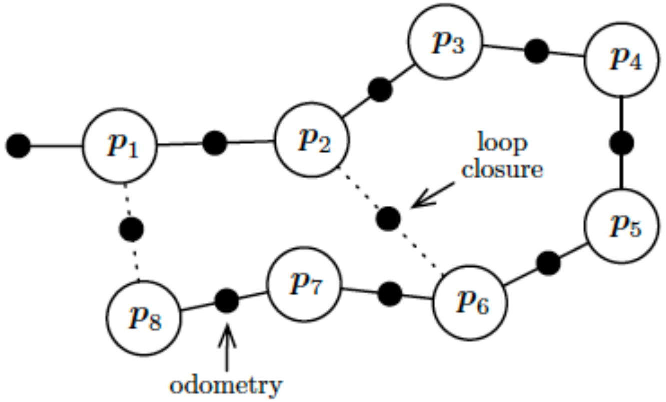
FACTOR GRAPHS IN SLAM



Barfoot et al 2023

POSE GRAPH SLAM





https://www.youtube.com/watch?v=KYvOqUB_odg

SLAM IN CHALLENGING ENVIRONMENTS

Test scenario: Mt. Etna, Sicily

Guibilato et al, FR 2022