академия больших данных

mail.ru group

girafe ai

Introduction to Mobile Robotics course, Lecture 2

Localization

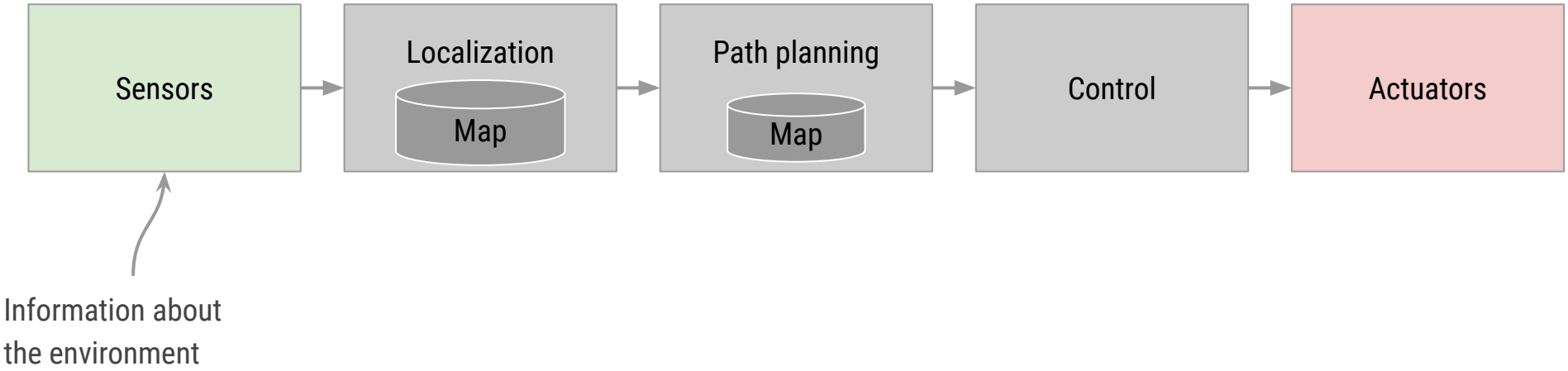
Vladislav Goncharenko, Fall 2021

Materials by Oleg Shipitko

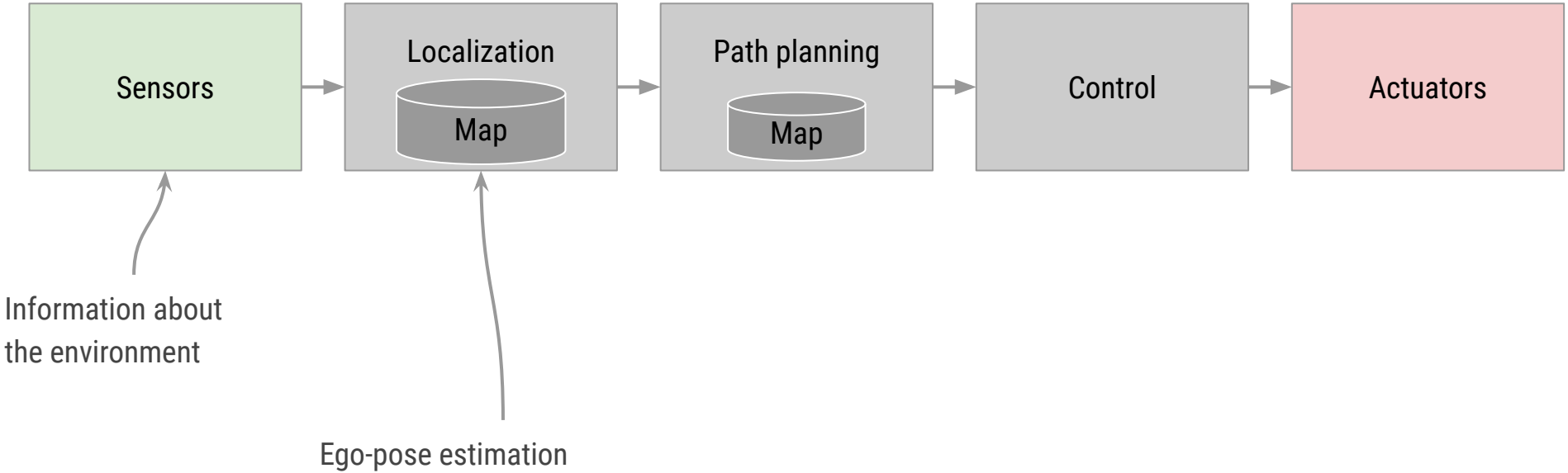
Outline

1. Localization problem definition
2. Probabilistic form of localization problem
    a. Recursive Bayesian estimation
3. Kalman filter
4. Particle filter

(SIMPLIFIED) CONTROL SCHEME OF MODERN MOBILE ROBOT



(SIMPLIFIED) CONTROL SCHEME OF MODERN MOBILE ROBOT



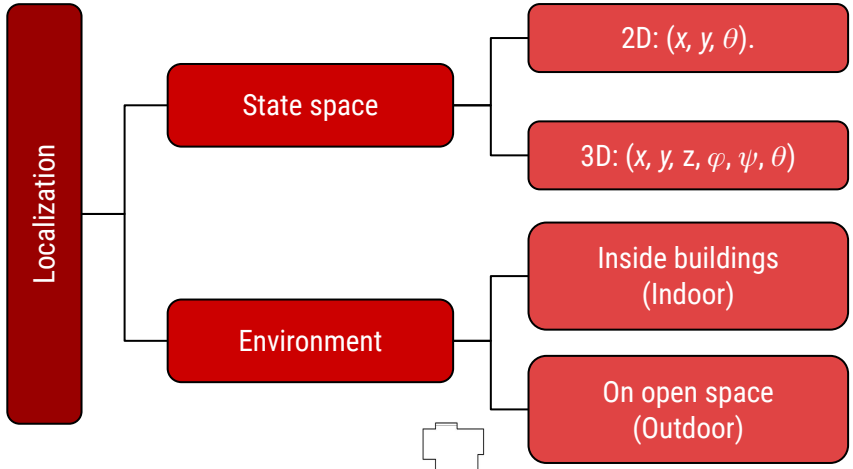
Localization problem definition

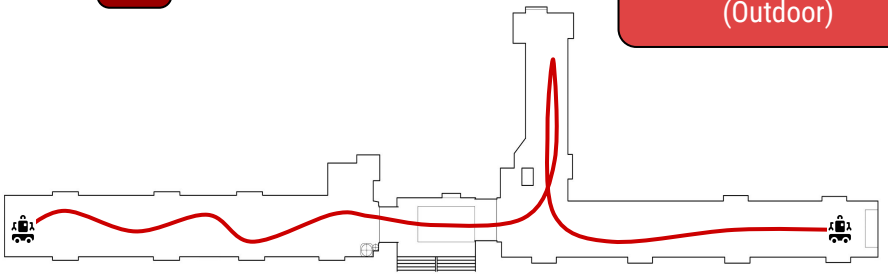
girafe ai

01

(SIMPLIFIED) CONTROL SCHEME OF MODERN MOBILE ROBOT

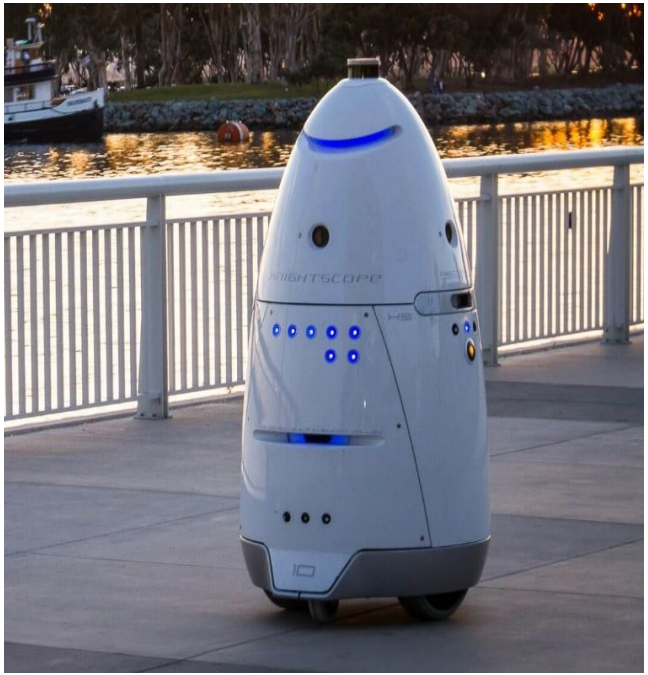
Localization (in robotics) stands for the process of tracking of robot pose (position and orientation) in space (some fixed reference frame)[1,2].





* Ziegler, Julius; Henning Lategahn; Markus Schreiber; Christoph G Keller; Carsten Knoppel; Jochen Hipp; Martin Haueis; and Christoph Stiller. 2014. "Video based localization for bertha." In Intelligent Vehicles Symposium Proceedings, 2014 IEEE, 1231–1238.

WHY LOCALIZATION MATTERS?



Knowing precise pose is a prerequisite for accurate motion planning and execution.

FORMAL LOCALIZATION PROBLEM DEFINITION

Given:

$\mathbf{x}_{1:t-1}$ – all previous states (poses)

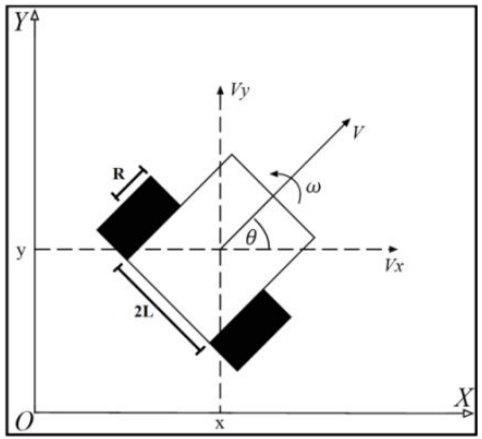
$\mathbf{u}_{1:t}$ – history of the control signals

$\mathbf{z}_{1:t}$ – sensors measurements

map – map

Find:

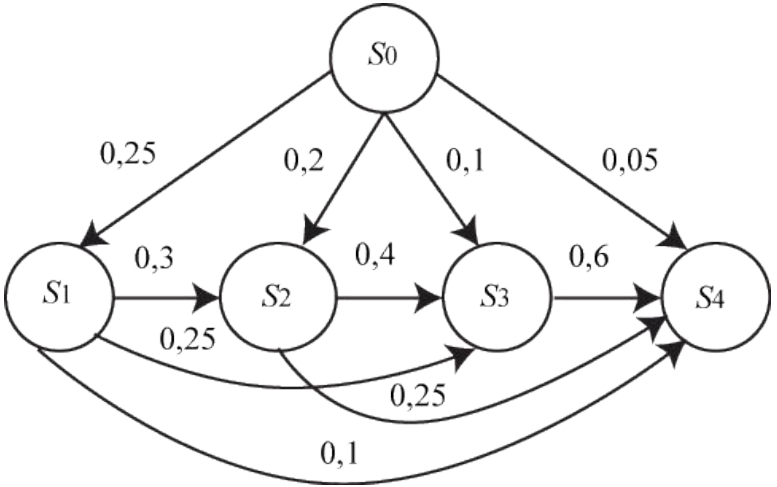
$\mathbf{x}_t$ – current robot pose



MARKOV PROCESSES

Markov process — model of stochastic process which:

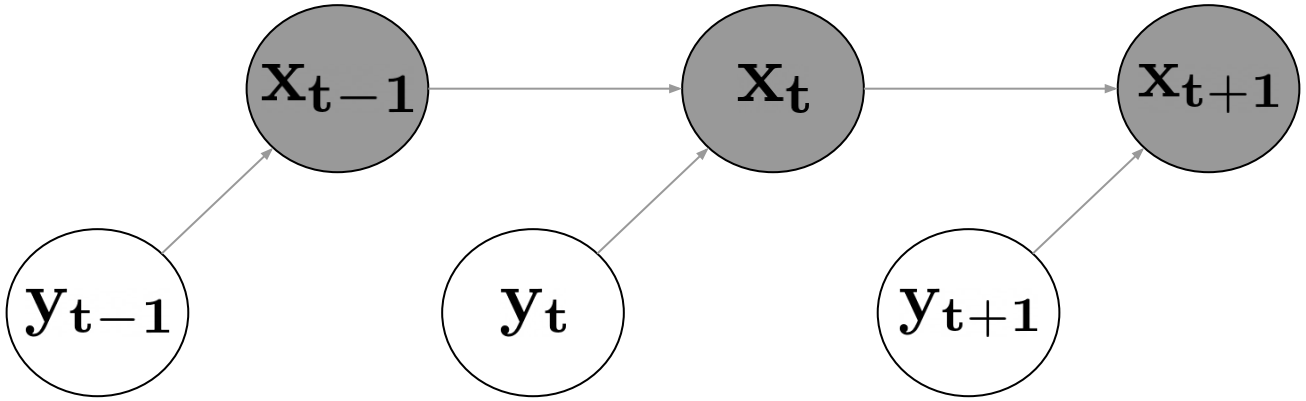
- after any given time moment *t* does not depend on the evolution preceding *t*, given that the state of the process at this moment is fixed
- «future» of the process does not depend on «past» given the known «present»
- satisfies the **Markov property**



Markov property (for the discrete case):

$$p(X_n = \mathbf{x_n} | X_{n-1} = \mathbf{x_{n-1}}, X_{n-2} = \mathbf{x_{n-2}}, ..., X_0 = \mathbf{x_0}) = p(X_n = \mathbf{x_n} | X_{n-1} = \mathbf{x_{n-1}})$$

HIDDEN MARKOV PROCESSES

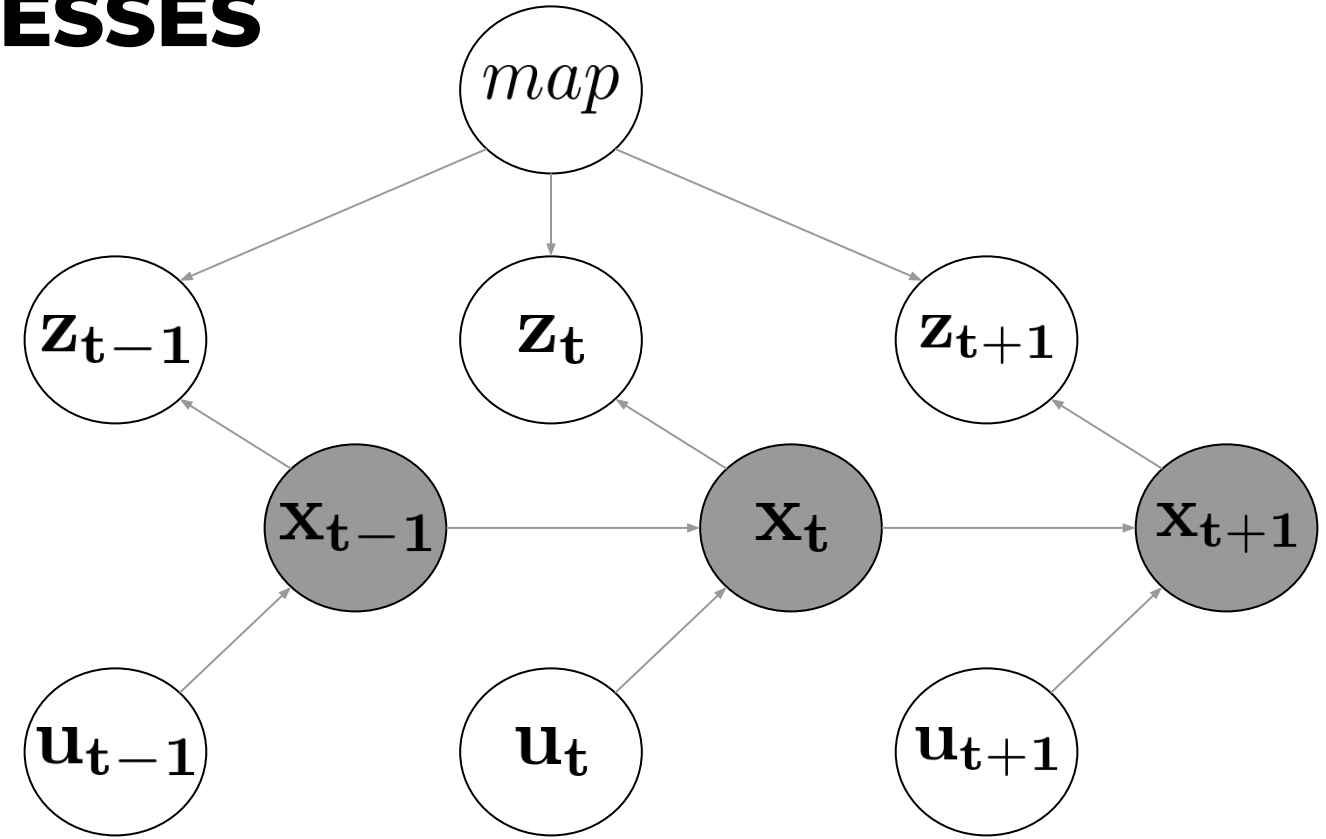


– observable process

– hidden process

girafe ai

LOCALIZATION AS HIDDEN MARKOV PROCESSES



Probabilistic form of localization problem

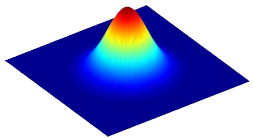
girafe ai

02

PROBABILISTIC FORM OF LOCALIZATION

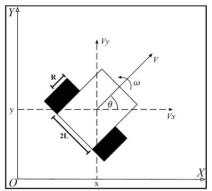
$$p(\mathbf{x_t} | map, \mathbf{z_{1:t}}, \mathbf{u_{1:t}}) = ?$$

Probability Density Function (PDF)



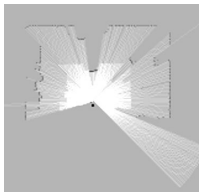
Robot pose

$$\mathbf{x_t} = (x_t, y_t, \theta_t)$$



Map

Measurement



Control signals vector

$$\mathbf{u_t} = (\Delta x_t, \Delta y_t, \Delta \theta_t)$$

$\Delta x_t, \Delta y_t, \Delta \theta_t$ – pose change for the current time increment

Thrun, S., Burgard, W. and Fox, D., 2005. *Probabilistic robotics*. MIT press.

PROBABILISTIC FORM OF LOCALIZATION

$$\hat{\mathbf{x}}_t = \arg\max_{\mathbf{x}} p(\mathbf{x}_t | map, \mathbf{z_{1:t}}, \mathbf{u_{1:t}})$$

Thrun, S., Burgard, W. and Fox, D., 2005. *Probabilistic robotics*. MIT press.

PROBABILISTIC FORM OF LOCALIZATION

Localization problem can be formulated as follows: given the vector of all successive sensors measurements $\mathbf{z}_{1:t} = \mathbf{z}_0 \ldots \mathbf{z}_t$, and control signals $\boldsymbol{u}_{1:t} = \boldsymbol{u}_0 \ldots \boldsymbol{u}_t$, it is needed to recover a posterior probability density function of robot pose $\boldsymbol{x}_t$ at any given moment of time t.

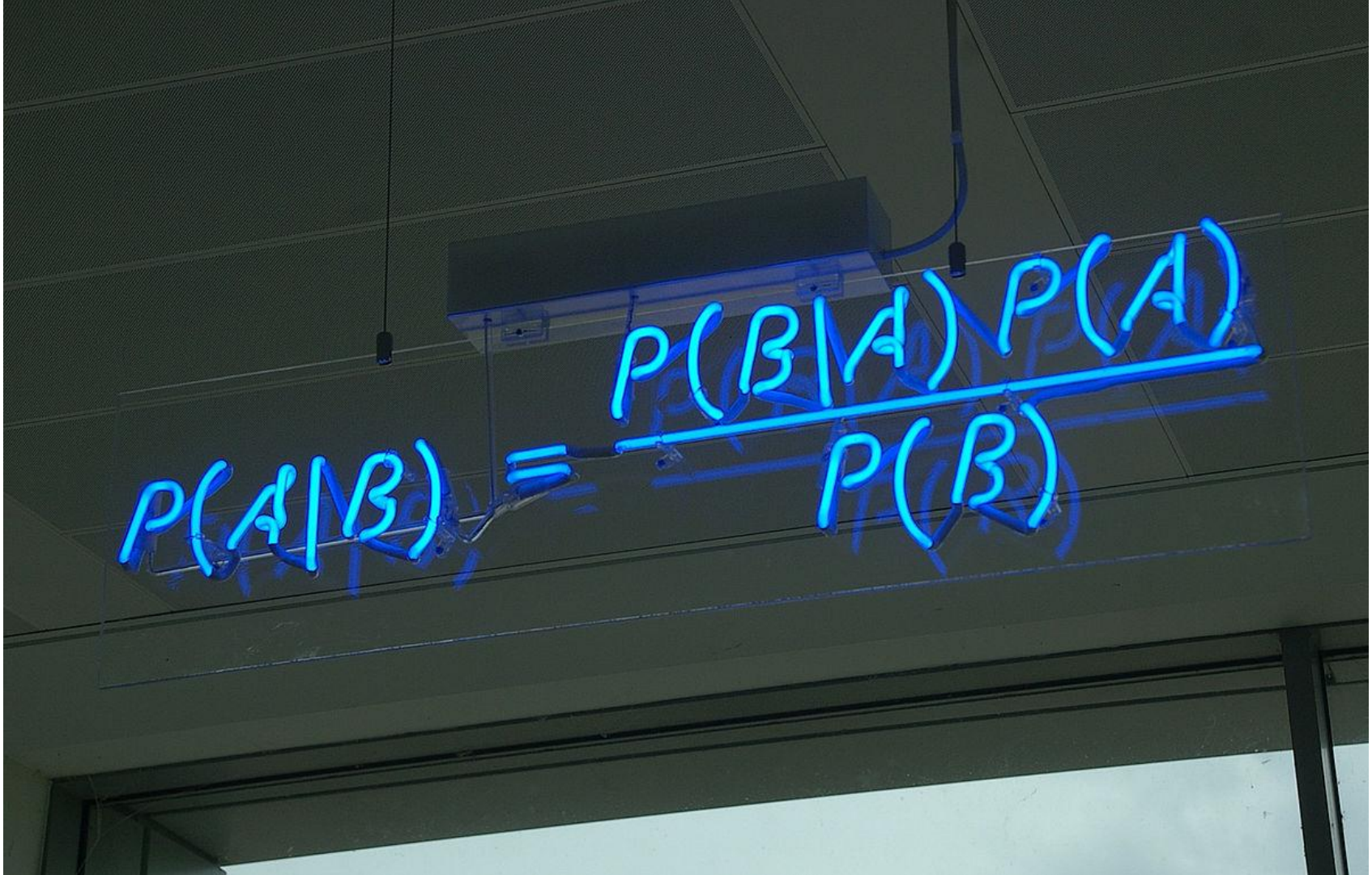
(Sebastian Thrun 2002)

Thrun, S., Burgard, W. and Fox, D., 2005. *Probabilistic robotics*. MIT press.

Recursive Bayesian estimation

girafe ai

03


$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$

BAYES' RULE (THEOREM)

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}$$

Posterior probability
How probable, that our hypothesis a is true given the **evidence** b

Likelihood
How probable is outcome b, given that out hypothesis a is true

Prior probability
How probable our hypothesis a was before obtaining the evidence b

Marginal probability
Probability of b, which does not depend on a

LAW OF TOTAL PROBABILITY

$$p(b) = \sum_{i=1}^{N} p(a_i)p(b|a_i)$$

BAYES' RULE. EXAMPLE

One of the three shooters is called into the firing line and fires two shots. The probability of hitting the target with one shot for the first shooter is **0.3**, for the second - **0.5**, for the third - **0.8**. The target is not hit. Find the probability that the shots were fired by the first shooter.

A_1 — the first shooter is called to the firing line

A_2 — the second shooter is called to the firing line

A_3 — the third shooter is called to the firing line

Since the call to the firing line of any shooter is equally possible, then

$$p(A_1) = p(A_2) = p(A_3) = \frac{1}{3}$$

BAYES' RULE: EXAMPLE

As a result of the experiment, event **B** was observed — after the shots fired, the target was not hit. The conditional probabilities of this event under the hypotheses made are:

$$p(B|A_1) = (1 - 0.3) * (1 - 0.3) = 0.49$$
$$p(B|A_2) = (1 - 0.5) * (1 - 0.5) = 0.25$$
$$p(B|A_3) = (1 - 0.8) * (1 - 0.8) = 0.04$$

Using Bayes' rule, we find the probability of hypothesis A_1 after the experiment:

$$p(A_1|B) = \frac{0.49 * 1/3}{0.49 * 1/3 + 0.25 * 1/3 + 0.4 * 1/3} = 0.628$$

RECURSIVE BAYESIAN ESTIMATION

$$\text{bel}(\mathbf{x}_t) = p(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

RECURSIVE BAYESIAN ESTIMATION

$$\mathrm{bel}(\mathbf{x}_t) = p(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) =$$

Bayes' rule

$$= \eta p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_t | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$$

RECURSIVE BAYESIAN ESTIMATION

$$\mathrm{bel}(\mathbf{x}_t) = p(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) =$$

$$= \eta p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_t | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) =$$

Markov property

$$= \eta p(\mathbf{z}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$$

RECURSIVE BAYESIAN ESTIMATION

$$\begin{aligned}
\mathrm{bel}(\mathbf{x}_t) &= p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_{t-1}
\end{aligned}$$

Law of Total Probability

RECURSIVE BAYESIAN ESTIMATION

$$\begin{aligned}
\mathrm{bel}(\mathbf{x}_t) &= p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_{t-1} = \\
&\text{Markov property} \quad = \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_{t-1}
\end{aligned}$$

RECURSIVE BAYESIAN ESTIMATION

$$\begin{aligned}
\mathrm{bel}(\mathbf{x}_t) &= p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_{t-1} = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_{t-1} = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) d\mathbf{x}_{t-1}
\end{aligned}$$

Markov property

RECURSIVE BAYESIAN ESTIMATION

$$\begin{aligned}
\mathrm{bel}(\mathbf{x}_t) &= p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_{t-1} = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_{t-1} = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) d\mathbf{x}_{t-1} = \\
&= \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \mathrm{bel}(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1}
\end{aligned}$$

Recursive member

RECURSIVE BAYESIAN POSE ESTIMATION

$$p(\mathbf{x_t}|map, \mathbf{z_t}, \mathbf{u_t}) = C \cdot p(\mathbf{z_t}|\mathbf{x_t}, map) \int_S p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}}) p(\mathbf{x_{t-1}}|map, \mathbf{z_{t-1}}, \mathbf{u_{t-1}}) d\mathbf{x_{t-1}}$$

C — normalization coefficient

S — the probabilistic space of robot poses

$p(\mathbf{z_t}|\mathbf{x_t}, map)$ — observation (measurement) model

$p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}})$ — motion model

$p(\mathbf{x_{t-1}}|map, \mathbf{z_{t-1}}, \mathbf{u_{t-1}})$ — previous system state (robot pose)

RECURSIVE BAYESIAN POSE ESTIMATION

$$p(\mathbf{x_t}|map, \mathbf{z_t}, \mathbf{u_t}) = C \cdot p(\mathbf{z_t}|\mathbf{x_t}, map) \int_S p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}}) p(\mathbf{x_{t-1}}|map, \mathbf{z_{t-1}}, \mathbf{u_{t-1}}) d\mathbf{x_{t-1}}$$

C — normalization coefficient

S — the probabilistic space of robot poses

$p(\mathbf{z_t}|\mathbf{x_t}, map)$ — observation (measurement) model

$p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}})$ — motion model

$p(\mathbf{x_{t-1}}|map, \mathbf{z_{t-1}}, \mathbf{u_{t-1}})$ — previous system state (robot pose)

RECURSIVE BAYESIAN POSE ESTIMATION

$$p(\mathbf{x_t}|map, \mathbf{z_t}, \mathbf{u_t}) = C \cdot p(\mathbf{z_t}|\mathbf{x_t}, map) \int_S p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}}) p(\mathbf{x_{t-1}}|map, \mathbf{z_{t-1}}, \mathbf{u_{t-1}}) d\mathbf{x_{t-1}}$$

Correction

Prediction

C — normalization coefficient

S — the probabilistic space of robot poses

$p(\mathbf{z_t}|\mathbf{x_t}, map)$ — observation (measurement) model

$p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}})$ — motion model

$p(\mathbf{x_{t-1}}|map, \mathbf{z_{t-1}}, \mathbf{u_{t-1}})$ — previous system state (robot pose)

IMPLEMENTATION OF RECURSIVE BAYESIAN ESTIMATION

- Recursive Bayesian estimation is a **framework**
- There are many algorithms realizing this framework:
  - Linear and nonlinear motion and observation models
  - Normal and arbitrary (multimodal) error distributions
  - Parametric and nonparametric algorithms
  - ....

- Kalman filter
- Information filter
- Histogram filter
- Particle filter
- ....

IMPLEMENTATION OF RECURSIVE BAYESIAN ESTIMATION

- Recursive Bayesian estimation is a **framework**
- There are many algorithms realizing this framework:
  - Linear and nonlinear motion and observation models
  - Normal and arbitrary (multimodal) error distributions
  - Parametric and nonparametric algorithms
  - ....

- **Kalman filter**
- Information filter
- Histogram filter
- **Particle filter**
- ....

Kalman filter

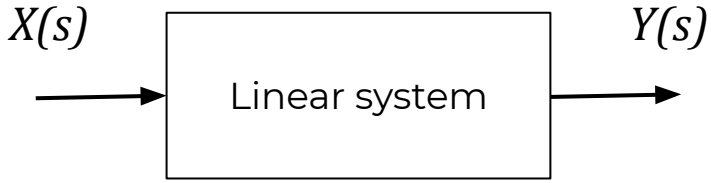
girafe
ai

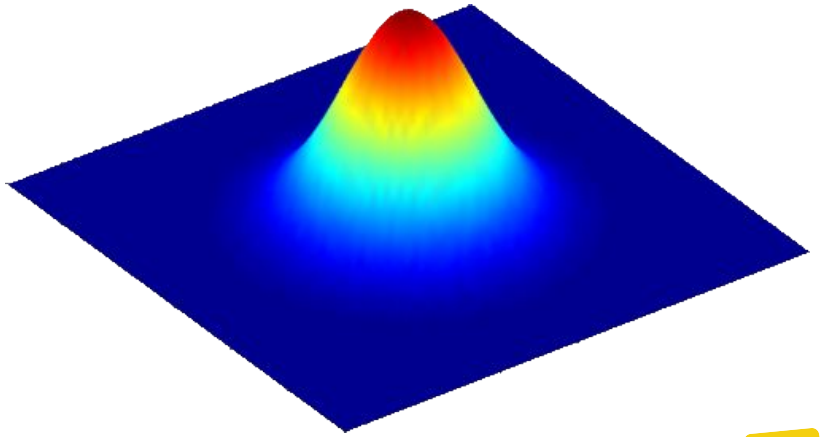
04

KALMAN FILTER

Provably optimal:

- Linear motion and observation models
- Normally distributed motion and measurement errors





girafe ai

LINEAR SYSTEM

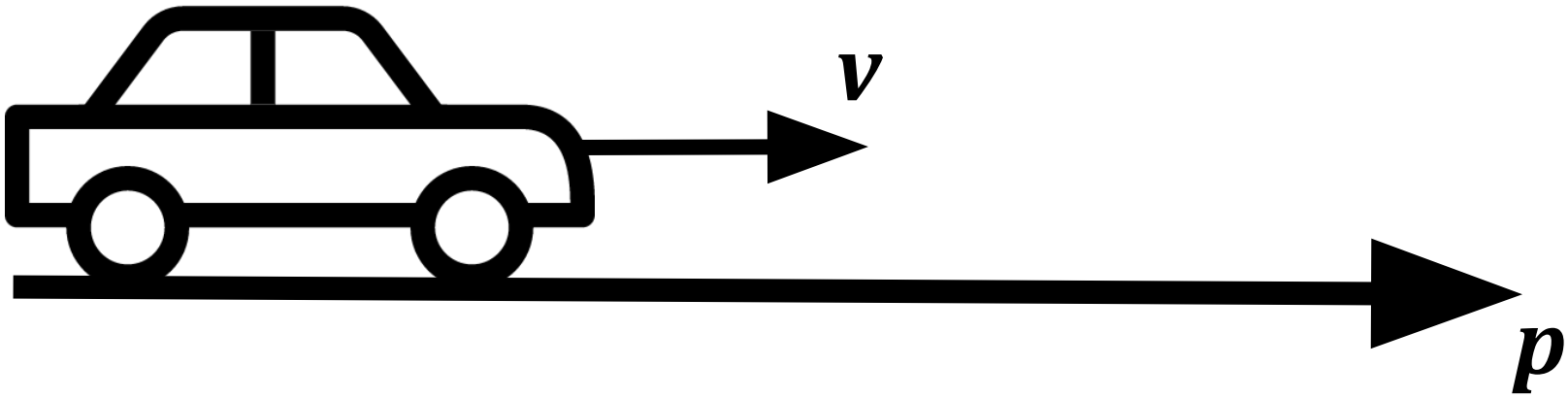
System is called linear if:

- the response of the system to the sum of input signals is equal to the sum of responses to each signal

Necessary linearity conditions:

- **Homogeneity** — when the amplitude of the input signal changes by a factor of *k*, the amplitude of the output signal also changes by a factor of *k*
- **Additivity** — when summing input signals, the resulting output signal will be equal to the sum of reactions from the original signals
- **Time invariance** — offsetting the input signal in time causes a similar offset in the output signal
- **Static linearity** — the basic laws in the system are described by linear equations
- **Harmonic fidelity** — if a sinusoidal signal is applied to the input of the system, then the output will be a signal of the same frequency

KALMAN FILTER. EXAMPLE



KALMAN FILTER. SYSTEM STATE SPACE

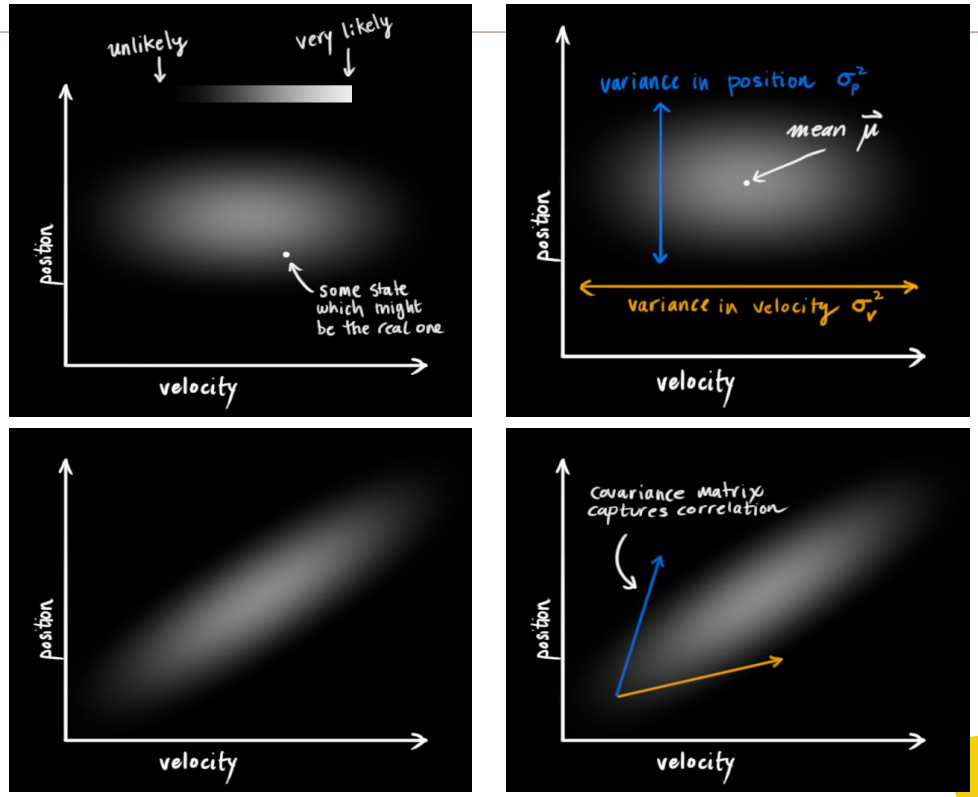
Our knowledge of the current system state is described by 2 elements:

1. Estimate of the current state vector:

$$\hat{\mathbf{x}}_\mathbf{t} = \begin{bmatrix} p \\ v \end{bmatrix}$$

2. Estimate of the covariance matrix:

$$\hat{\mathbf{\Sigma}}_\mathbf{t} = \begin{bmatrix} \sigma_{pp} & \sigma_{pv} \\ \sigma_{vp} & \sigma_{vv} \end{bmatrix} = \begin{bmatrix} \sigma_p^2 & \sigma_{pv} \\ \sigma_{vp} & \sigma_v^2 \end{bmatrix}$$

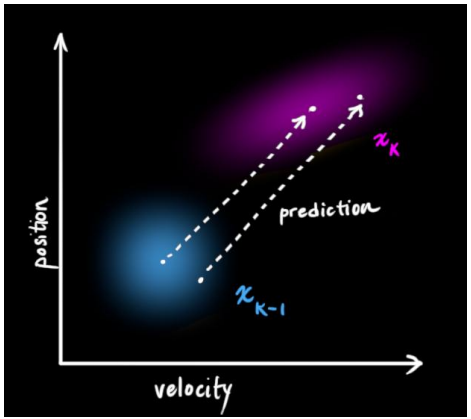


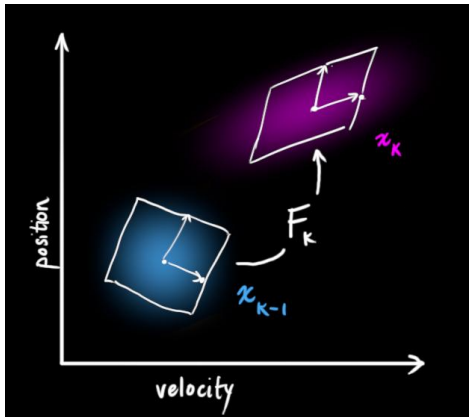
Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

KALMAN FILTER. PREDICTION

Prediction of the system state at the next time moment is made based on the motion model:

$$p_t = p_{t-1} + \Delta t v_{t-1}$$
$$v_t = v_{t-1}$$





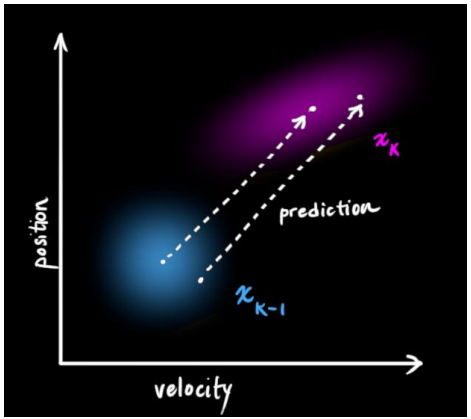
In matrix form:

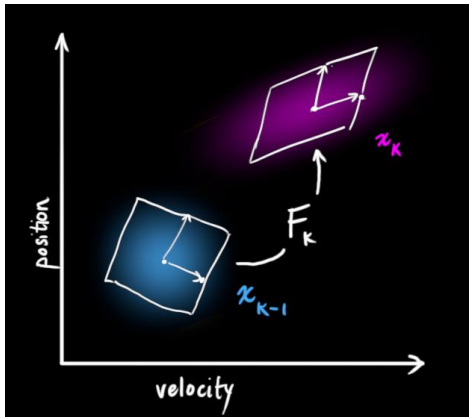
$$\hat{\mathbf{x}}_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{t-1} = \mathbf{F}_t \hat{\mathbf{x}}_{t-1}$$

$\mathbf{F}_t$ — the matrix of the process / system evolution (in our case, the motion model) from *t-1* to *t*

KALMAN FILTER. PREDICTION

$$cov(x) = \Sigma$$
$$cov(\mathbf{A}x) = \mathbf{A}\Sigma\mathbf{A}^T$$



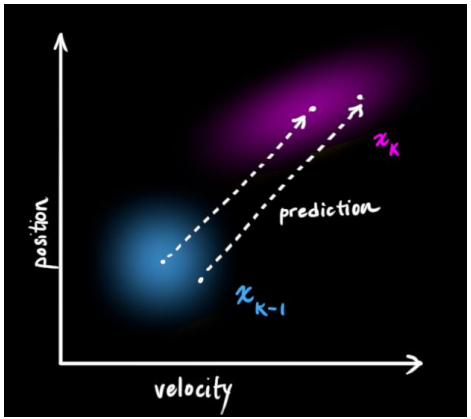


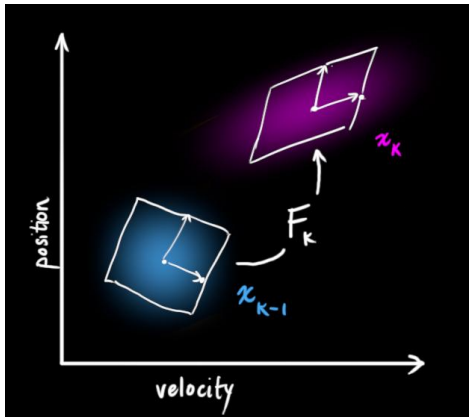
$$\hat{\mathbf{x}}_t = \mathbf{F}_t\hat{\mathbf{x}}_{t-1}$$
$$\hat{\Sigma}_t = \mathbf{F}_t\Sigma_{t-1}\mathbf{F}_t^T$$

KALMAN FILTER. CONTROL SIGNALS

$$p_t = p_{t-1} + \Delta t v_{t-1} + \frac{1}{2} a \Delta t^2$$

$$v_t = v_{t-1} + a\Delta t$$





In matrix form:

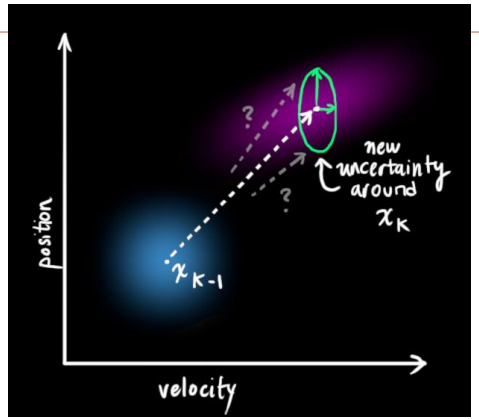
$$\hat{\mathbf{x}}_t = \mathbf{F}_t \hat{\mathbf{x}}_{t-1} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a = \mathbf{F}_t \hat{\mathbf{x}}_{t-1} + \mathbf{B}_t \vec{\mathbf{u}}_t$$

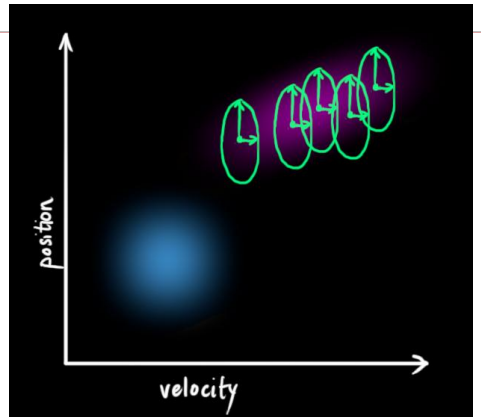
$\mathbf{B}_t$ — the matrix describing the change in control signals from *t-1* to *t*

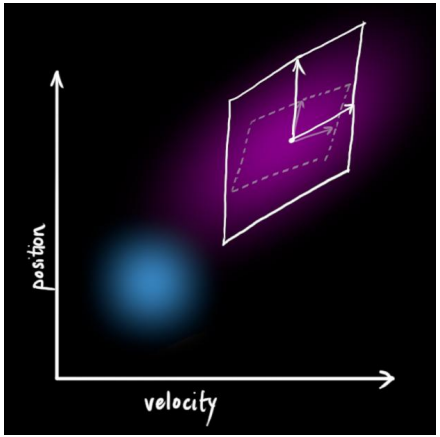
KALMAN FILTER. MOTION UNCERTAINTY

$$\hat{\mathbf{x}}_t = \mathbf{F}_t\hat{\mathbf{x}}_{t-1} + \mathbf{B}_k\vec{\mathbf{u}}_t$$

$$\hat{\mathbf{\Sigma}}_t = \mathbf{F}_t\mathbf{\Sigma}_{t-1}\mathbf{F}_t^T + \mathbf{Q}_t$$





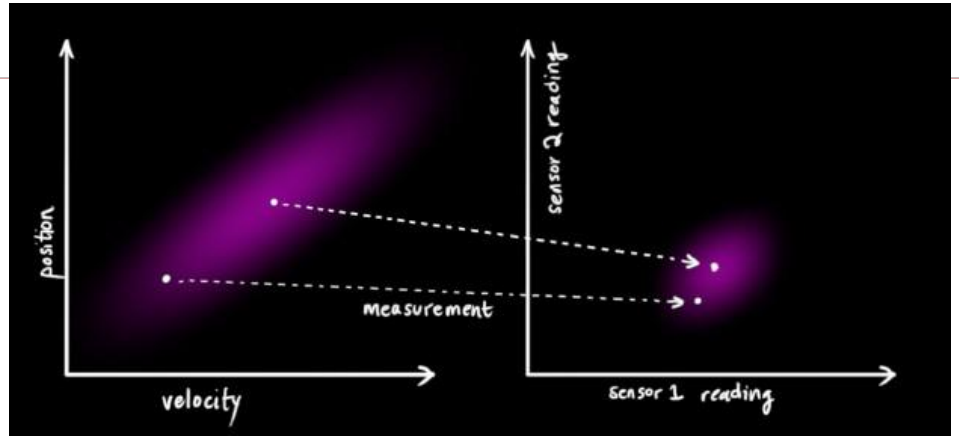


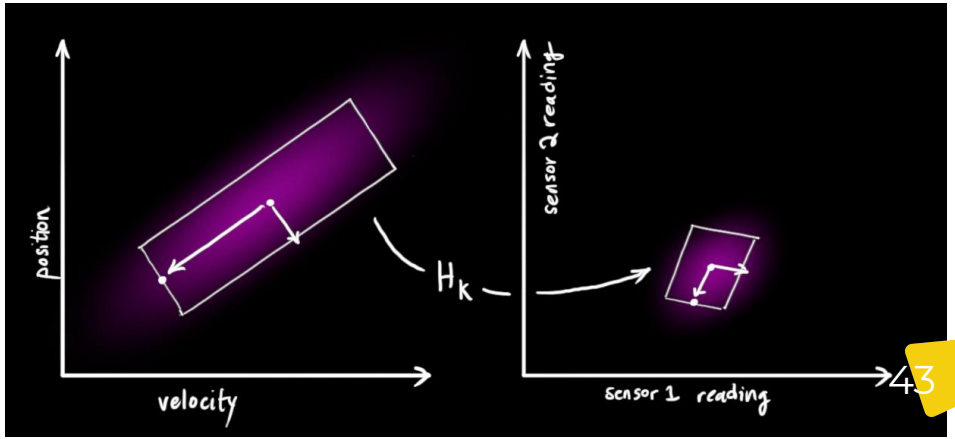
$\mathbf{Q}_t$ — the covariance matrix, which describes the random nature of the system evolution

KALMAN FILTER. CORRECTION

$$\vec{\mu}_{\text{expected}} = \mathbf{H}_t \hat{\mathbf{x}}_t$$

$$\Sigma_{\text{expected}} = \mathbf{H}_t \Sigma_t \mathbf{H}_t^T$$



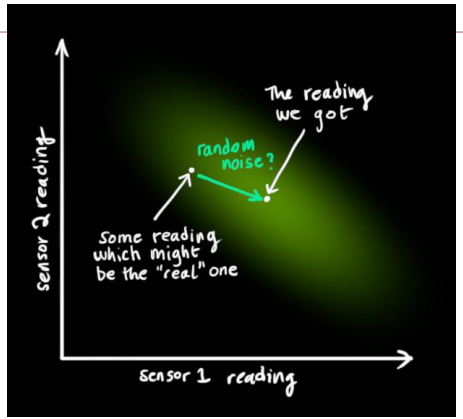


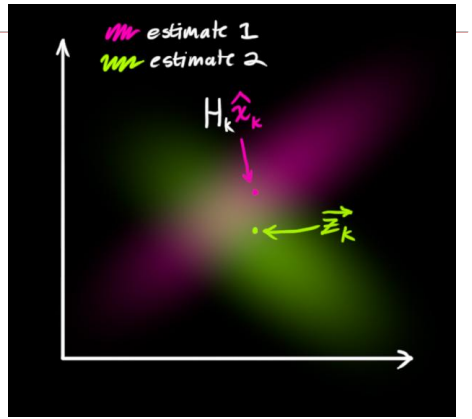
$\mathbf{H}_t$ — the matrix describing a measurement model: how to get a measurement $\mathbf{z}_t$ from a state $\hat{\mathbf{x}}_t$

KALMAN FILTER. MEASUREMENT UNCERTAINTY

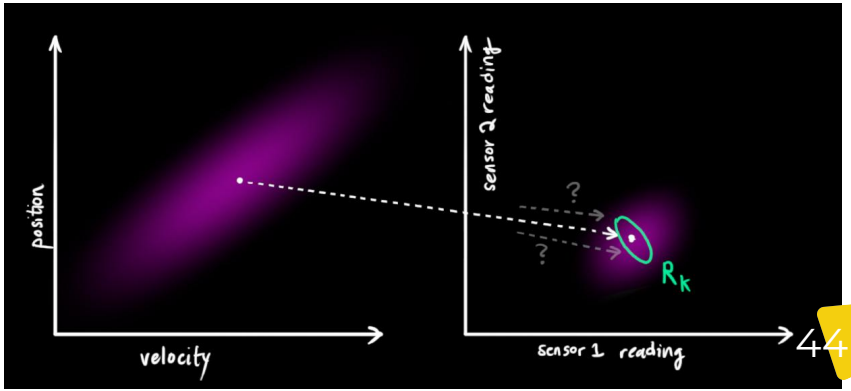
$$\vec{\mu}_{\text{measured}} = \vec{z}_t$$

$$\Sigma_{\text{measured}} = R_t$$





R_t — the covariance matrix that describes the random nature of the measurements

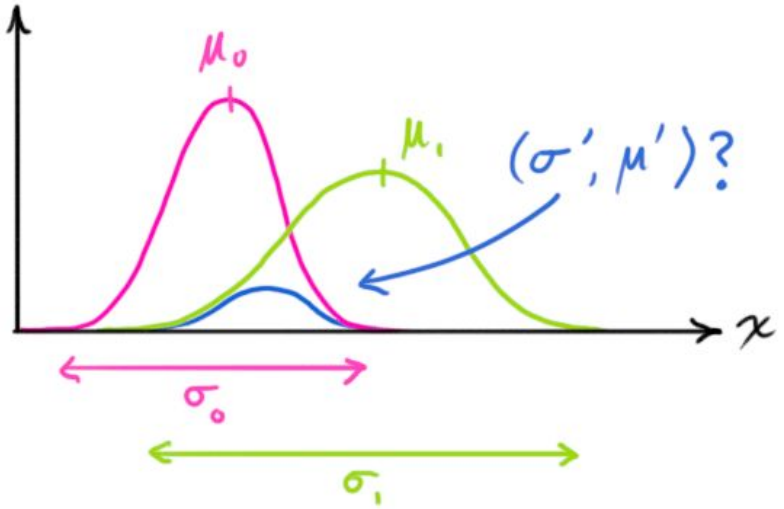


NORMAL DISTRIBUTION

$$\mathcal{N}(x, \mu_0, \sigma_0) \cdot \mathcal{N}(x, \mu_1, \sigma_1) \stackrel{?}{=} \mathcal{N}(x, \mu, \sigma)$$

$$\mu = \mu_0 + \frac{\sigma_0^2(\mu_1 \mu_0)}{\sigma_0^2 + \sigma_1^2}$$

$$\sigma^2 = \sigma_0^2 - \frac{\sigma_0^4}{\sigma_0^2 + \sigma_1^2}$$



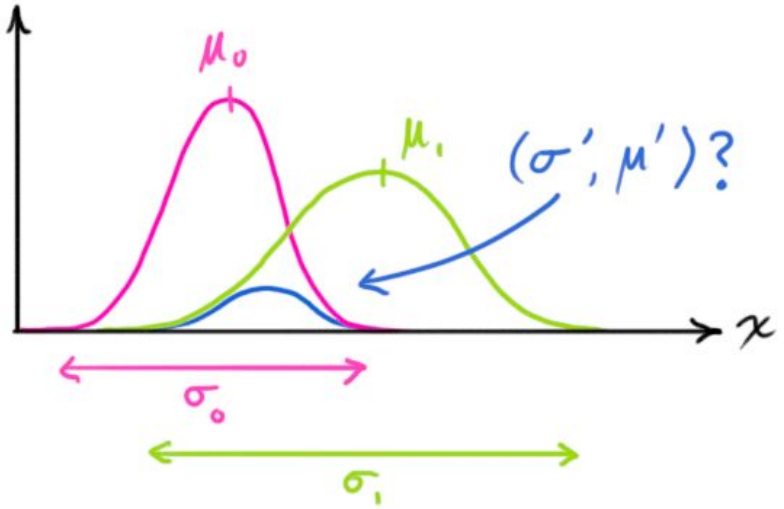
NORMAL DISTRIBUTION

$$\mathcal{N}(x, \mu_0, \sigma_0) \cdot \mathcal{N}(x, \mu_1, \sigma_1) \stackrel{?}{=} \mathcal{N}(x, \mu, \sigma)$$

$$\mathbf{K} = \Sigma_0(\Sigma_0 + \Sigma_1)^{-1}$$

$$\vec{\mu} = \vec{\mu_0} + \mathbf{K}(\vec{\mu_1} - \vec{\mu_0})$$

$$\Sigma = \Sigma_0 - \mathbf{K}\Sigma_0$$



KALMAN FILTER. PUTTING THIS ALL TOGETHER

$$\mathbf{K} = \Sigma_0(\Sigma_0 + \Sigma_1)^{-1}$$
$$\vec{\mu} = \vec{\mu_0} + \mathbf{K}(\vec{\mu_1} - \vec{\mu_0})$$
$$\Sigma = \Sigma_0 - \mathbf{K}\Sigma_0$$

$$(\mu_0, \Sigma_0) = (\mathbf{H}_t\hat{\mathbf{x}}_t, \mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T)$$
$$(\mu_1, \Sigma_1) = (\vec{\mathbf{z}}_t, \mathbf{R}_t)$$

$$\mathbf{H}_t\hat{\mathbf{x}'}_t = \mathbf{H}_t\hat{\mathbf{x}}_t + \mathbf{K}(\vec{\mathbf{z}}_t - \mathbf{H}_t\hat{\mathbf{x}}_t)$$

$$\mathbf{H}_t\mathbf{\Sigma'}_t\mathbf{H}_t^T = \mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T - \mathbf{K}\mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T$$

KALMAN FILTER. PUTTING THIS ALL TOGETHER

$$\mathbf{K} = \Sigma_0(\Sigma_0 + \Sigma_1)^{-1}$$
$$\vec{\mu} = \vec{\mu_0} + \mathbf{K}(\vec{\mu_1} - \vec{\mu_0})$$
$$\Sigma = \Sigma_0 - \mathbf{K}\Sigma_0$$

$$(\mu_0, \Sigma_0) = (\mathbf{H}_t\hat{\mathbf{x}}_t, \mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T)$$
$$(\mu_1, \Sigma_1) = (\vec{\mathbf{z}}_t, \mathbf{R}_t)$$

$$\mathbf{K} = \mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T(\mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T + \mathbf{R}_t)^{-1}$$

$$\mathbf{H}_t\mathbf{x'}_t = \mathbf{H}_t\hat{\mathbf{x}}_t + \mathbf{K}(\vec{\mathbf{z}}_t - \mathbf{H}_t\hat{\mathbf{x}}_t)$$

$$\mathbf{H}_t\mathbf{\Sigma'}_t\mathbf{H}_t^T = \mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T - \mathbf{K}\mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T$$

KALMAN FILTER. PUTTING THIS ALL TOGETHER

$$\mathbf{K} = \Sigma_0(\Sigma_0 + \Sigma_1)^{-1}$$
$$\vec{\mu} = \vec{\mu_0} + \mathbf{K}(\vec{\mu_1} - \vec{\mu_0})$$
$$\Sigma = \Sigma_0 - \mathbf{K}\Sigma_0$$

$$(\mu_0, \Sigma_0) = (\mathbf{H}_t\hat{\mathbf{x}}_t, \mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T)$$
$$(\mu_1, \Sigma_1) = (\vec{\mathbf{z}}_t, \mathbf{R}_t)$$

$$\mathbf{K}' = \hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T(\mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T + \mathbf{R}_t)^{-1}$$

$$\mathbf{x}'_t = \mathbf{H}_t\hat{\mathbf{x}}_t + \mathbf{K}'(\vec{\mathbf{z}}_t - \mathbf{H}_t\hat{\mathbf{x}}_t)$$

$$\mathbf{\Sigma}'_t = \hat{\mathbf{\Sigma}}_t - \mathbf{K}'\mathbf{H}_t\hat{\mathbf{\Sigma}}_t$$

KALMAN FILTER

Prediction:

$$\hat{\mathbf{x}}_t = \mathbf{F}_t\hat{\mathbf{x}}_{t-1} + \mathbf{B}_k\vec{\mathbf{u}}_t$$

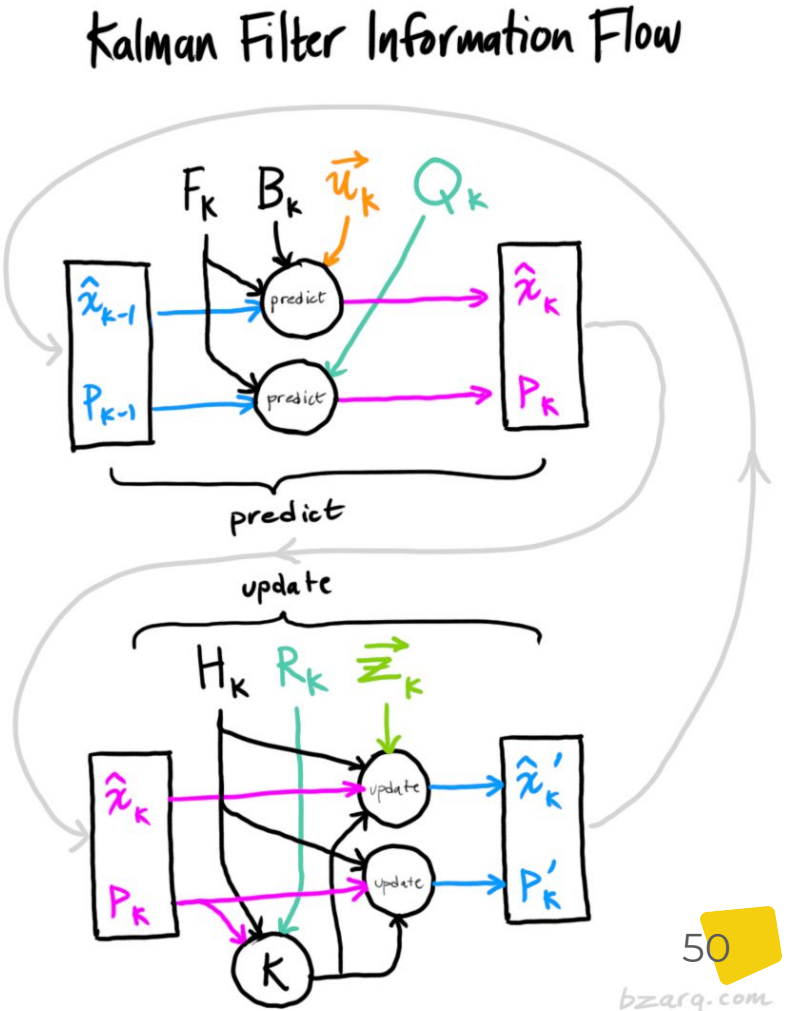
$$\hat{\mathbf{\Sigma}}_t = \mathbf{F}_t\mathbf{\Sigma}_{t-1}\mathbf{F}_t^T + \mathbf{Q}_t$$

Correction:

$$\mathbf{K}' = \hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T(\mathbf{H}_t\hat{\mathbf{\Sigma}}_t\mathbf{H}_t^T + \mathbf{R}_t)^{-1}$$

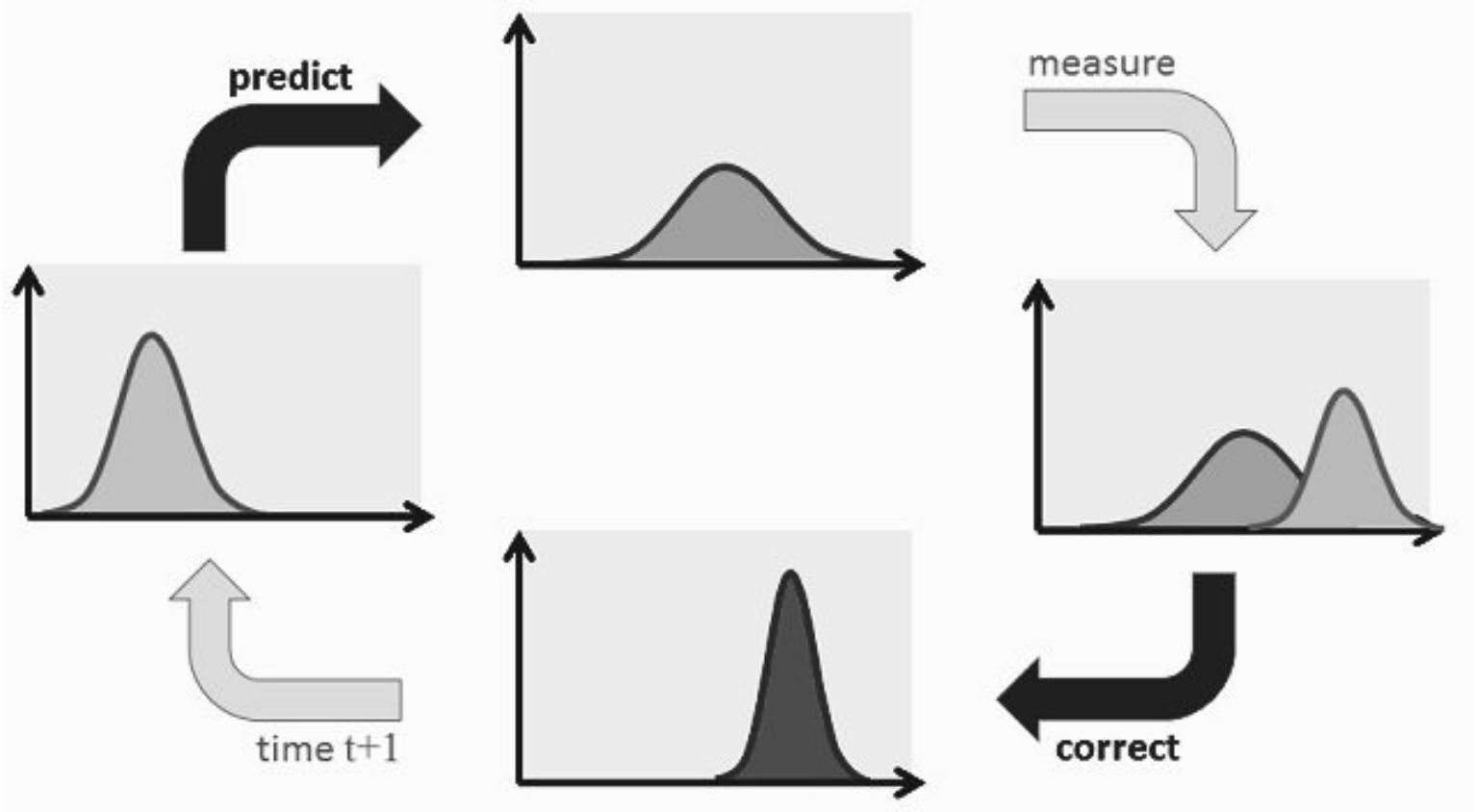
$$\mathbf{x}'_t = \mathbf{H}_t\hat{\mathbf{x}}_t + \mathbf{K}'(\vec{\mathbf{z}}_t - \mathbf{H}_t\hat{\mathbf{x}}_t)$$

$$\mathbf{\Sigma}'_t = \hat{\mathbf{\Sigma}}_t - \mathbf{K}'\mathbf{H}_t\hat{\mathbf{\Sigma}}_t$$

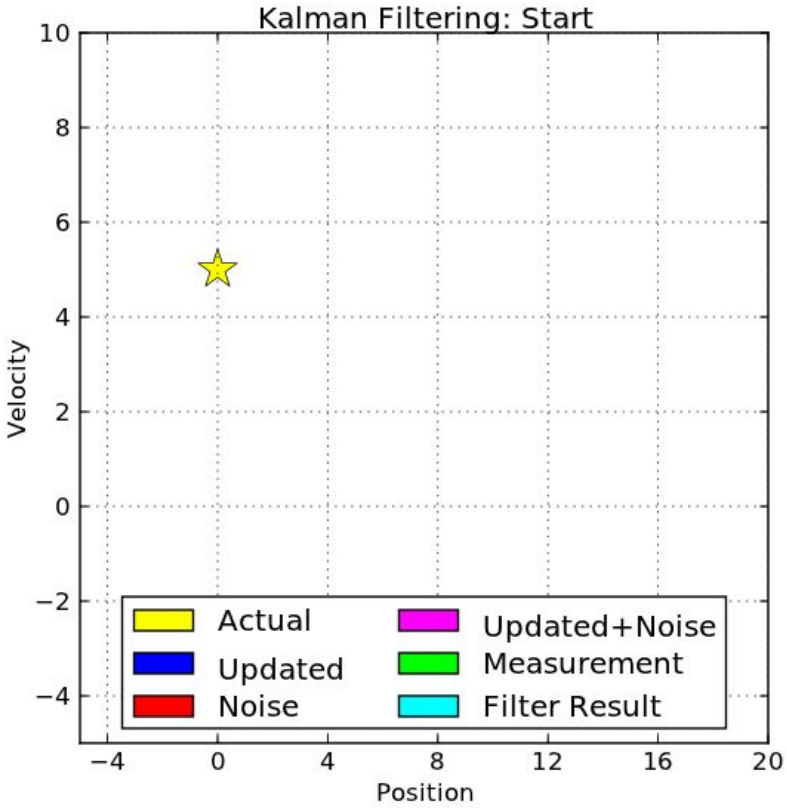


bzarg.com

KALMAN FILTER



KALMAN FILTER



HOW TO FIGHT NONLINEARITY?

Imagine that now we also want to estimate the orientation of the robot:

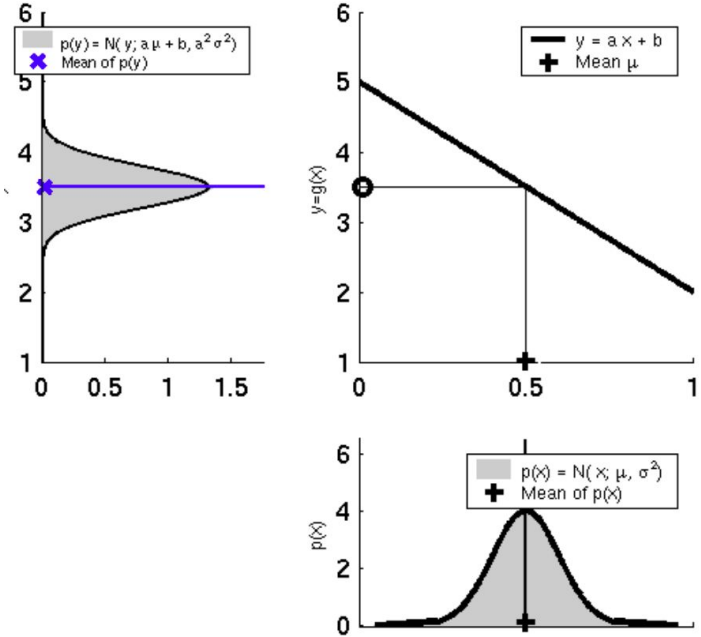
- ❏ This will lead to the appearance of trigonometric functions in the system model
- ❏ The output of a nonlinear system is **no longer a normal distribution**
- ❏ Kalman filter **no longer applicable**

Solution:

- ❏ Local linearization of models
- ❏ Sigma points approximation (Unscented transform)

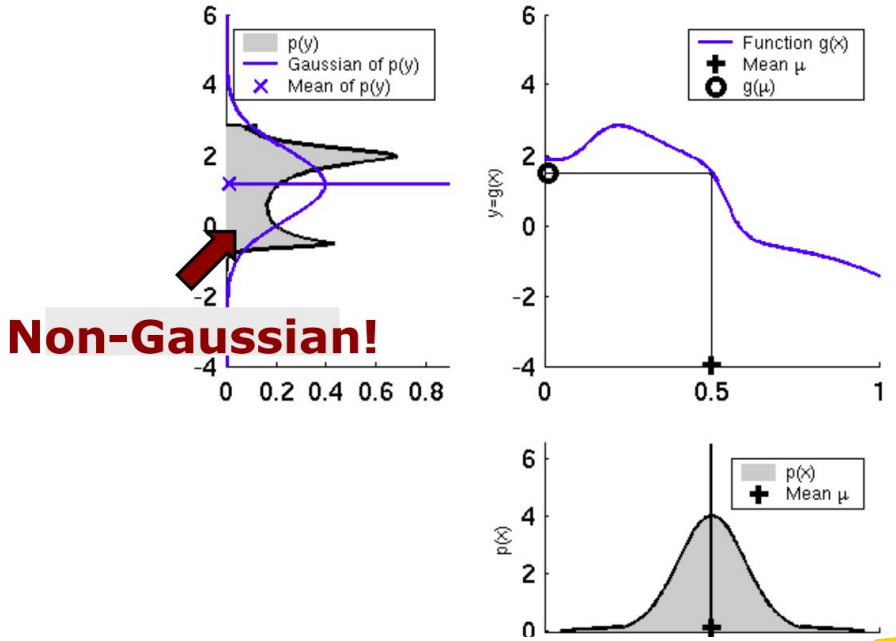
HOW TO FIGHT NONLINEARITY?

Linear system



VS

Nonlinear system



Source: Thrun, Burgard, Fox

HOW TO FIGHT NONLINEARITY?

Linear system **VS** Nonlinear system

$$\hat{\mathbf{x}}_t = \mathbf{F}_t\hat{\mathbf{x}}_{t-1} + \mathbf{B}_t\vec{\mathbf{u}}_t \qquad \hat{\mathbf{x}}_t = g(\hat{\mathbf{x}}_{t-1}, \vec{\mathbf{u}}_t)$$

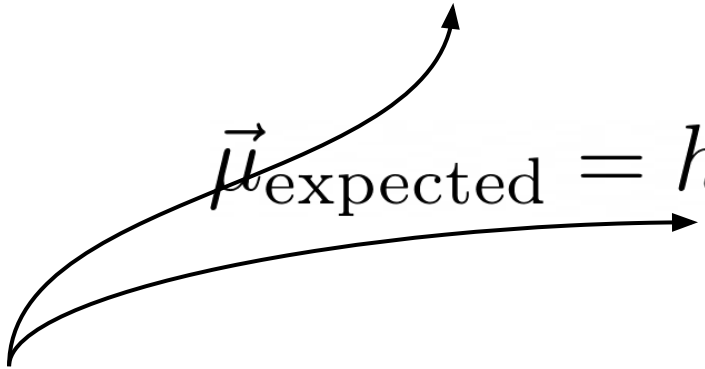
$$\vec{\mu}_{\text{expected}} = \mathbf{H}_t\hat{\mathbf{x}}_t \qquad \vec{\mu}_{\text{expected}} = h(\hat{\mathbf{x}}_t)$$

HOW TO FIGHT NONLINEARITY?

Linear system **VS** Nonlinear system

$$\hat{\mathbf{x}}_t = \mathbf{F}_t\hat{\mathbf{x}}_{t-1} + \mathbf{B}_t\vec{\mathbf{u}}_t \qquad \hat{\mathbf{x}}_t = g(\hat{\mathbf{x}}_{t-1}, \vec{\mathbf{u}}_t)$$

$$\vec{\mu}_{\text{expected}} = \mathbf{H}_t\hat{\mathbf{x}}_t \qquad \vec{\mu}_{\text{expected}} = h(\hat{\mathbf{x}}_t)$$



Nonlinear functions

EXTENDED KALMAN FILTER (EKF)

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

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

EXTENDED KALMAN FILTER (EKF)

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

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

Jacobian matrices

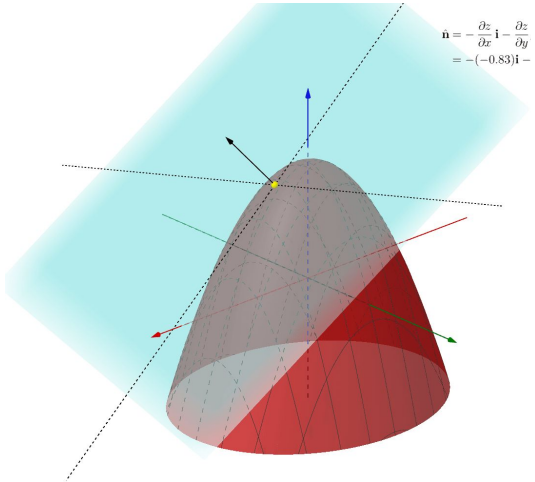
EXTENDED KALMAN FILTER (EKF)

Let a vector function be given:

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

Then the Jacobi matrix is defined as:

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



Specifies the orientation of the plane normal to the vector function at a given point.

Generalizes the concept of the gradient of a scalar function to the multidimensional case.

EXTENDED KALMAN FILTER (EKF)

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

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

Jacobian matrices

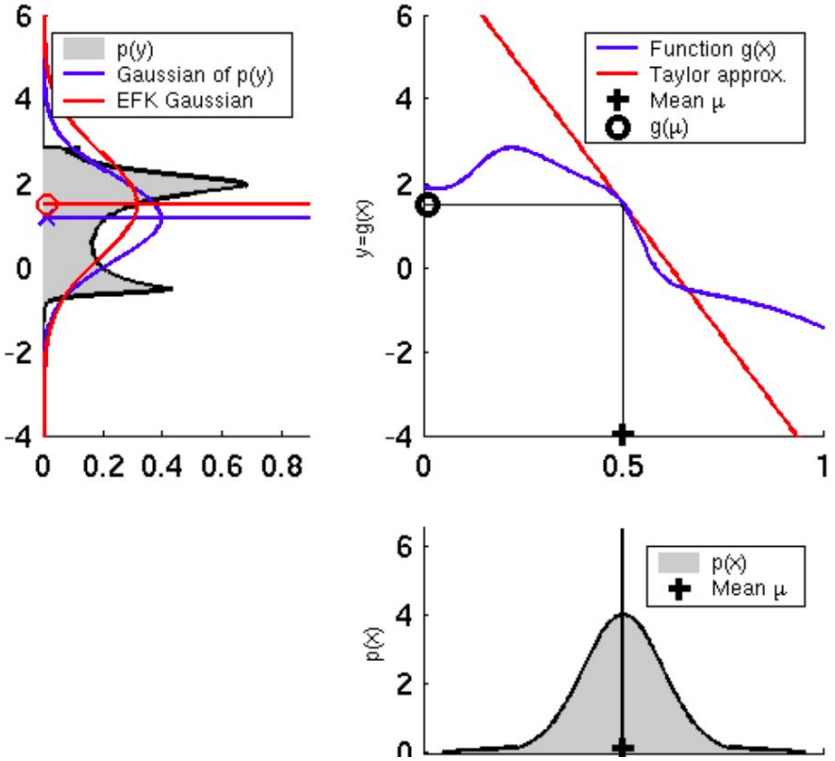
EXTENDED KALMAN FILTER (EKF)

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

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

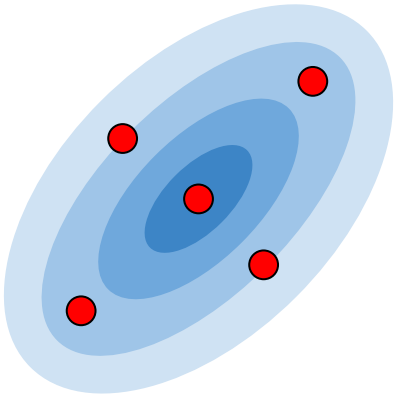
Linear functions!

EXTENDED KALMAN FILTER (EKF)

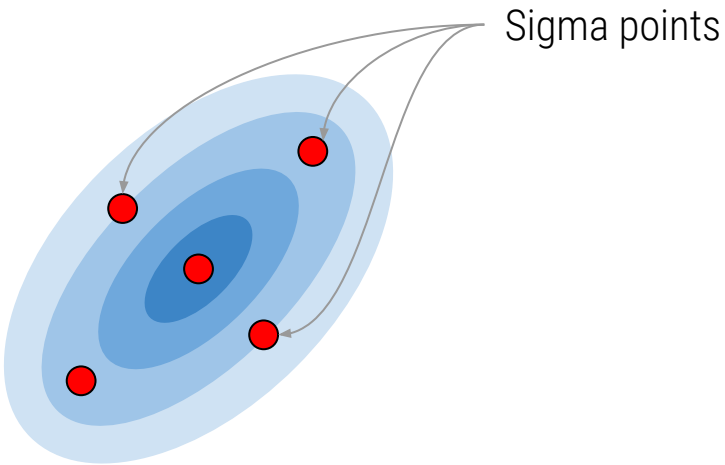


SIGMA-POINT KALMAN FILTER
UNSCENTED KALMAN FILTER (UKF)

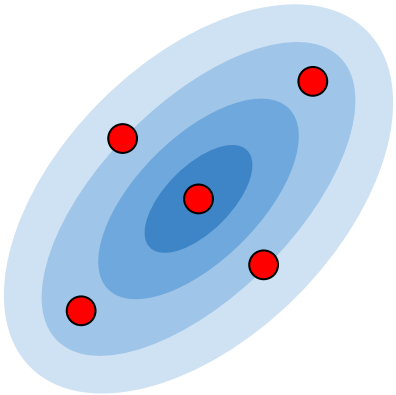
SIGMA-POINT KALMAN FILTER
UNSCENTED KALMAN FILTER (UKF)

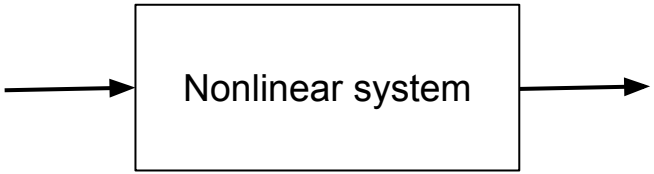


SIGMA-POINT KALMAN FILTER
UNSCENTED KALMAN FILTER (UKF)

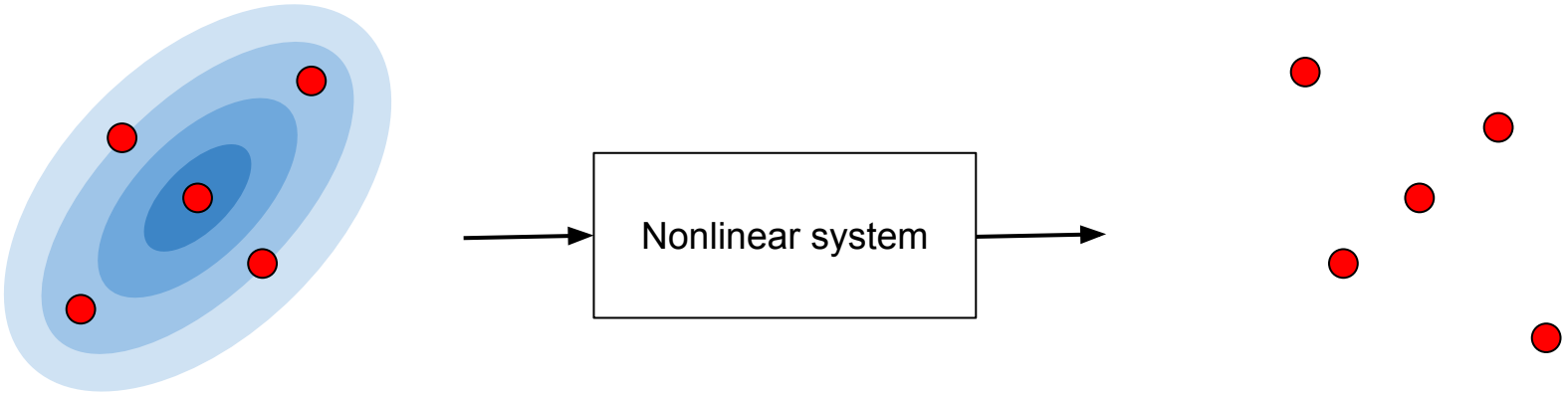


SIGMA-POINT KALMAN FILTER
UNSCENTED KALMAN FILTER (UKF)

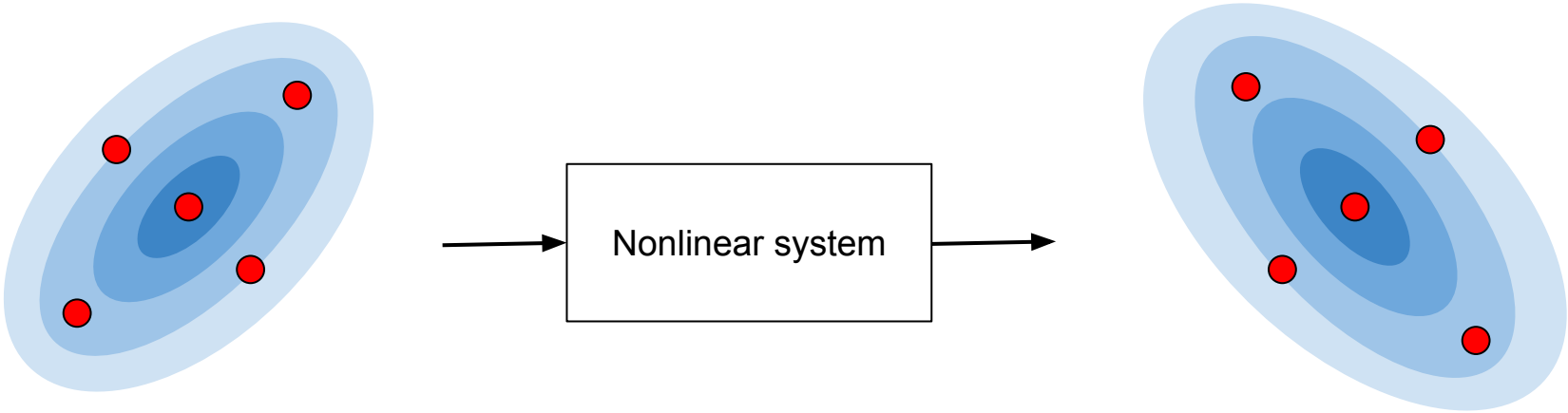




SIGMA-POINT KALMAN FILTER
UNSCENTED KALMAN FILTER (UKF)



SIGMA-POINT KALMAN FILTER
UNSCENTED KALMAN FILTER (UKF)



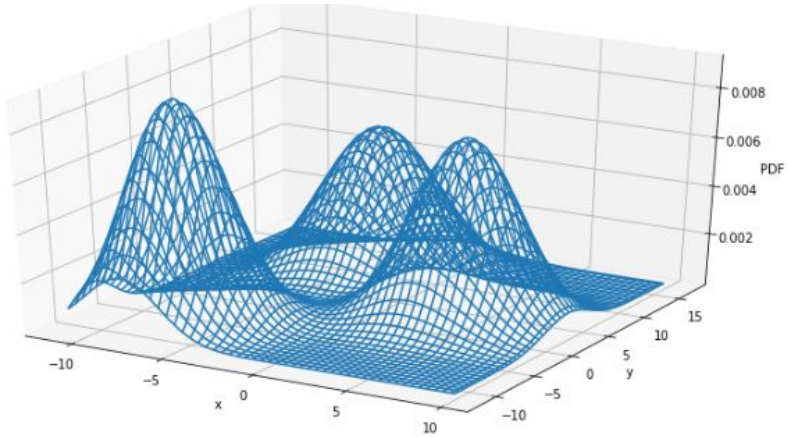
Particle filter

girafe ai

05

PARTICLE FILTER (MONTE CARLO LOCALIZATION)

- Works with any (even **nonlinear**) motion and observation **models**
- Can describe **multimodal** distribution
- Can solve **global localization** problem — localization without an initial approximation



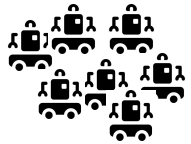
PARTICLE FILTER

- The particle filter allows one to estimate the posterior distribution of the state vector of a system modeled by a hidden Markov process.
- x_t — state of a Markov process at a time t.
- x_t depends on the previous state x_{t-1} in accordance with the **motion model** $p(x_t \mid u_t, x_{t-1})$, where u_t is the vector of control signals applied at the moment $t - 1$.
- x_t is described by the measurement vector z_t generated according to the observation model $p(z_t|x_t)$.
- **Particle filter idea** — approximation of an unknown posterior distribution by a set of hypotheses $\{x^n_t\}$ and the corresponding weights $\{w^n_t\}$, where $n = 1, \ldots, N$ – number of hypotheses / particles.
- In the localization problem, each particle corresponds to the pose of the robot: $x_t = (x_t, y_t, \boldsymbol{\theta}_t)^T$.

Algorithm 1 Generic Monte-Carlo localization algorithm

1: procedure MCL($\mathbf{x_{t-1}}, m, \mathbf{u_t}, \mathbf{z_t}$)
2: $\{\mathbf{x_t^n}\} = \{\widetilde{\mathbf{x_t^n}}\} = \emptyset$
3: for $n = 1$ to N do
4: sample $x_t^n \sim p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}^n})$
5: $w_t^n = p(\mathbf{z_t}|\mathbf{x_t^n}, map)$
6: $\{\widetilde{\mathbf{x_t^n}}\} = \{\widetilde{\mathbf{x_t^n}}\} + \langle x_t^n, w_t^n \rangle$
7: end for
8: for $n = 1$ to N do
9: draw i with probability $\propto \widetilde{w_t^i}$
10: $\{\mathbf{x_t^n}\} = \{\mathbf{x_t^n}\} + \langle \widetilde{x_t^i}, \widetilde{w_t^i} \rangle$
11: end for
12: return $\{\mathbf{x_t^n}\}$
13: end procedure

PARTICLE FILTER

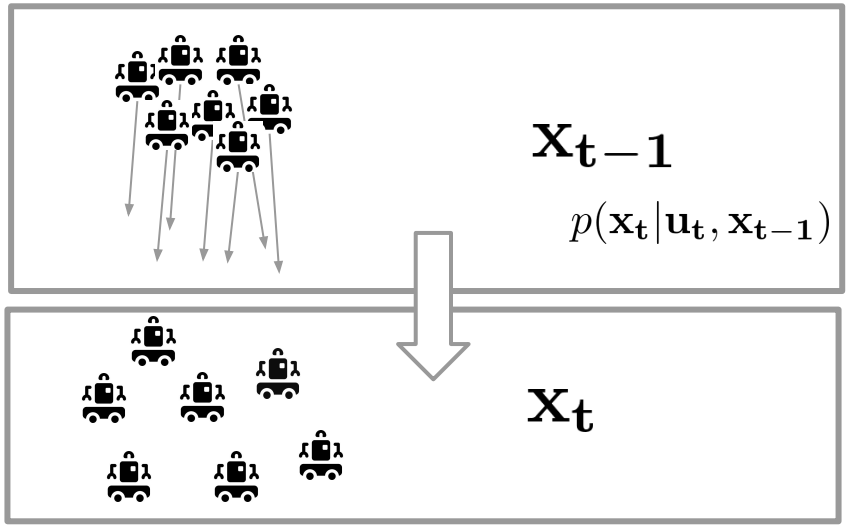


$\mathbf{X_{t-1}}$

Algorithm 1 Generic Monte-Carlo localization algorithm

1: procedure MCL($\mathbf{x_{t-1}}, m, \mathbf{u_t}, \mathbf{z_t}$)
2: $\{\mathbf{x_t^n}\} = \{\widetilde{\mathbf{x_t^n}}\} = \emptyset$
3: for $n = 1$ to N do
4: sample $x_t^n \sim p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}^n})$
5: $w_t^n = p(\mathbf{z_t}|\mathbf{x_t^n}, map)$
6: $\{\widetilde{\mathbf{x_t^n}}\} = \{\widetilde{\mathbf{x_t^n}}\} + \langle x_t^n, w_t^n \rangle$
7: end for
8: for $n = 1$ to N do
9: draw i with probability $\propto \widetilde{w_t^i}$
10: $\{\mathbf{x_t^n}\} = \{\mathbf{x_t^n}\} + \langle \widetilde{x_t^i}, \widetilde{w_t^i} \rangle$
11: end for
12: return $\{\mathbf{x_t^n}\}$
13: end procedure

PARTICLE FILTER

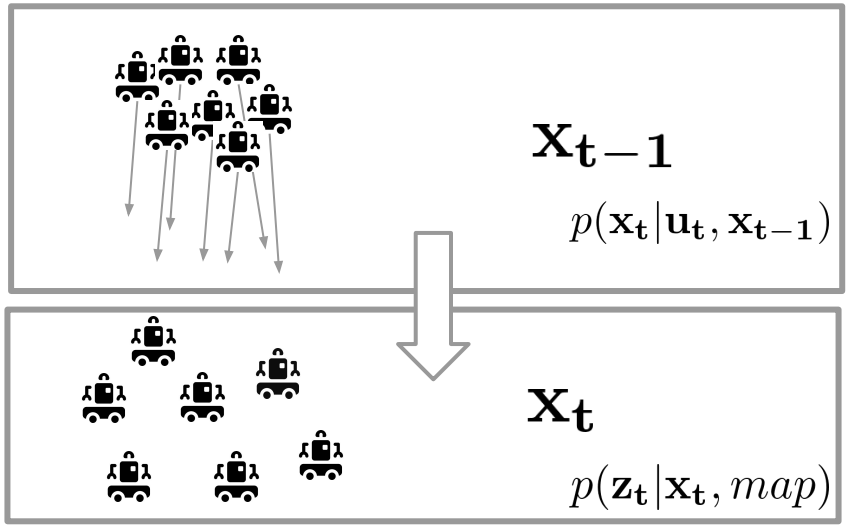


Algorithm 1 Generic Monte-Carlo localization algorithm

1: procedure MCL($\mathbf{x_{t-1}}, m, \mathbf{u_t}, \mathbf{z_t}$)
2: $\{\mathbf{x_t^n}\} = \{\widetilde{\mathbf{x_t^n}}\} = \emptyset$
3: for $n = 1$ to N do
4: sample $x_t^n \sim p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}^n})$
5: $w_t^n = p(\mathbf{z_t}|\mathbf{x_t^n}, map)$
6: $\{\widetilde{\mathbf{x_t^n}}\} = \{\widetilde{\mathbf{x_t^n}}\} + \langle x_t^n, w_t^n \rangle$
7: end for
8: for $n = 1$ to N do
9: draw i with probability $\propto \widetilde{w_t^i}$
10: $\{\mathbf{x_t^n}\} = \{\mathbf{x_t^n}\} + \langle \widetilde{x_t^i}, \widetilde{w_t^i} \rangle$
11: end for
12: return $\{\mathbf{x_t^n}\}$
13: end procedure

Motion model

PARTICLE FILTER

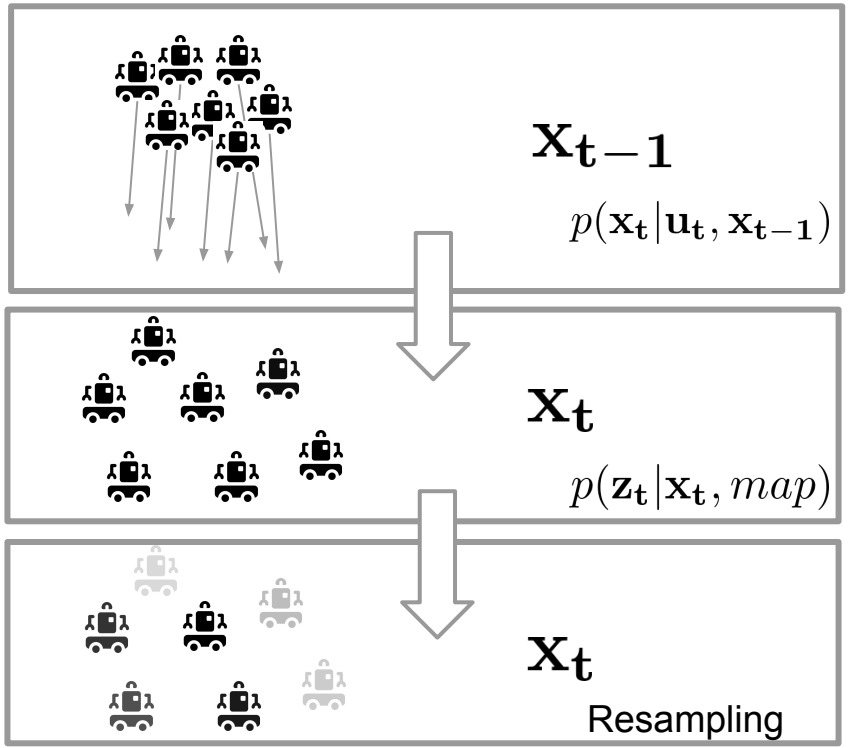


Algorithm 1 Generic Monte-Carlo localization algorithm

1: procedure MCL($\mathbf{x_{t-1}}, m, \mathbf{u_t}, \mathbf{z_t}$)
2: $\{\mathbf{x_t^n}\} = \{\widetilde{\mathbf{x_t^n}}\} = \emptyset$
3: for $n = 1$ to N do
4: sample $x_t^n \sim p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}^n})$
5: $w_t^n = p(\mathbf{z_t}|\mathbf{x_t^n}, map)$
6: $\{\widetilde{\mathbf{x_t^n}}\} = \{\widetilde{\mathbf{x_t^n}}\} + \langle x_t^n, w_t^n \rangle$
7: end for
8: for $n = 1$ to N do
9: draw i with probability $\propto \widetilde{w_t^i}$
10: $\{\mathbf{x_t^n}\} = \{\mathbf{x_t^n}\} + \langle \widetilde{x_t^i}, \widetilde{w_t^i} \rangle$
11: end for
12: return $\{\mathbf{x_t^n}\}$
13: end procedure

Observation model

PARTICLE FILTER

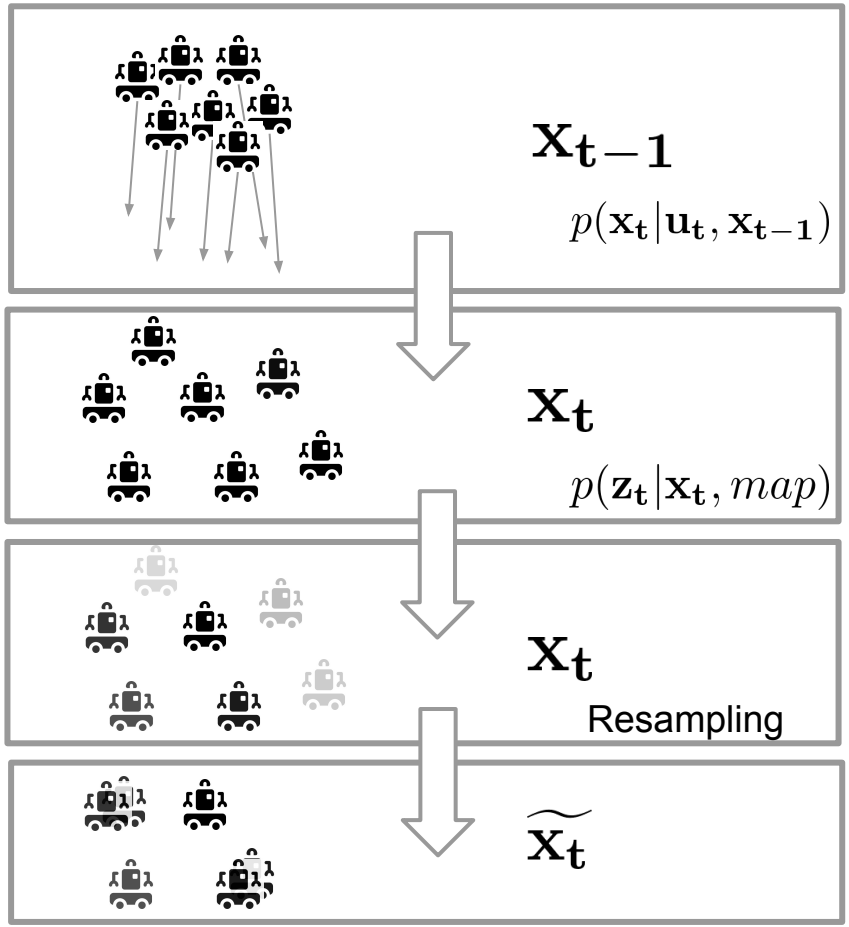


Algorithm 1 Generic Monte-Carlo localization algorithm

1: procedure MCL($\mathbf{x_{t-1}}, m, \mathbf{u_t}, \mathbf{z_t}$)
2: $\{\mathbf{x_t^n}\} = \{\widetilde{\mathbf{x_t^n}}\} = \emptyset$
3: for $n = 1$ to N do
4: sample $x_t^n \sim p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}^n})$
5: $w_t^n = p(\mathbf{z_t}|\mathbf{x_t^n}, map)$
6: $\{\widetilde{\mathbf{x_t^n}}\} = \{\widetilde{\mathbf{x_t^n}}\} + \langle x_t^n, w_t^n \rangle$
7: end for
8: for $n = 1$ to N do
9: draw i with probability $\propto \widetilde{w_t^i}$
10: $\{\mathbf{x_t^n}\} = \{\mathbf{x_t^n}\} + \langle \widetilde{x_t^i}, \widetilde{w_t^i} \rangle$
11: end for
12: return $\{\mathbf{x_t^n}\}$
13: end procedure

Resampling

PARTICLE FILTER



Algorithm 1 Generic Monte-Carlo localization algorithm

1: procedure MCL($\mathbf{x_{t-1}}, m, \mathbf{u_t}, \mathbf{z_t}$)
2: $\{\mathbf{x_t^n}\} = \{\widetilde{\mathbf{x_t^n}}\} = \emptyset$
3: for $n = 1$ to N do
4: sample $x_t^n \sim p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}^n})$ — Motion model
5: $w_t^n = p(\mathbf{z_t}|\mathbf{x_t^n}, map)$ — Observation model
6: $\{\widetilde{\mathbf{x_t^n}}\} = \{\widetilde{\mathbf{x_t^n}}\} + \langle x_t^n, w_t^n \rangle$
7: end for
8: for $n = 1$ to N do
9: draw i with probability $\propto \widetilde{w_t^i}$ — Resampling
10: $\{\mathbf{x_t^n}\} = \{\mathbf{x_t^n}\} + \langle \widetilde{x_t^i}, \widetilde{w_t^i} \rangle$
11: end for
12: return $\{\mathbf{x_t^n}\}$
13: end procedure

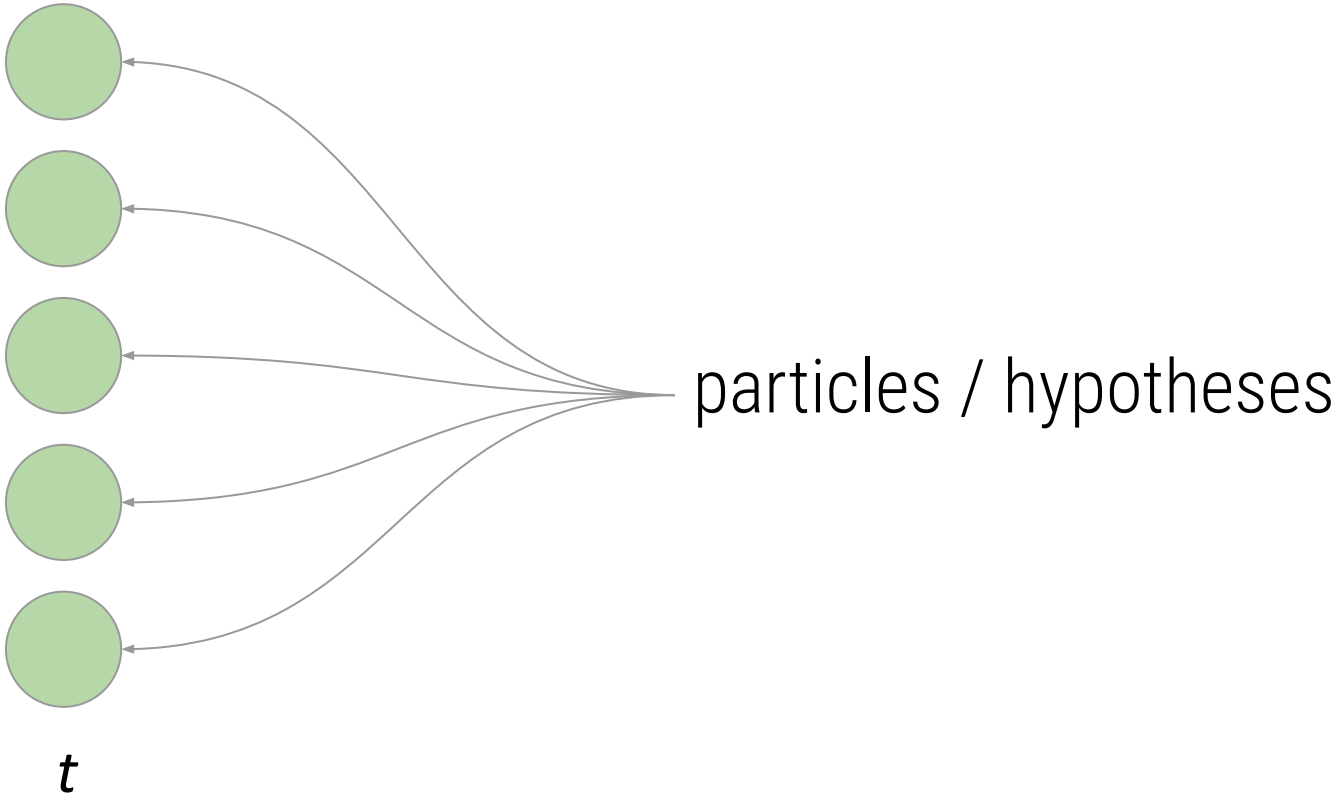
PARTICLE FILTER
MOTION MODEL

NEXT LECTURE

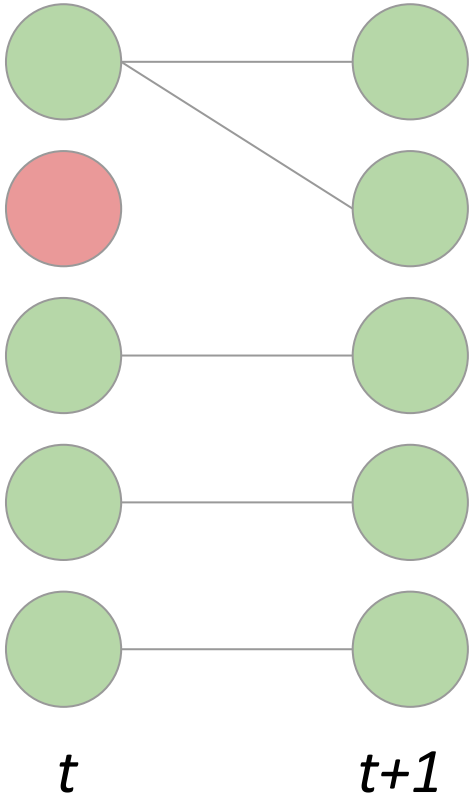
PARTICLE FILTER
OBSERVATION MODEL

NEXT LECTURE

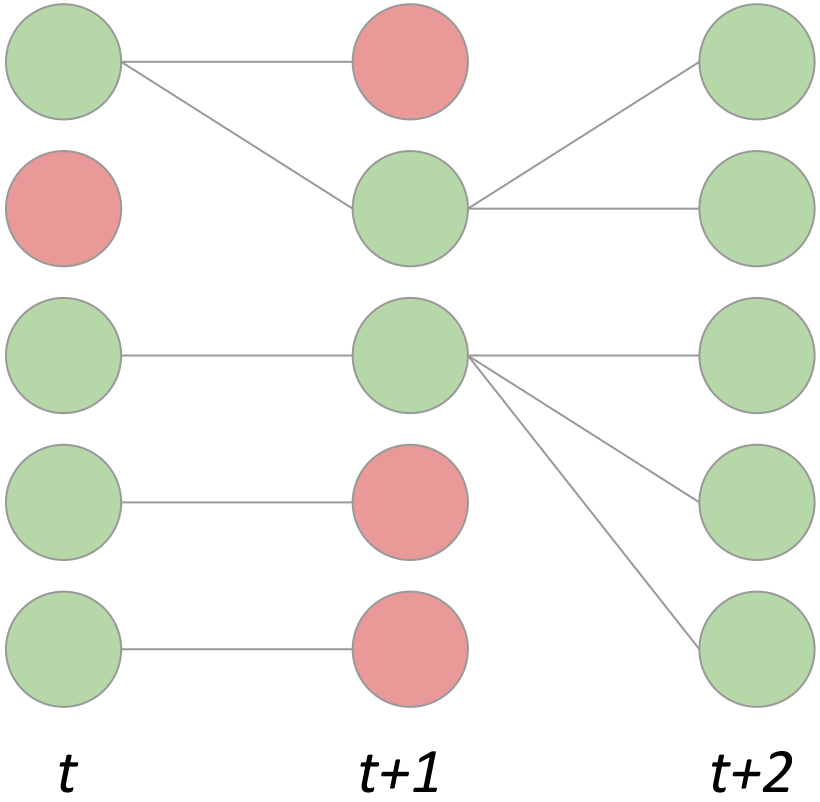
PARTICLE FILTER
RESAMPLING



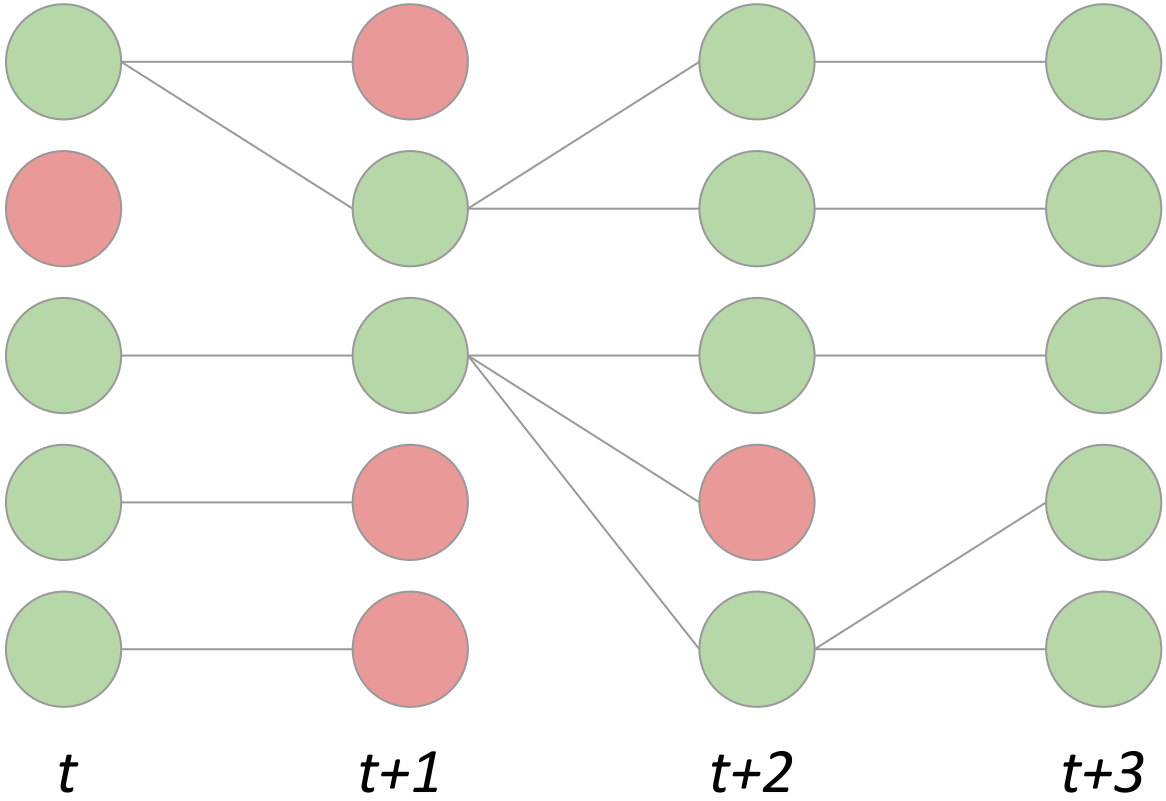
PARTICLE FILTER
RESAMPLING



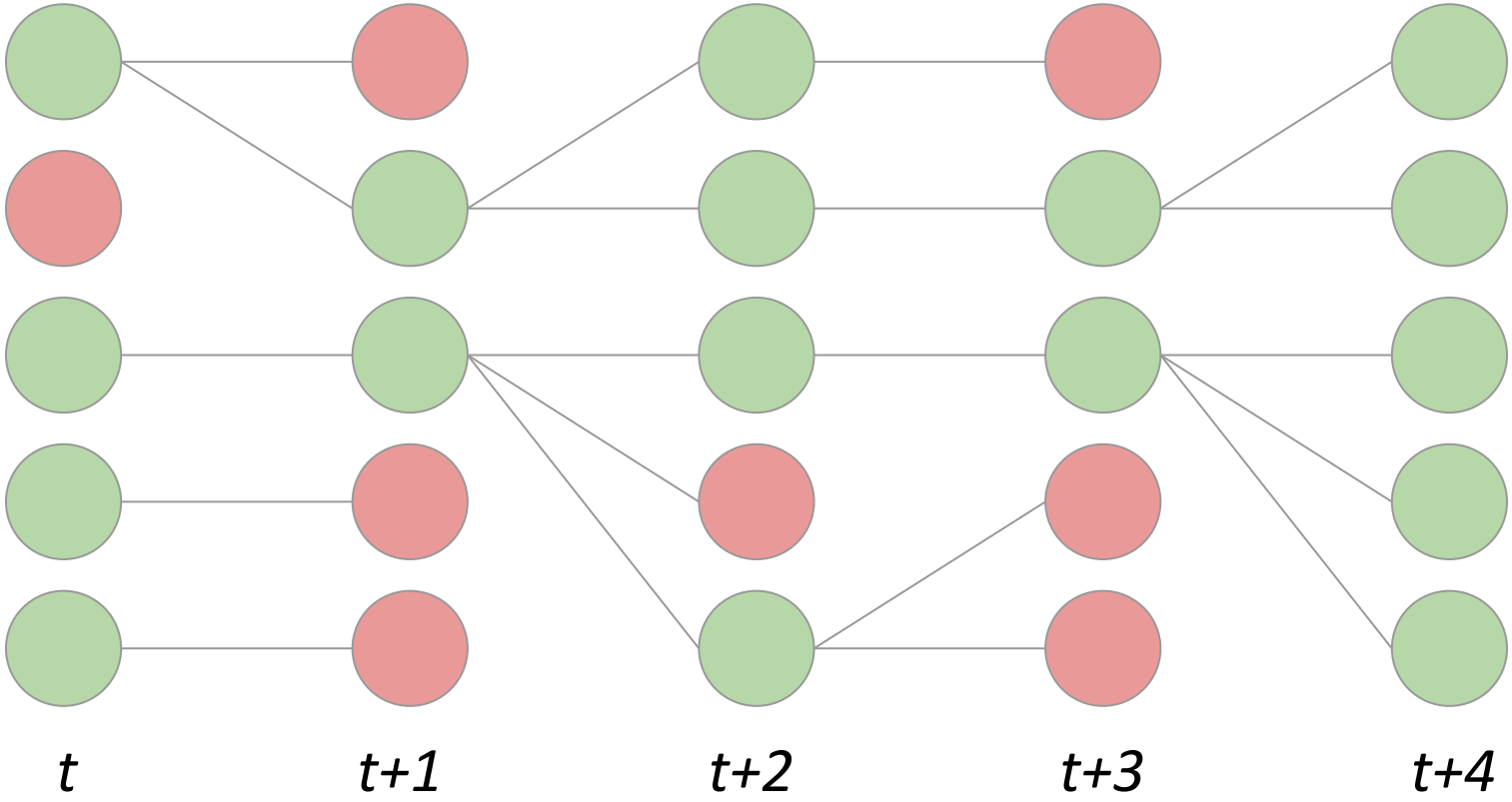
PARTICLE FILTER
RESAMPLING



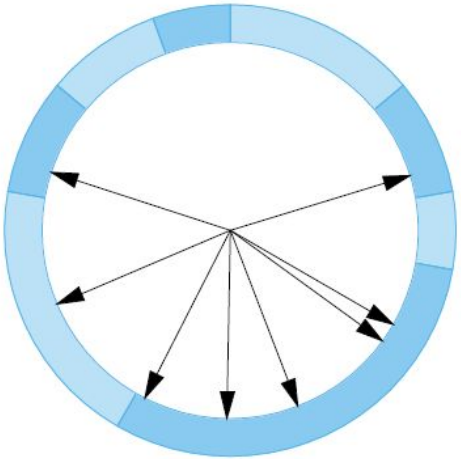
PARTICLE FILTER
RESAMPLING



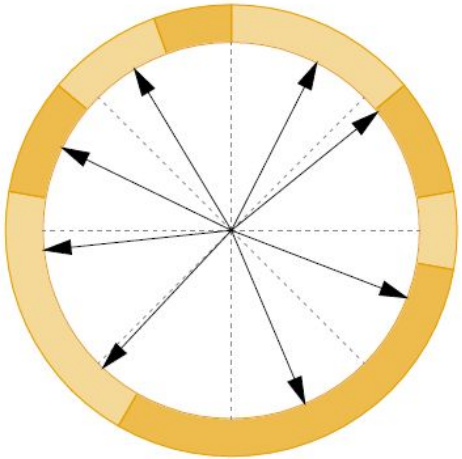
PARTICLE FILTER
RESAMPLING



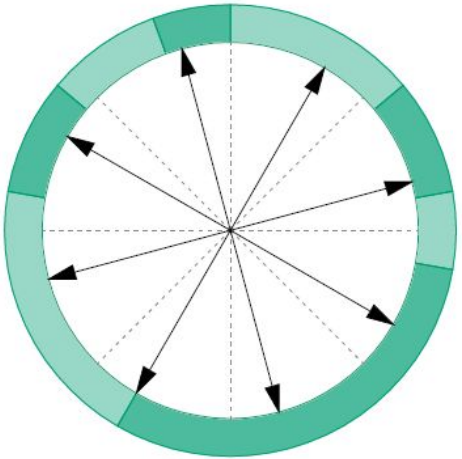
PARTICLE FILTER
RESAMPLING



(a) Multinomial

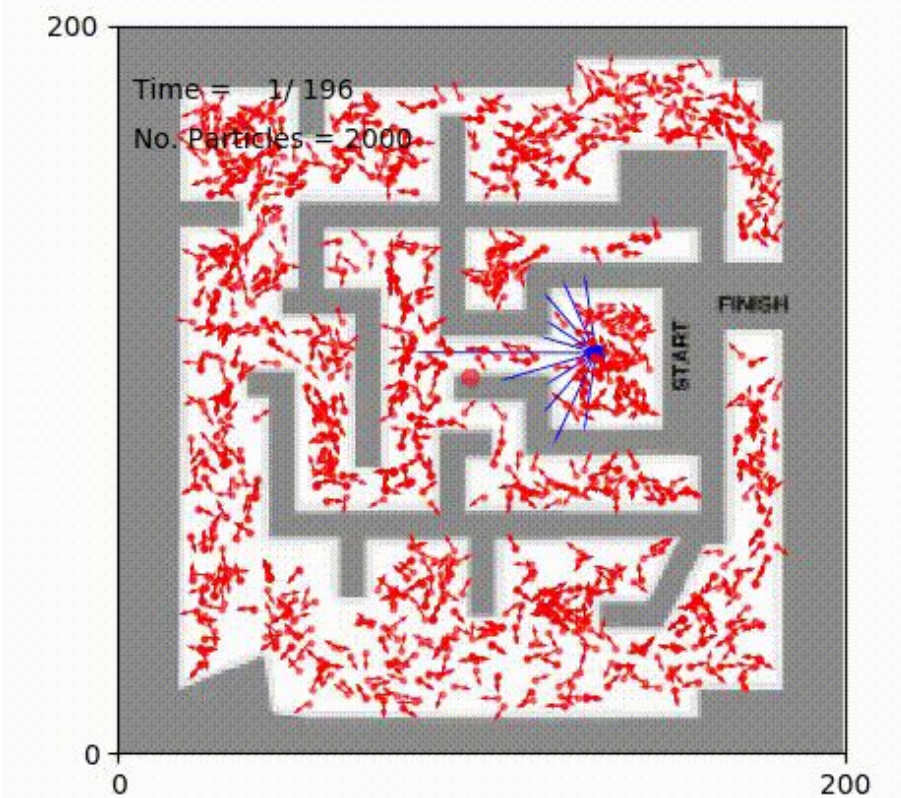


(b) Stratified

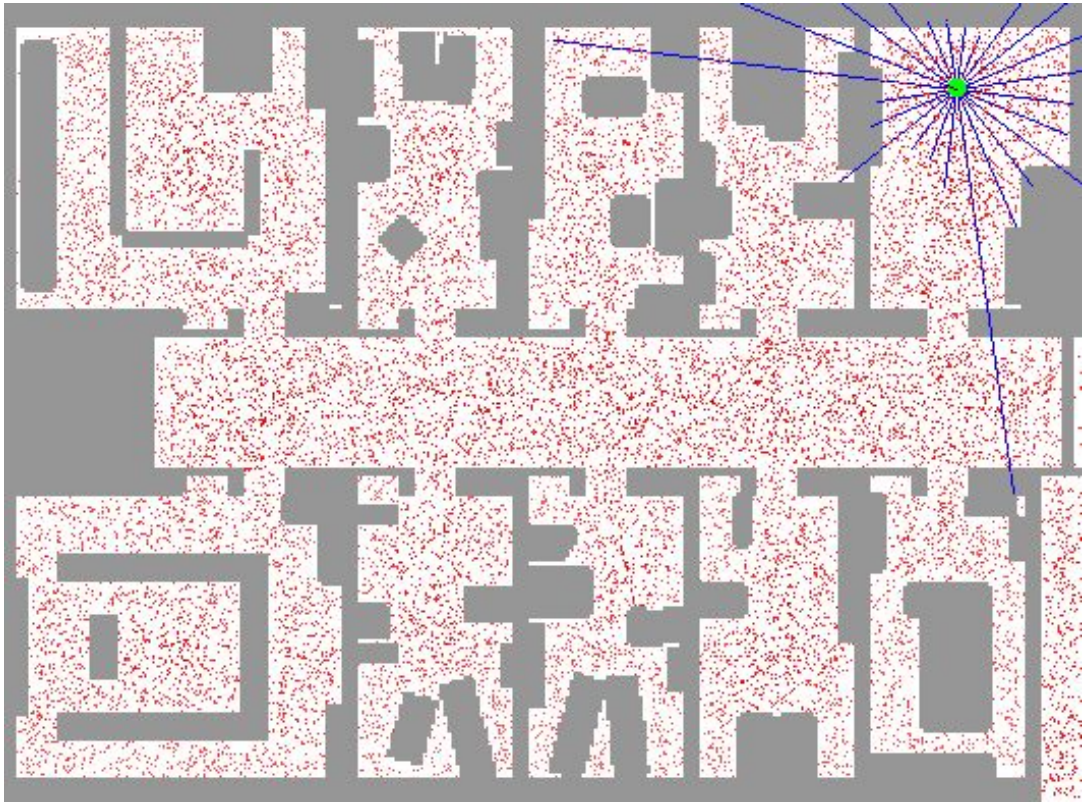


(c) Systematic

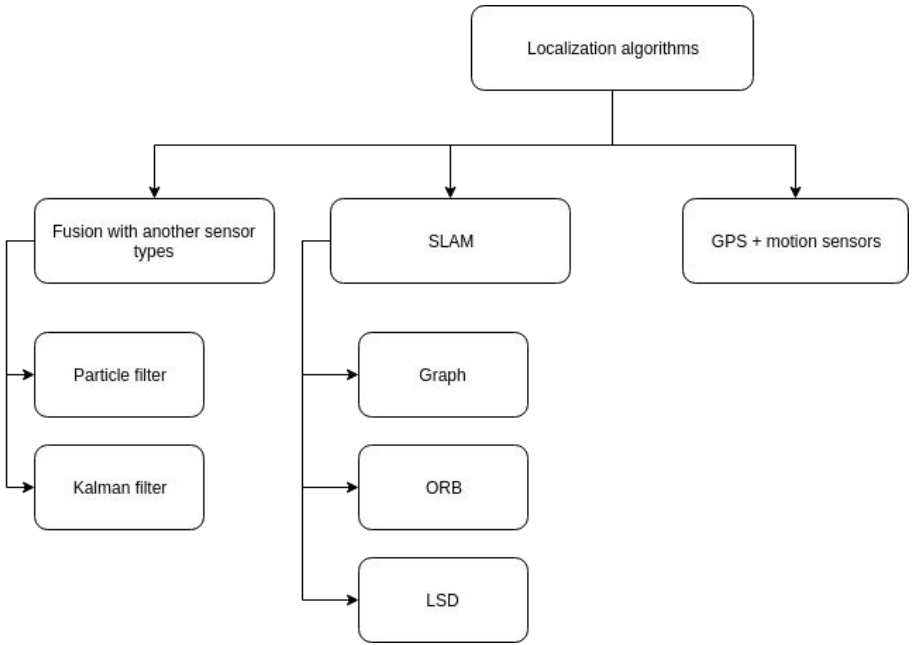
PARTICLE FILTER
EXAMPLE



GLOBAL LOCALIZATION AND KIDNAPPED ROBOT PROBLEM

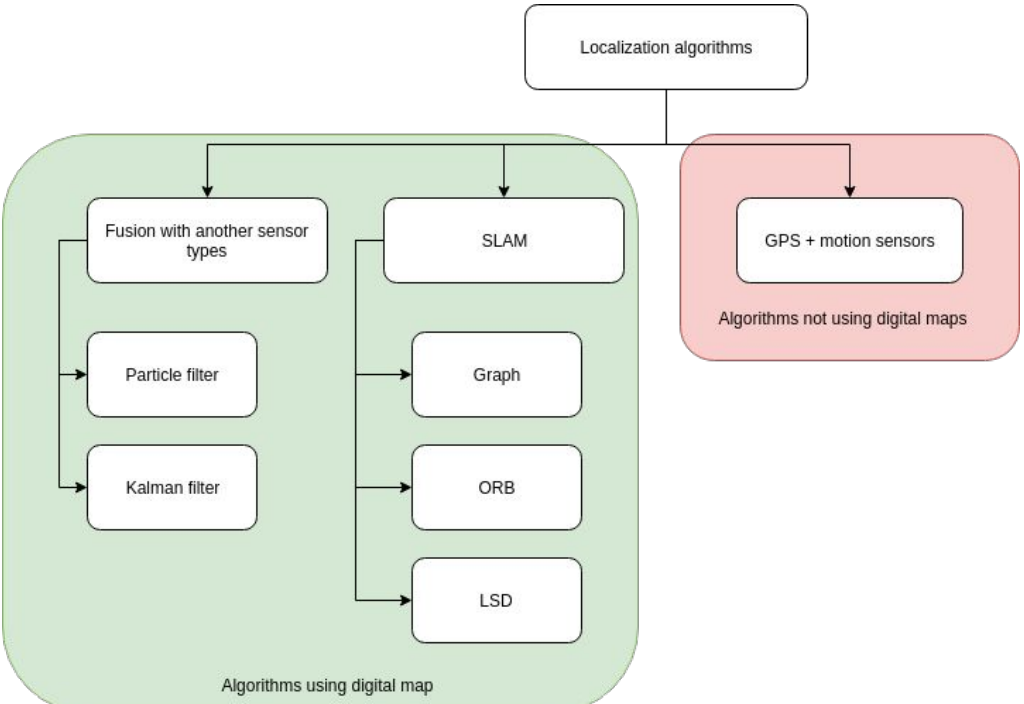


WHICH ONE TO CHOOSE?



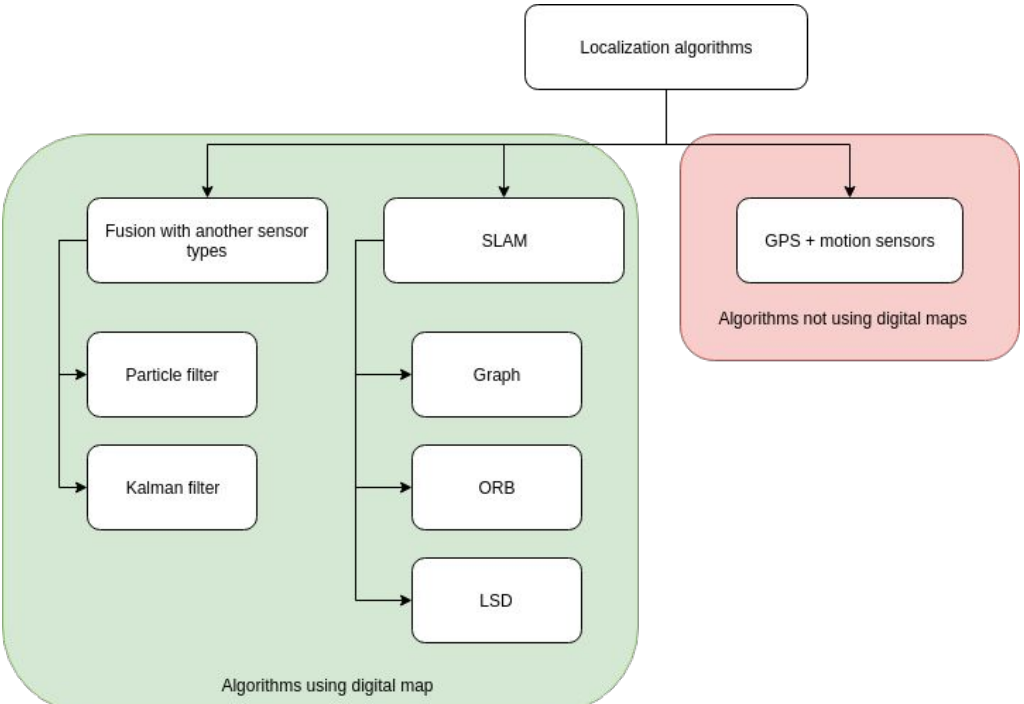
	Kalman filter	Particle filter
Noise type	Gaussian (normal), unimodal	Arbitrary multimodal distribution
Solution	Optimal	Approximate
Implementation difficulty	Hard (derivation of physical model needed)	Simple
Robustness	Moderate	High
Global localization	No	Yes
Computational speed	Fast	Depends on implementation (usually, slower)

WHICH ONE TO CHOOSE?



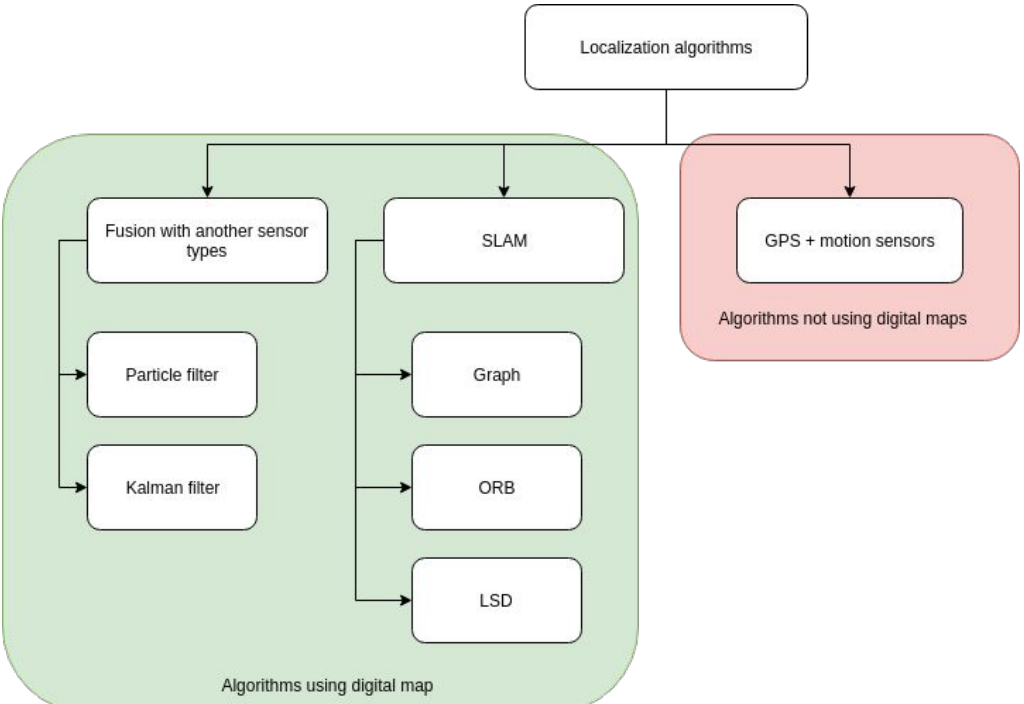
	Kalman filter	Particle filter
Noise type	Gaussian (normal), unimodal	Arbitrary multimodal distribution
Solution	Optimal	Approximate
Implementation difficulty	Hard (derivation of physical model needed)	Simple
Robustness	Moderate	High
Global localization	No	Yes
Computational speed	Fast	Depends on implementation (usually, slower)

WHICH ONE TO CHOOSE?



	Kalman filter	Particle filter
Noise type	Gaussian (normal), unimodal	Arbitrary multimodal distribution
Solution	Optimal	Approximate
Implementation difficulty	Hard (derivation of physical model needed)	Simple
Robustness	Moderate	High
Global localization	No	Yes
Computational speed	Fast	Depends on implementation (usually, slower)

WHICH ONE TO CHOOSE?



	Extended Kalman filter	Particle filter
Noise type	Arbitrary, unimodal	Arbitrary multimodal distribution
Solution	Approximate	Approximate
Implementation difficulty	Hard (derivation of physical model needed)	Simple
Robustness	Moderate	High
Global localization	No	Yes
Computational speed	Fast	Depends on implementation (usually, slower)

ADDITIONAL RESOURCES

1. How a Kalman filter works, in pictures
2. Probabilistic Robotics (in PIAZZA).
   Chapters 1, 2, 3, 4, 7, 8.
3. Cyrill Stachniss. SLAM -Course

Thanks for attention!

Questions? Additions? Welcome!

---

girafe
ai