"Where Am I?" - An Introduction towards Localization, Mapping and Navigation in Modern Robotic Systems

Presentation by

Sebastian Starke

University of Hamburg
Department of Computer Science

Overview

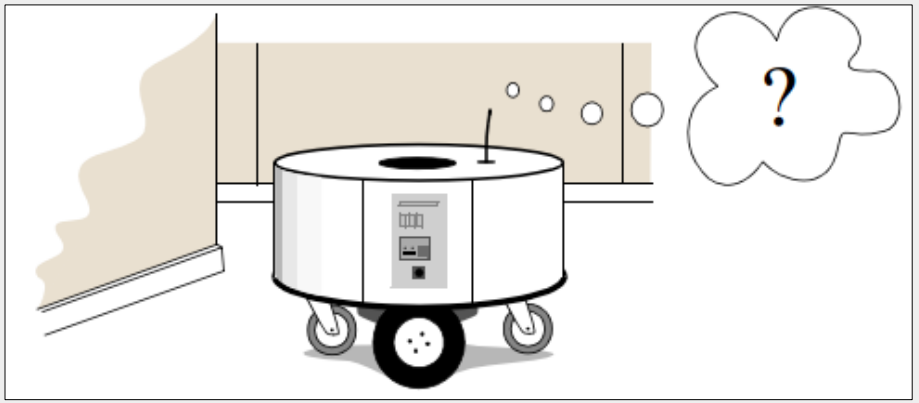
Contents

- Introduction & General Knowledge
- Sensors
- Basic Concepts
- Algorithms

Introduction & General Knowledge

Introduction & General Knowledge

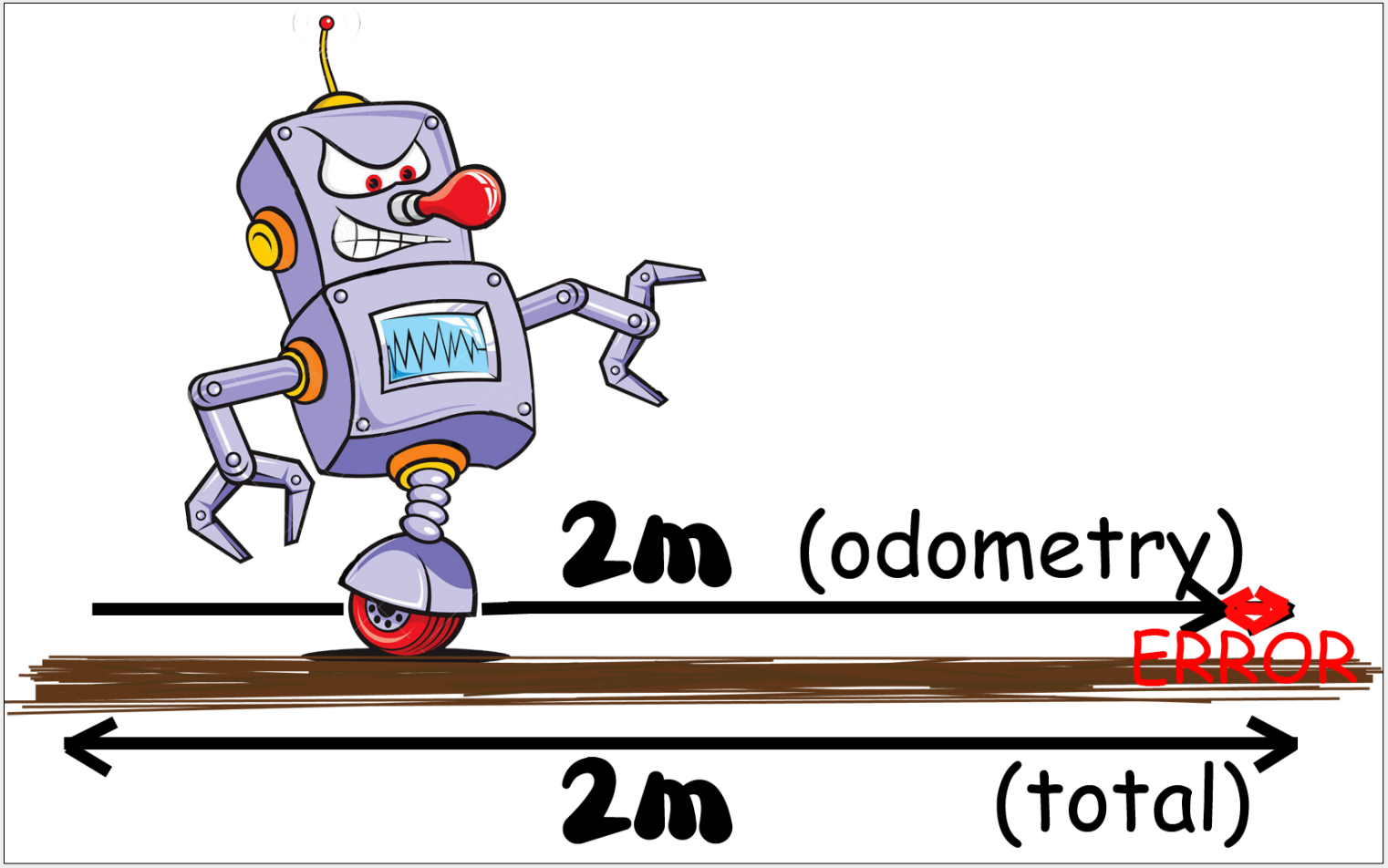
Where Am I?



- Robots find themselves asking this question frequently!
- Localization is one of the most fundamental problems in robotics of all time! *(...and will probably always be!)*
- Localization sounds pretty easy... - **but it completely isn't!**

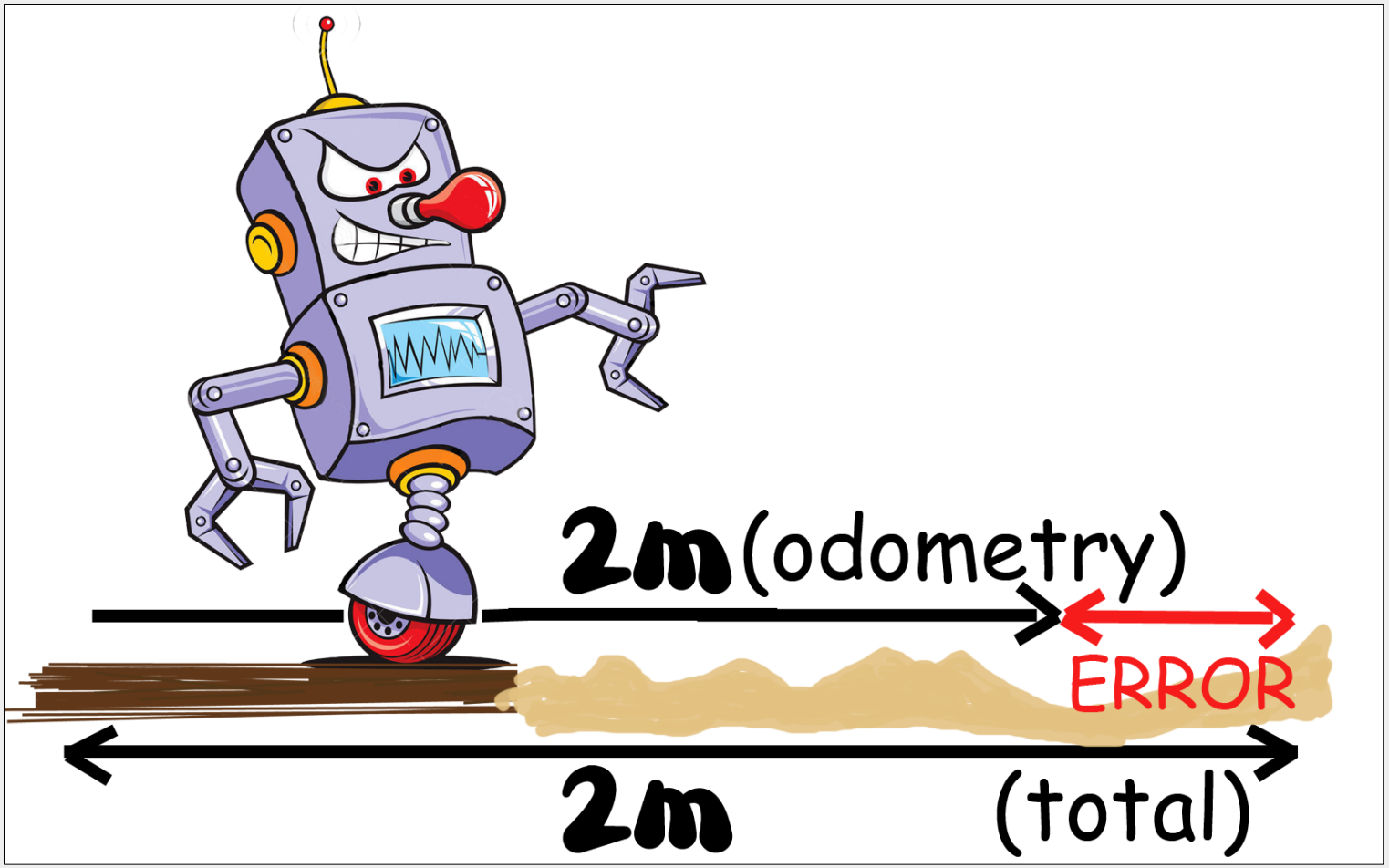
Introduction & General Knowledge

One simple example... on **flat** ground:



Introduction & General Knowledge

One simple example... on **slippery** ground:



So, why is Localization important?

- Knowing the actual location *(and being able to navigate)* is essentially important for **autonomous robots**!
- If a robot does not know where he is, it might be hard to determine **what to do next**!
- Robots have to **react adaptively** to their environment, which of course can dynamically change! *(imagine your dog fluffy or a small child)*

Let's split it up into 3 keywords:

(which are of course always in a correlation to each other)

Localization

Navigation

Mapping

Localization...

- ...is basically the act of *finding one's location* within a map.

(those are mostly in 2D, but can also be in 3D!)

- *„Location“*, *„Pose“* or *„Position“* actually means the

X and Y coordinates connected with the **heading direction** *(orientation)* of the robot within the map!

$$l = (x, y, \theta)$$

Localization...

Global Localization

→ *The robot is not told its initial position*

→ *It has to solve a much more difficult localization problem*

Local Localization

→ *One the robot has localized itself against a map,*
but it has to keep track of its actual position.

Navigation...

- ...describes the act of *moving through* a *known* or *unknown environment*.

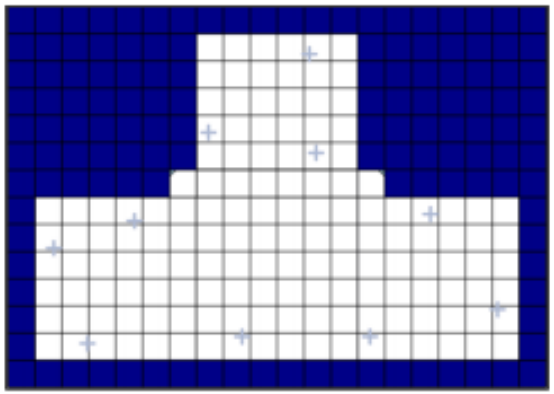
  Note: This also affects the part of localization!

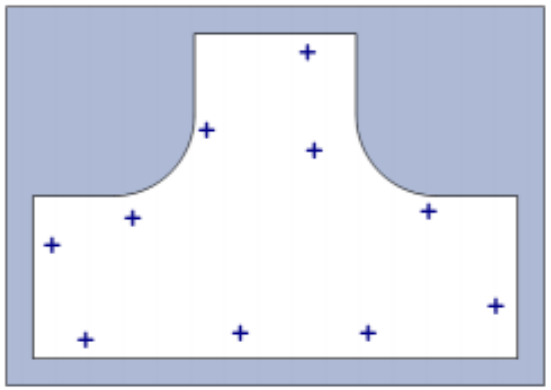
Mapping...

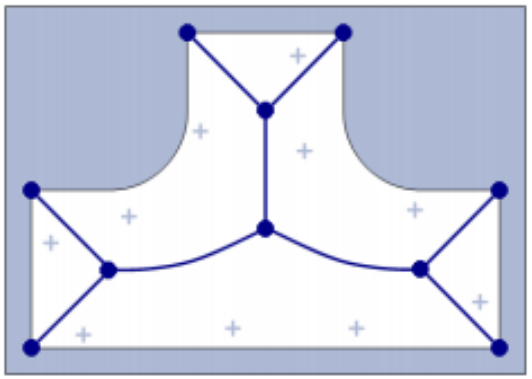
- ...is the problem, to *generate a map* from a **completely unknown** or **less known** map **or to keep updating** a given map.

What do these maps look like?

Metrical („grid-based pixel maps“)	**Feature-Based** (landmark locations)	**Topological** (nodes and connections)







What if we don't have a map?

We have both to...

build the map
AND
to do localization

... at once!

(This idea is generally known as SLAM)
*(„**S**imultaneous **L**ocalization **a**nd **M**apping“... later more!)*

Sensors

Sensors

- Using sensors is essentially needed to grant information to the robot!
- Sensors always underly a certain noise *(mostly normal-distributed)* which can also differ in some environment! *(f.e. Ultrasonic-sensors)*
- Almost always > 1 sensor is used what leads to towards *sensor-fusion*!
  (In general, this is done by the Kalman-Filter, later more...)

Sensors

Odometry-sensors
→ Magnetic compass (absolute heading)
→ Gyroscope (change of heading)
→ Acceleration sensors (acceleration)
→ Tachometer (speed, distance)

Distance-sensors
→ Infrared
→ Laserscanner

Sound-sensors
→ Microphone

Cameras
→ 2D
→ 3D

List of various sensors for robotic systems:
http://www.robotshop.com/en/sensors.html

Sensors

- **GPS (Global Positioning System)**
  → exact up to some metres

- **One Improvement: Differential GPS**
  → A fixed GPS-station on the world *(location known)* is able to calculate the error by using GPS
  → If the robot *(location unknown)* calculates the distance to the satellite knowing the approximate error, it can be much more exact! *(up to centimetres)*

Sensors

- **One more improvement of GPS is RTK *(Realtime Kinematic Satellite Navigation)***
  → Technique, used to enhance the precision of satellite-based positioning systems even more!
  → Uses measurements of the signal's carrier wave rather than the information content of the signal.

- **RTK+GPS = CPGPS** *(Carrier-Phase-GPS)*

Sensors

Odometry Sensors

+

Indoors?

We might use IR-sensors or 2D-Laserscanners since we are mostly concerned with obstacles!

Outdoors?

We might use GPS / CPGPS or even a Compass-sensor to localize ourself!

Note: GPS does not work properly inside buildings!

Sensors

Use the right sensors for the right applications!

•

Noise induces a limitation of the consistency given by sensor readings!

•

Not handling these errors causes them to accumulate rapidly!

•

Sensor-Fusion is extremely important!

Basic Concepts

Basic Concepts

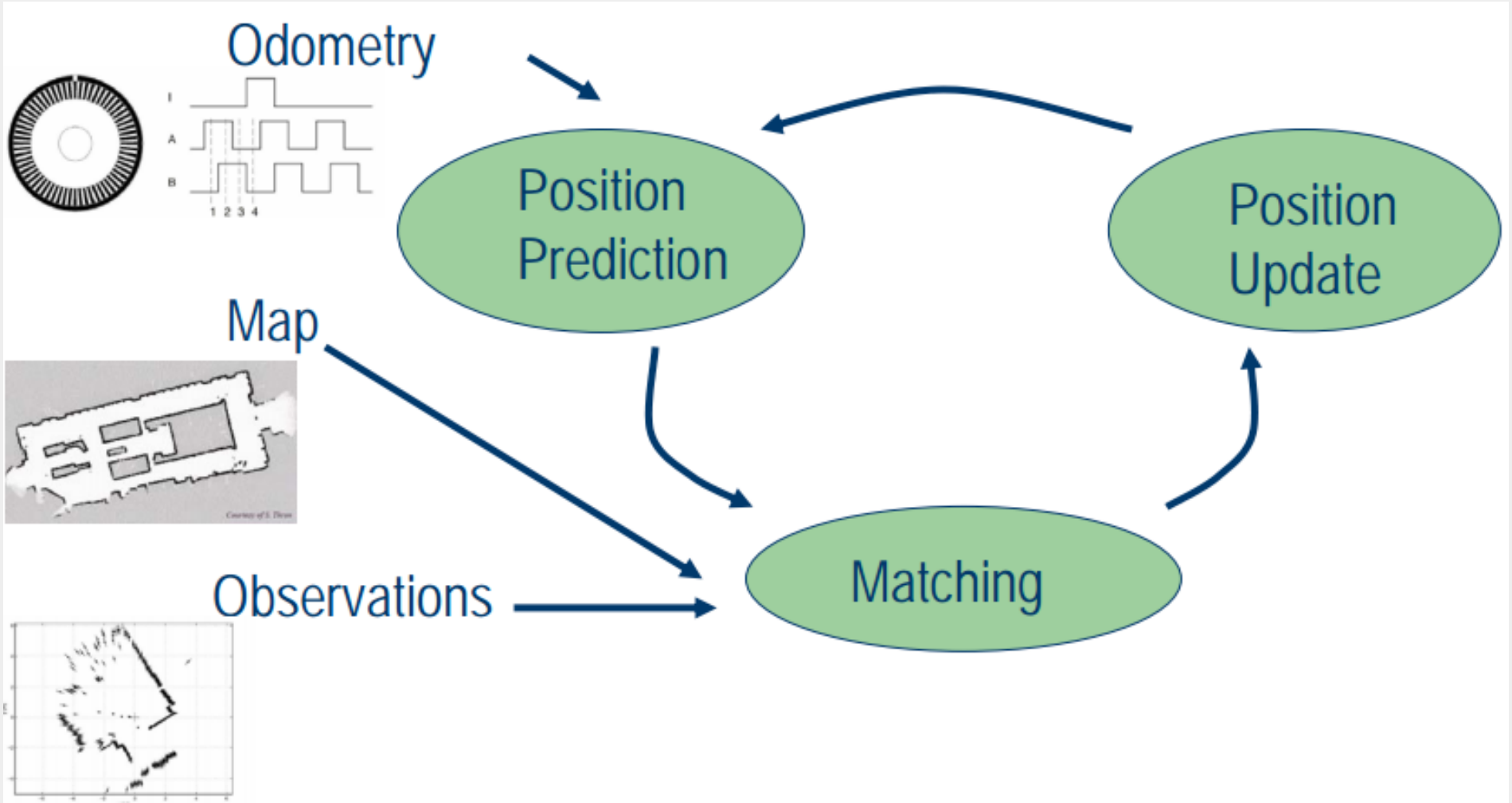
Navigation → Localization → Mapping

...is done in an iterative way!

- **We will typically do the following:**

1 – Move some distance *(this will induce some error!)*
2 – Take some sensor measurements *(get some noisy data)*
3 – Process/Fuse these sensor data (*reduce the noise)*
4 – Localize ourselves against the map *(using the data)*
5 – If we are building a map as well → update it
6 – Set a new course if necessary and repeat

Basic Concepts



Basic Concepts

There are two fundamental strategies helping us to localize ourselves...

<u>Probabilistic Methods</u>

→ Play some very important role over all! *(f.e. Bayesian-Belief-Networks for multiple sensor readings)*

<u>Prior Knowledge</u>

→ This is mostly needed, the more accurate, the better! *(f.e. sensor-noise, start-pos, ...)*

Probabilistic Methods

→ concerned with the „uncertainty“ of events occurring!

- Assume X is some event, so P(X) is its probability to happen (between „0“ and „1“, normal-distributed)

- In context of localization:
  → *The most likely location is the one with the greatest likelihood!*

Basic Concepts

Prior Knowledge

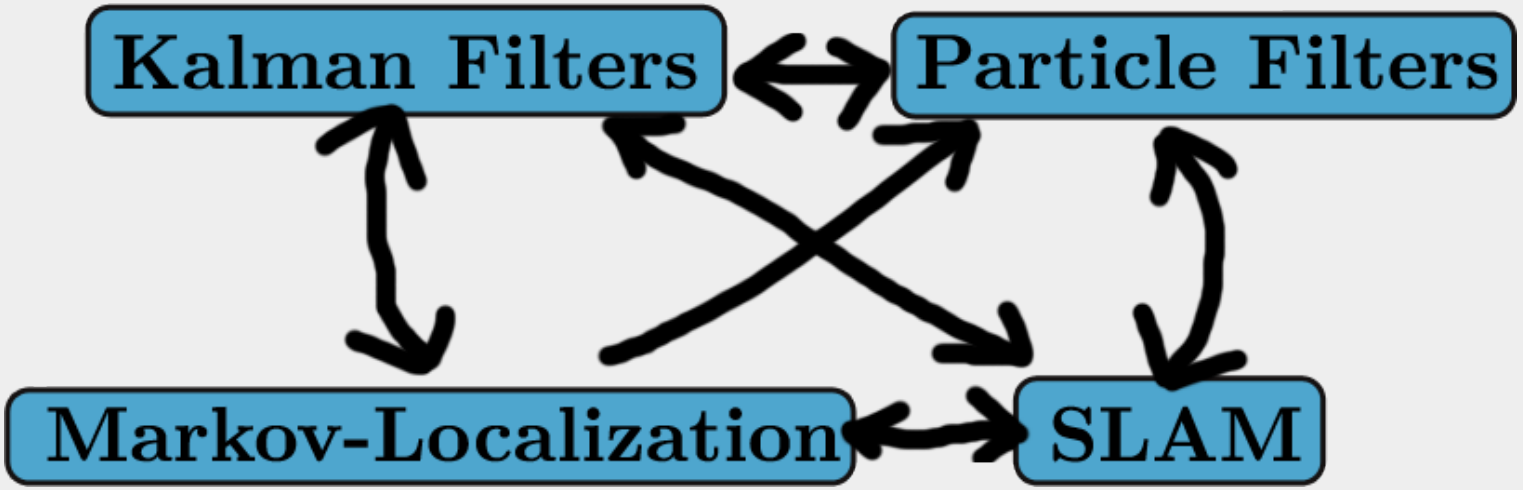
→ *provides us infomation about the system and environment!*

- Probabilistic approaches are always concerned with Prior Knowledge!
- It is always important to have **good** and **meaningful** priors!

Algorithms

Algorithms

The Algorithms for Localization are often being combined!



Some SLAM-Examples:

→ Extended-Kalman-Filter-SLAM

→ Particle-Filter-SLAM

→ Markov-Chain-Monte-Carlo-SLAM

→ ...

Algorithms

Kalman-Filters

- Describe a class of several filters dealing with **mathematical equations** *(linear, extended, unscented, ...)*
- Mostly **pre-applied** to other algorithms by doing **sensor-fusion** from normal distributed error measurements
- Use knowledge of the **probabilistic sensor errors** and **dynamics**, the known **noise** and the **system**!
- **GOAL:** Recursively remove the noise from the sensor data!

Kalman-Filters (example: linear)

The *(linear)* Kalman-Filter estimates the state of a noisy linear dynamic system

→ Given a linear dynamical system:

$$x(k+1) = F(k)x(k) + G(k)u(k) + v(k)$$
$$y(k) = H(k)x(k) + w(k)$$

...in which:

$x(k)$ is the n-dimensional state vector (unknown)

$u(k)$ is the m-dimensional input vector (known)

$y(k)$ is the p-dimensional output vector (known, measured)

$F(k), G(k), H(k)$ are appropriately dimensioned system matrices (known)

$v(k), w(k)$ are zero-mean, white Gaussian noise with (known) covariance matrices $Q(k), R(k)$

Kalman-Filters

The great things about Kalman-Filters

Noise smoothing

Good state estimation

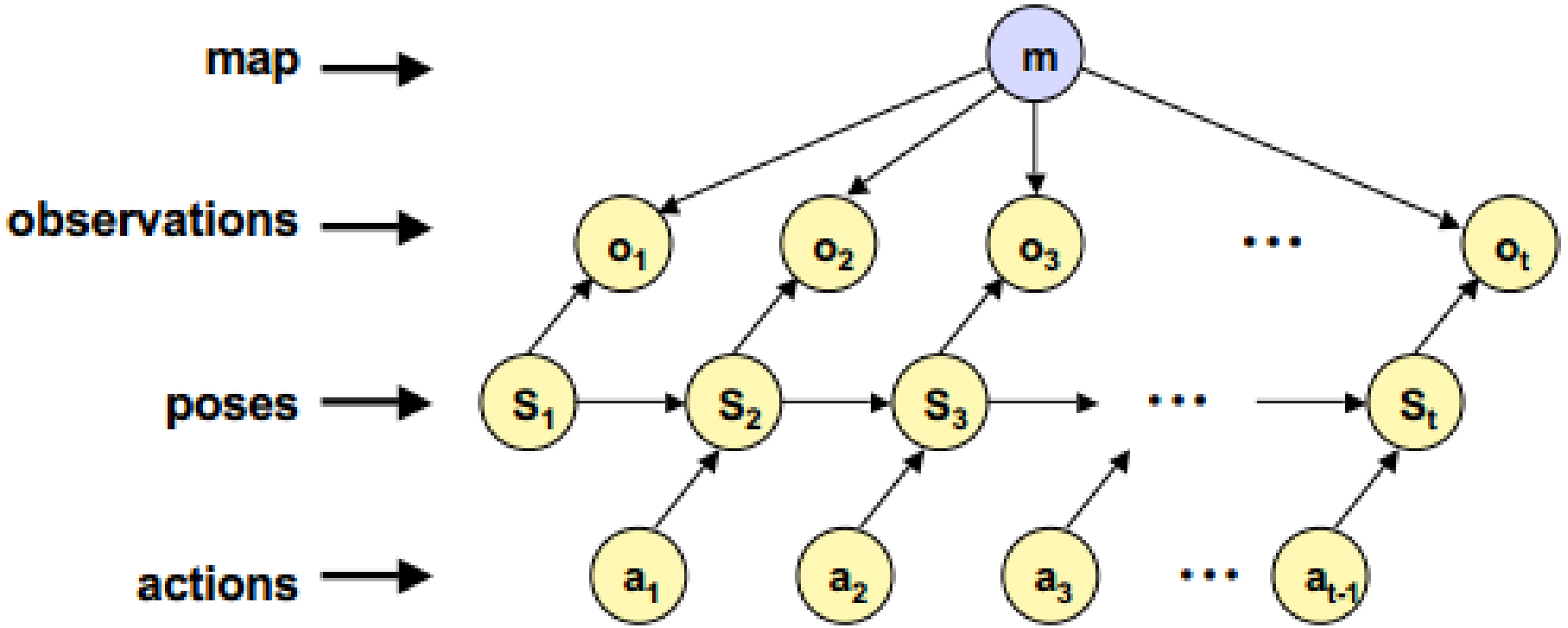
Recursive (low computational cost!!!)

Markov-Localization

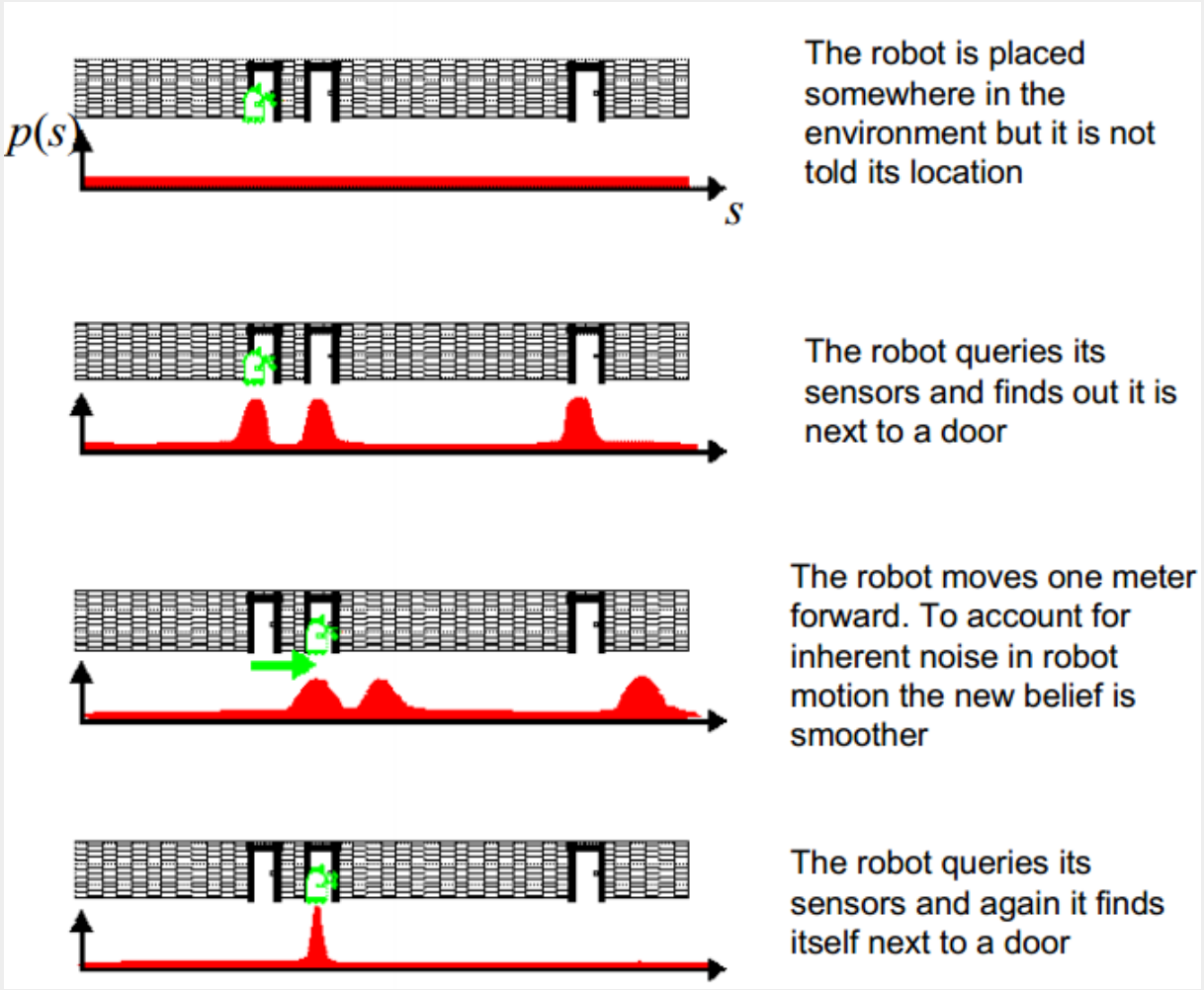
Robot-pose $l = (x, y, \theta)$

- Solution is born out of probability theory.
- Tracks the robot's belief „states“ of positions by each step
  - *“Prediction Phase”* → *Where the robot thinks he is*
  - *“Update-Phase”* → *The probabilities of all states get updated*
- Has a configuration space that is a finite discrete number of possible robot poses in the map.
  Note: There can be hundred to millions of possible positions!

Markov-Localization



Markov-Localization



Markov-Localization

→ Video for Markov-Localization using Bayesian Belief Networks:

https://www.youtube.com/watch?v=XFoGDvTOR28

Markov-Localization

The Markov-Localization encounters problems in dynamic environments, but works well in static environments!

•

On large maps:
Quite high computational costs to update the cells, since at **each iteration** the probability of **each state** within the **entire space** is updated!

Particle-Filters

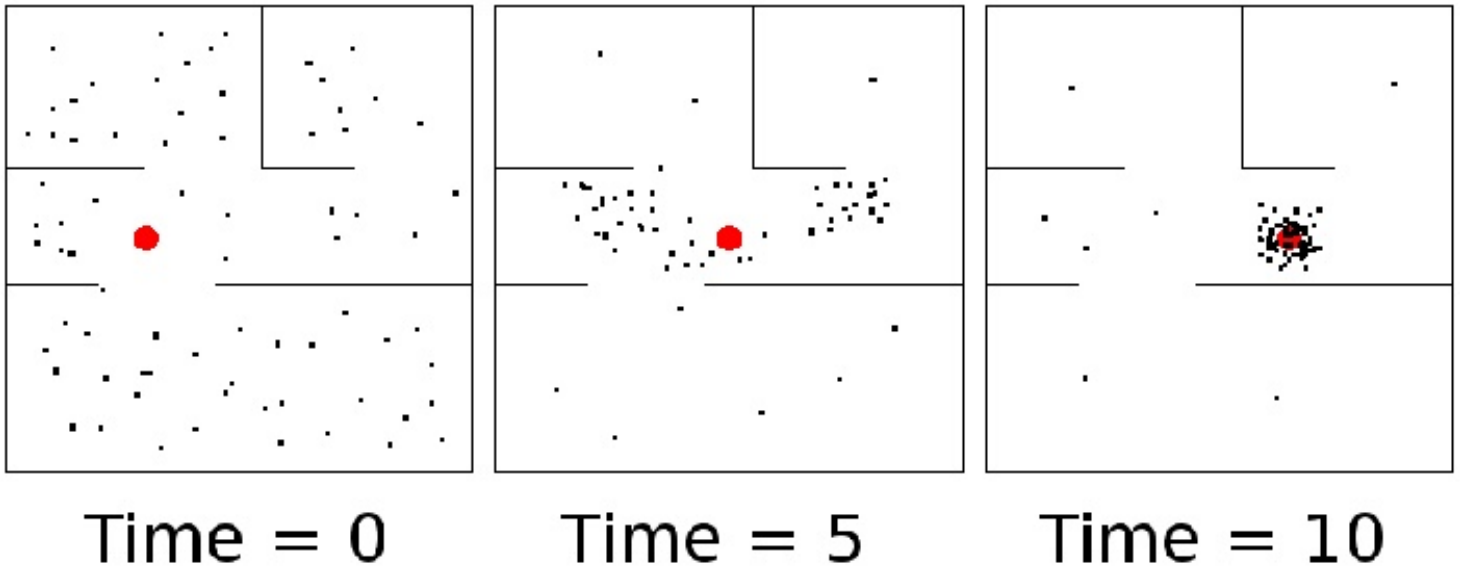
- **Monte-Carlo-Localization (MCL)**
  (Subgroup of Markov-Localization)

  Robot-pose $l = ((x, y, \theta), p)$

  → *Start:* uniformly distributed particles over the map
  → With each step, particles with a smaller likelihood than others are replaced by new ones in accumulated areas
  → Over the iterations, the particles converge against the area(s) where the robot's likelihood to be is the greatest!

  Note: Computational costs might also be quite high if there are hundreds or thousands of particles!

Particle-Filters



Particle-Filters

- **Adaptive-Monte-Carlo-Localization (AMCL)**

  → Limited minimum and maximum number of particles
  ***Note: NEVER less and NEVER more*!**

  → This reduces the potential computational cost dramatically! ***(great advantage to other algorithms)***

By the way: This one is used in ROS :-) See the video below!
https://www.youtube.com/watch?v=F6T3dtXviNY

Algorithms

SLAM *(„Simultaneous Localization **and** Mapping“)*

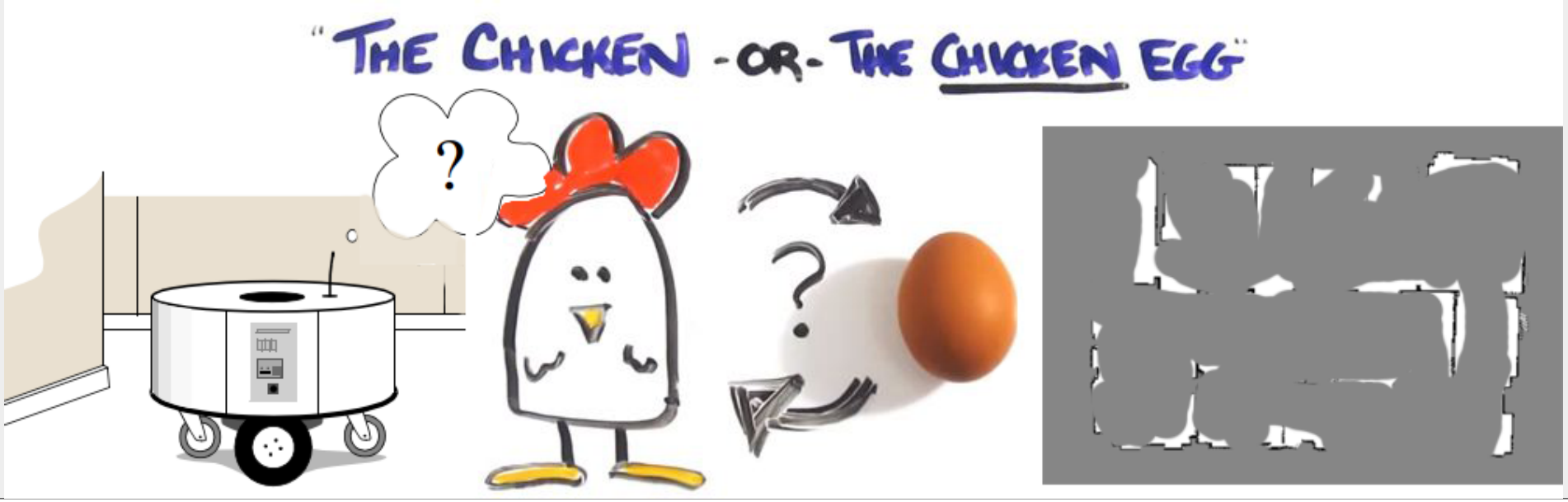
- Important, because there might be several reasons for which a map does not have to exist already!
  (or is only barely started)
- We need a technique, that is able to automatically explore and/or to update a new *(unknown or less known)* environment!
- The “*Chicken And Egg*” problem occurs!

The „Chicken and Egg“-problem occurs:

To apply our sensor measurements on our map, we have to know our localization, but...

What if our map is incomplete or barely started?

How can we localize ourself?



Algorithms

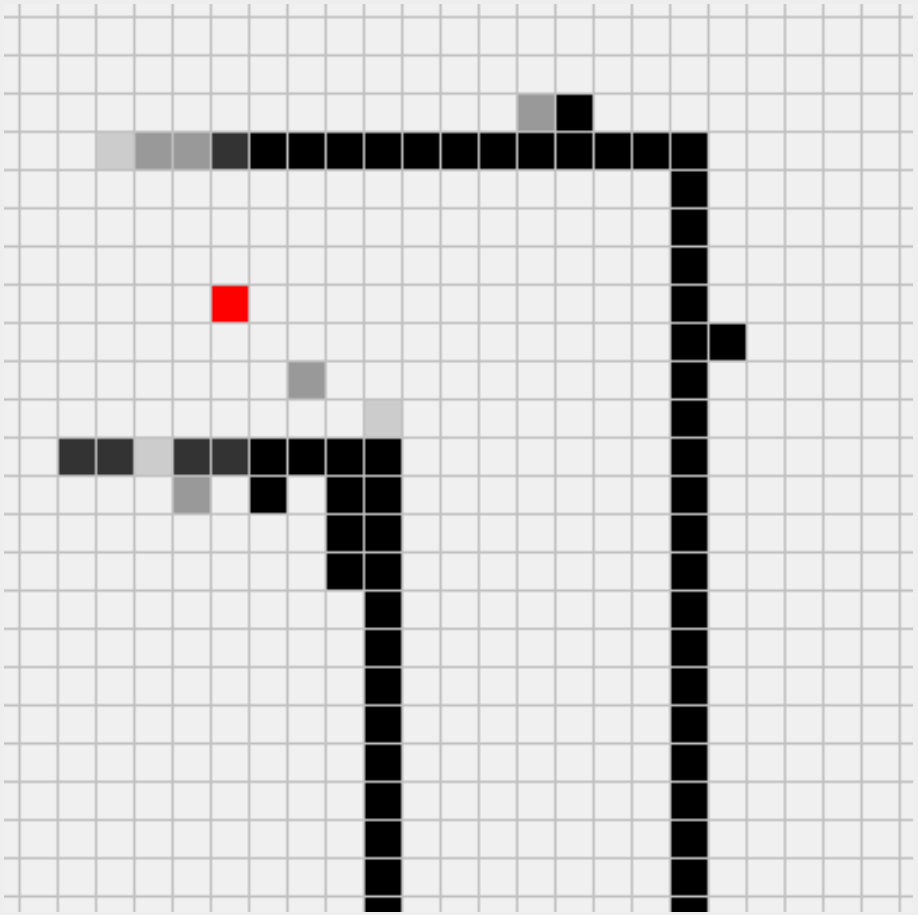
SLAM *(„Simultaneous Localization **and** Mapping“)*

- **1st characteristic question *(Mapping-Part):***
  → *What does the world around me look like?*
  → *Sense from various positions; integrate measurements to produce map; assume good knowledge of position*

- **2nd characteristic question *(Localization-Part):***
  → *Where am I?*
  → *relate sensor readings to a world model; compute location relative to this model; assume a good world model!*

Algorithms

SLAM *(„Simultaneous Localization **and** Mapping“)*



- *At each state: the robot (red) takes sensor-measurements*
- *The greyscale gives a likelihood for a cell to be either occupied (black) or empty (lightgrey)*
- *We might start with $P(X)=0.5$ for each cell in the beginning*
- ***GOAL**: Predict the right labels*

Algorithms

SLAM *(„Simultaneous Localization **and** Mapping“)*

ICP (Iterative-Closest-Points)-Algorithm

- Tries to minimize the difference between two point clouds
- Often used to reconstruct 2D or 3D surfaces from different scans

Finds the best matching maps!

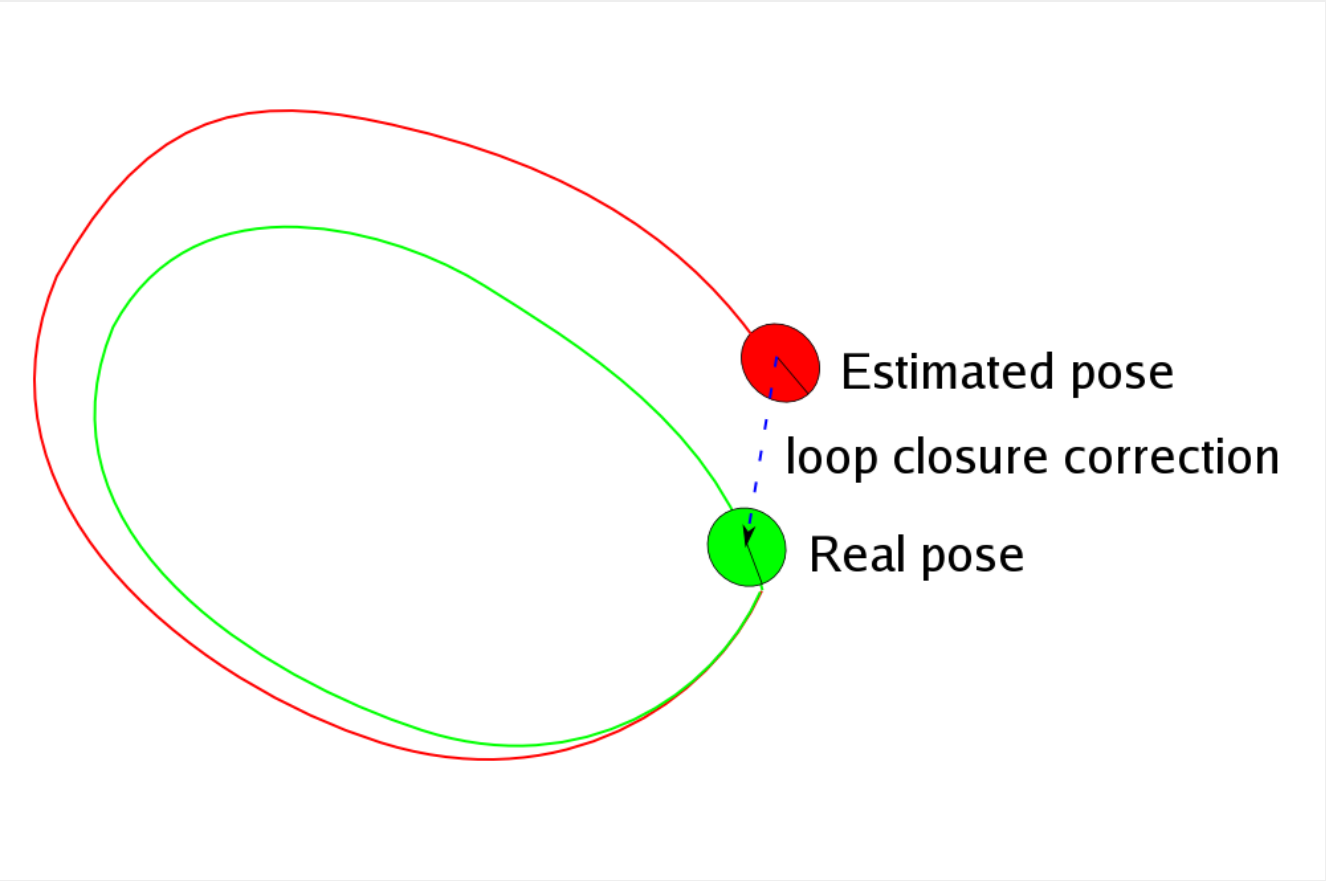
Loop-Closure (Topological Maps)

Imagine, the robot explored his environment and collected a lot of data by the paths he went → A question might occur:

Have I already been here before?

Loop-Closure connects paths that are very likely to belong together while trying to minimize the error of the generated map!

SLAM *(„Simultaneous Localization **and** Mapping“)*



Outlook

One last inspiring video of what is possible in research by using only a

- Laptop-Camera
- A Biologically Inspired SLAM System

https://www.youtube.com/watch?v=-0XSUi69Yvs

Outlook

Literature

- Robot Localization and Kalman Filters *(Rudy Negenborn, Thesis, Utrecht University, 2003)*
- Introduction to Autonomous Mobile Robots, R. Siegwart and I. Nourbakhsh, The MIT Press, Cambridge, Massachusetts 02142, ISBN: 0-262-19502-X, 2004
- Approaches to Mobile Robot Localization in Indoor Environments *(Patric Jensfelt, Thesis, Royal Institute of Technology, 2001)*
- Markov Localization for Mobile Robots in Dynamic Environments *(Fox, Burgard, Thrun, 1999)*
- Lokalisierung auf Basis kontinuierlicher und diskreter Drehwinkelveränderungen zwischen Landmarken in semistrukturierten Umgebungen *(Immo Colonius, Diplomarbeit, Universität Bremen, 2009)*
- *http://kogs-www.informatik.uni-hamburg.de/~neumann/HBD-SS-2008/KBSI-Lecture11.pdf*
- *http://users.isr.ist.utl.pt/~mir/cadeiras/robmovel/Markov-Localization.pdf*
- *http://forums.trossenrobotics.com/*
- *http://cogrob.ensta-paristech.fr/loopclosure.html*
- *http://www.cs.cmu.edu/~motionplanning/lecture/Chap8-Kalman-Mapping_howie.pdf*

End Of Presentation

Thank you for your attention!

Any Questions?