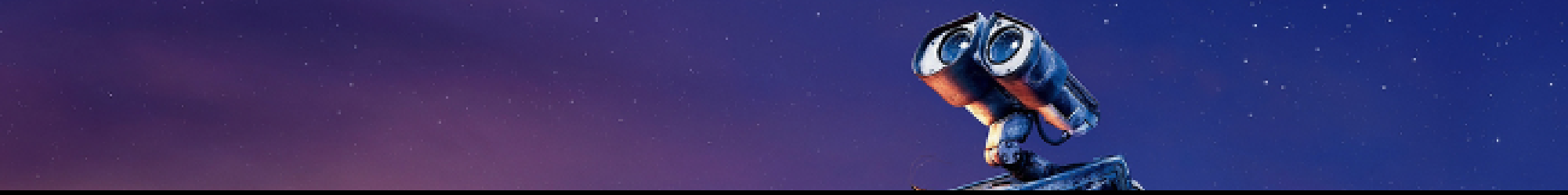


UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Navigation

RO51 - Introduction to Mobile Robotics

Zhi Yan

April 28, 2022

https://yzrobot.github.io/

www.utbm.fr

utbm
université de technologie
Belfort-Montbéliard

What?

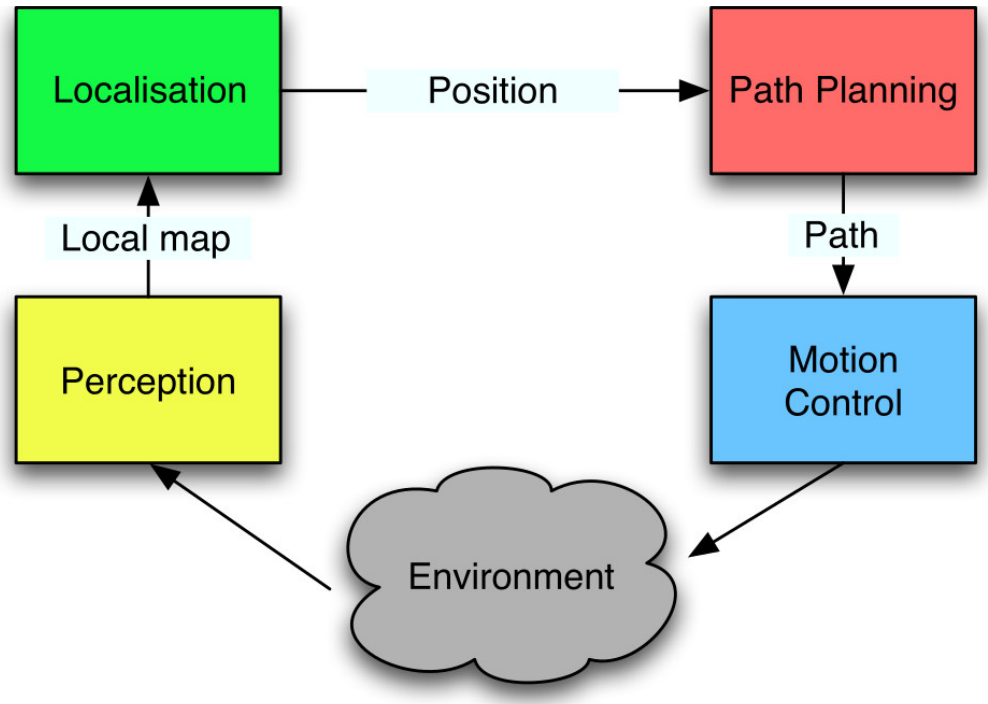
- Navigation is the skill or the process of **planning** a route for a mobile device and **taking** it there.
- Autonomous robot navigation can be defined as the combination of the four fundamental competences:
  - Perception (c.f. Lecture 4)
  - (self-) localization
  - Path planning (c.f. Lecture 7)
  - Motion control
- Navigation actually **holistically** answers three fundamental questions in mobile robotics:
  - Where am I?
  - Where am I going?
  - How to get there?

Why?

- Mobile (service) robots need to move autonomously in the working environment.

How?

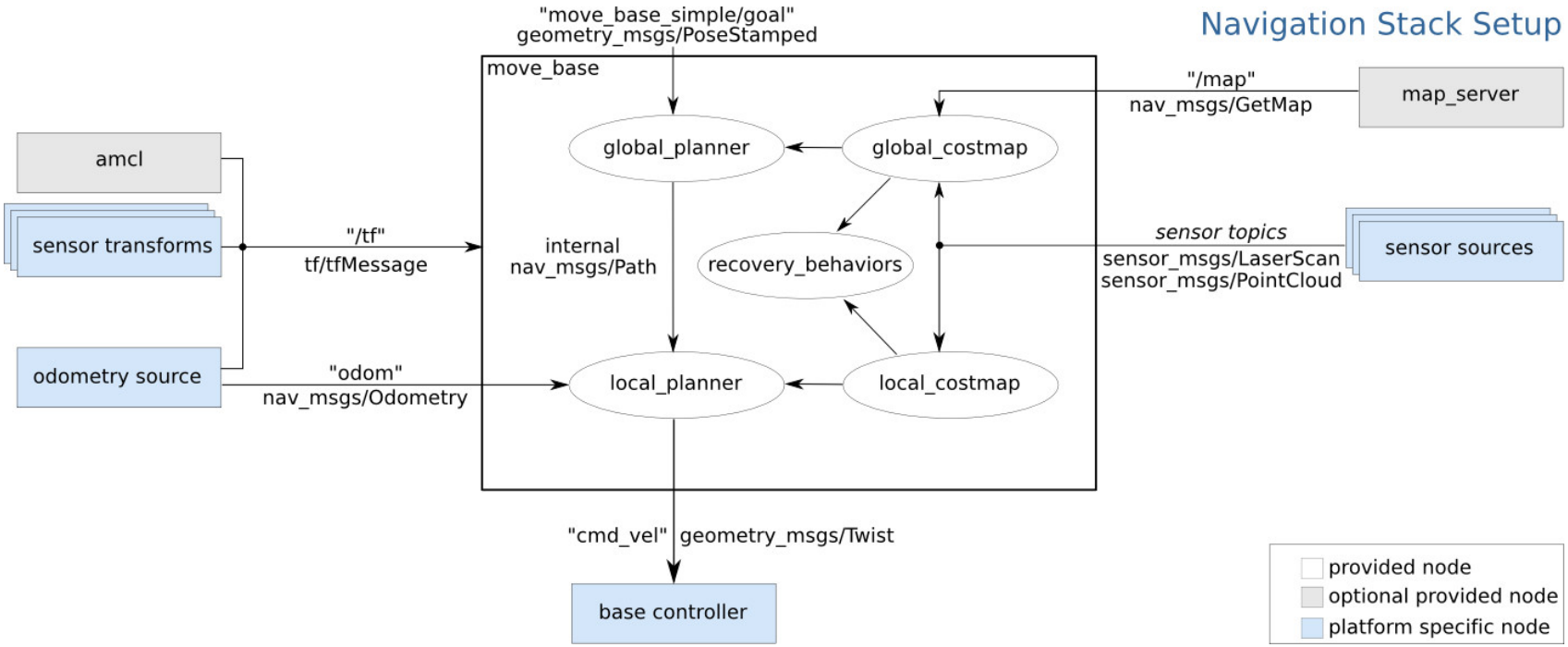
- Typical robot navigation system architecture:



- Navigation can be done with or without maps:
  - **With map**: a priori about the environment, technically mature
  - **Without map**: no structured priors, mainly relying on online reasoning, technology expected in the future

How?

- Robot navigation from a ROS perspective:



Localization

- The ability of a robot to determine its position in an environment. <= Where am I?
- The localization capability improves the "intelligence" of robot navigation.
  - When you are blindfolded, you will intuitively feel the decline in your "navigation performance".
  - There are robot navigation methods that do not require localization: wall-following (e.g. bug algorithm), heading toward the goal, etc.
- How the robot gets its position: someone helps it, or it calculates by itself:
  - Beacon systems (through communication/interaction): Global Navigation Satellite System (GNSS), Wi-Fi Positioning System (WPS), etc.
  - Self-estimation (through the measurement of the body and the outside world): wheel encoder, IMU, lidar, camera, etc.

Localization

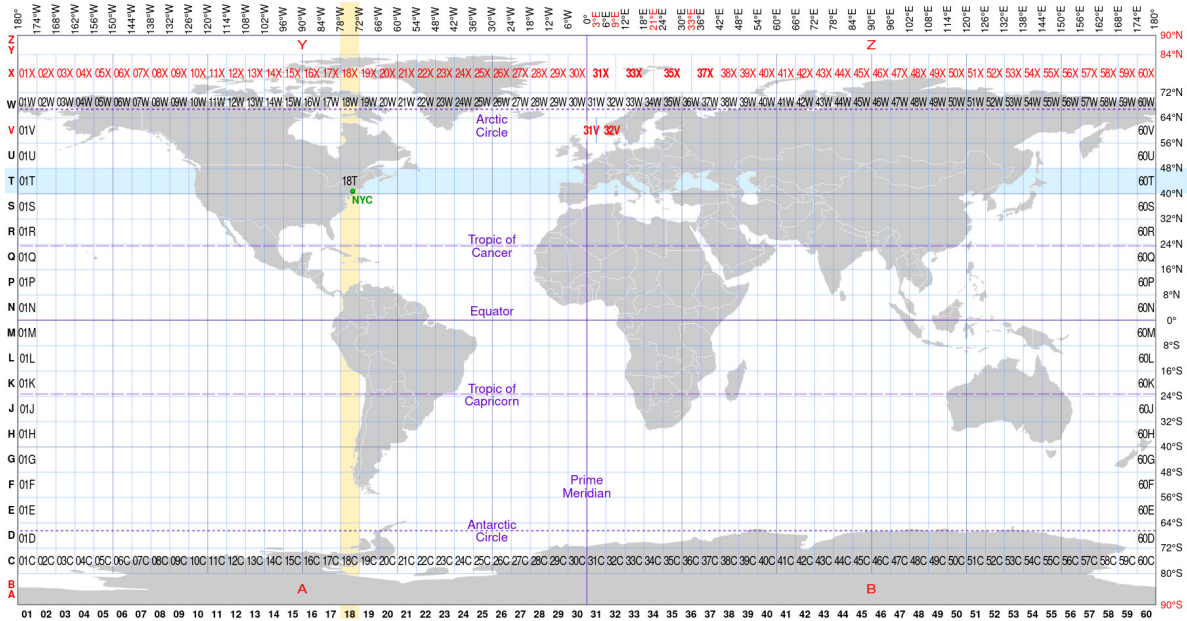
GNSS

- The basic principles have been covered in Lecture 4.
- Initially developed for military use, it has since expanded to civilian use.
- Well integrated into our daily life:
  - GNSS receivers are a standard module in modern smartphones.
  - Widely used for (self-driving) car navigation.
  - Navigation of other means of transport (airplanes, ships), precision agriculture, smart city, etc.
- According to its working principle, it is suitable for outdoor, not indoor.

Localization

GNSS for (self-driving) car navigation

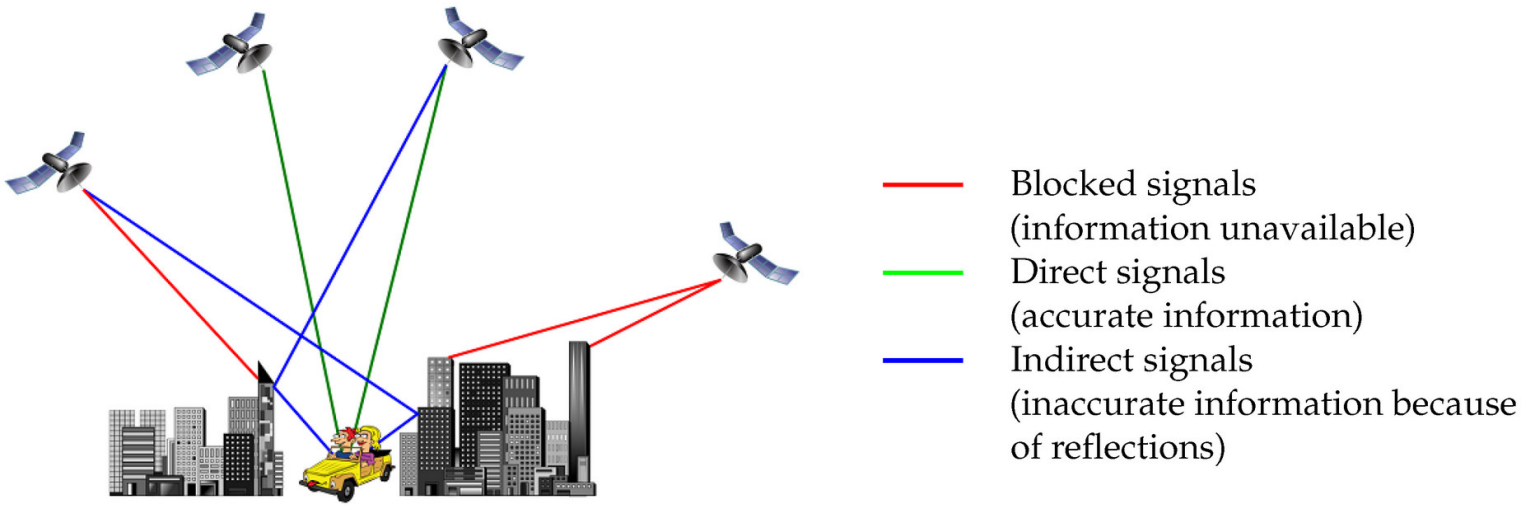
- Usually in conjunction with digital and/or high-definition maps.
- GNSS can tell the receiver's longitude, latitude and altitude (also time).
- Usually necessary to convert latitude-longitude readings into UTM (Universal Transverse Mercator) coordinates to match the map display.



Localization

GNSS for (self-driving) car navigation

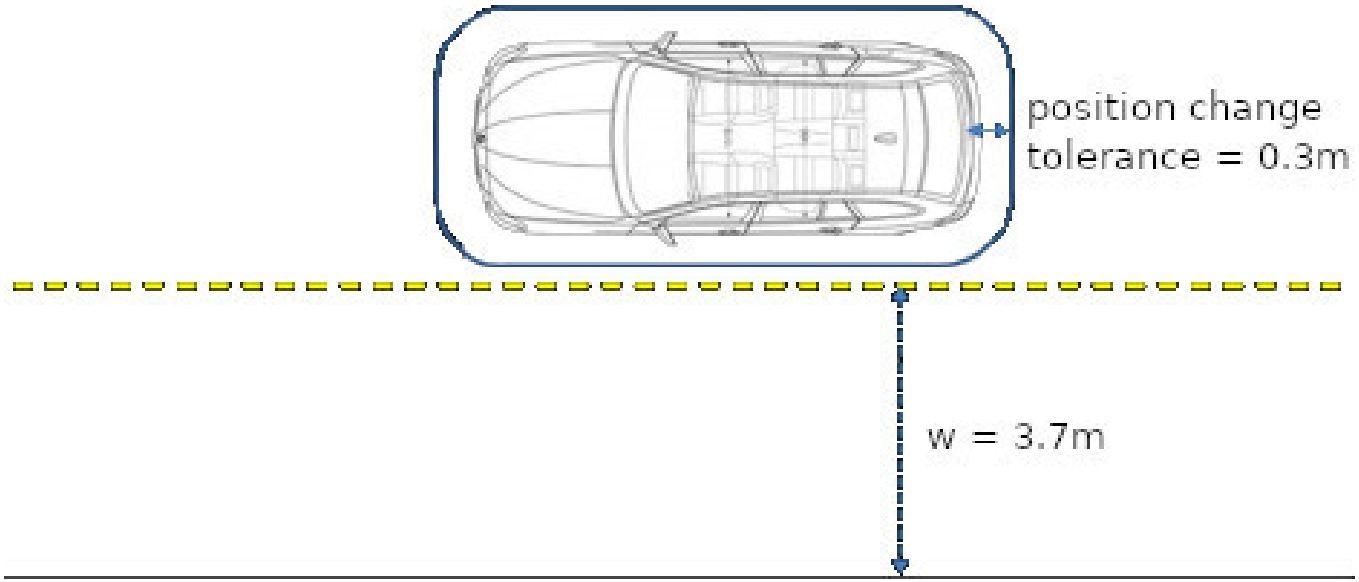
- Pros: mature technology, wide coverage (worldwide), low civilian price, easy operation, strong adaptability, etc.
- Cons:
  - Low positioning data update frequency (10 Hz).
  - Meter-level positioning accuracy (for self-driving cars, at least centimeter-level is required).
  - Signal occlusion and reflection problems in cities (especially urban canyons with tall buildings).



Localization

GNSS error sources

1. Satellite clock errors, ephemeris errors, ionospheric and tropospheric errors (delay), etc. <= For all users
2. Signal propagation errors (delay) that cannot be measured by the user or corrected by the calibration model.
3. Errors inherent in receivers such as internal noise, channel delay, multipath effects, etc.

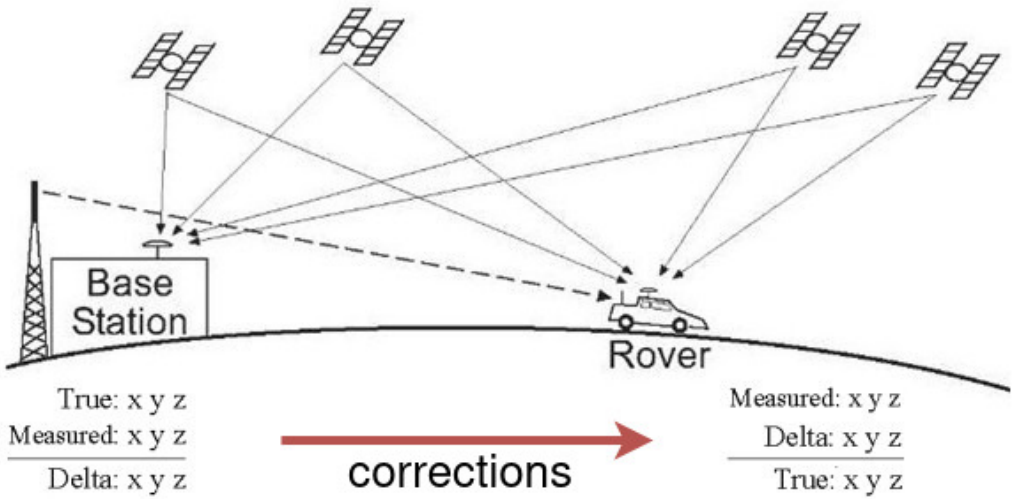


Localization

Ways to reduce errors

- **Differential GPS**: a network of fixed ground reference stations is used to broadcast the difference between the positions indicated by the satellites and the known fixed positions.
- **Real Time Kinematic (RTK)**: DGPS + carrier-phase enhancement (measurements of the phase of the signal's carrier wave).

=> Both can provide up to centimeter-level accuracy within 30 km.



Localization

IMU

- The basic principles have been covered in Lecture 4: using the accelerometer and gyroscope to infer the current position and orientation based on the position and orientation at the previous moment (i.e. dead reckoning).
- More specifically, according to Newton's laws of motion, by measuring the angular velocity and acceleration of the carrier in the inertial reference frame, integrating it with time, and transforming it into the navigation reference frame, the IMU can output the position and orientation of the carrier.
- The output frequency of IMU is high, generally 100Hz or even higher.
- IMU can provide accurate output for a short period of time, but errors will accumulate over time.

Localization

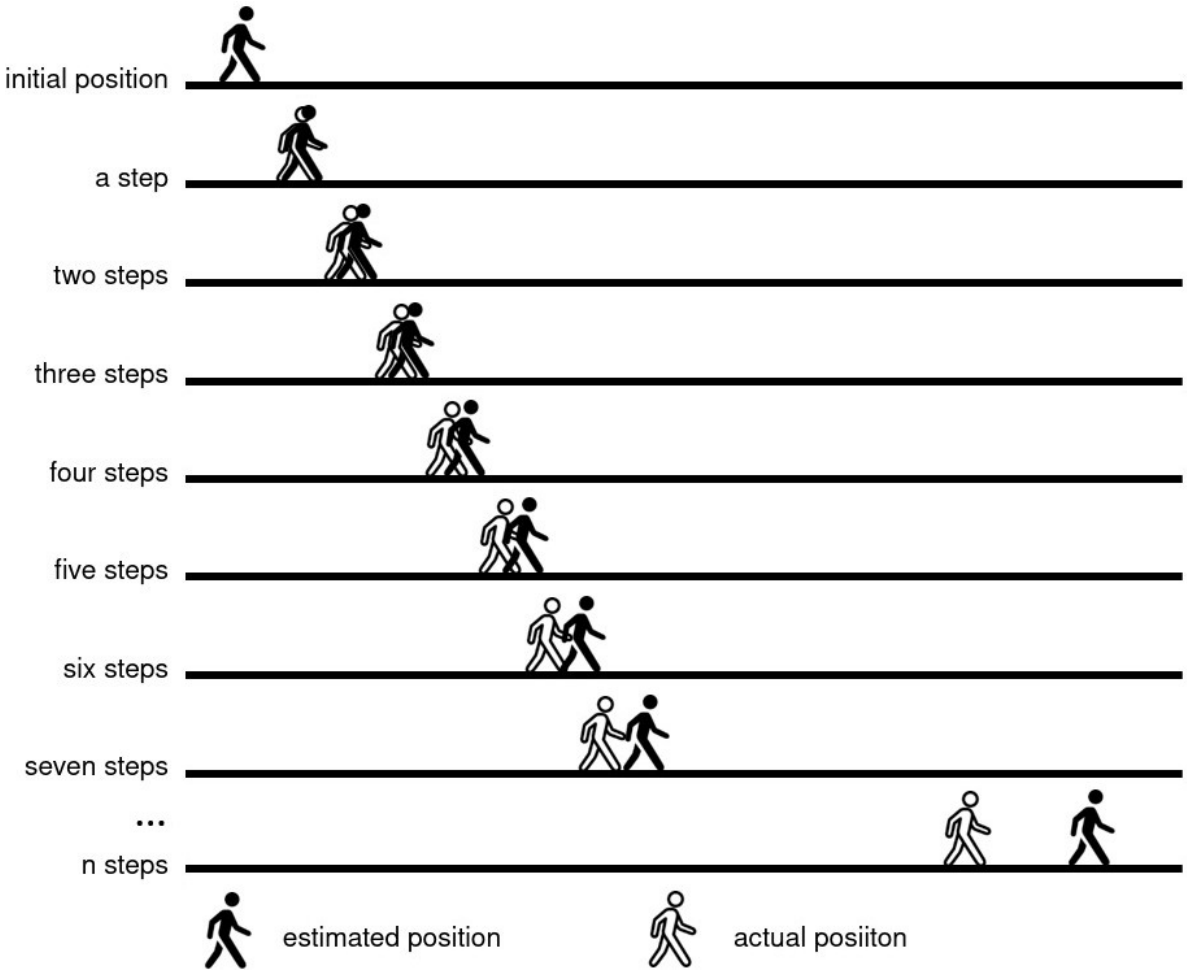
Self-estimation: IMU

- IMUs are usually used in conjunction with other sensors and have become one of the standard components of mobile robots.
- For self-driving cars (or outdoor robots in general), the IMU can work with GNSS to improve the positioning accuracy of the vehicle:
  - IMU can make up for the defect of the low update frequency of GNSS.
  - GNSS can correct the motion error of IMU (caused by accumulation).

Localization

Error accumulation of IMU

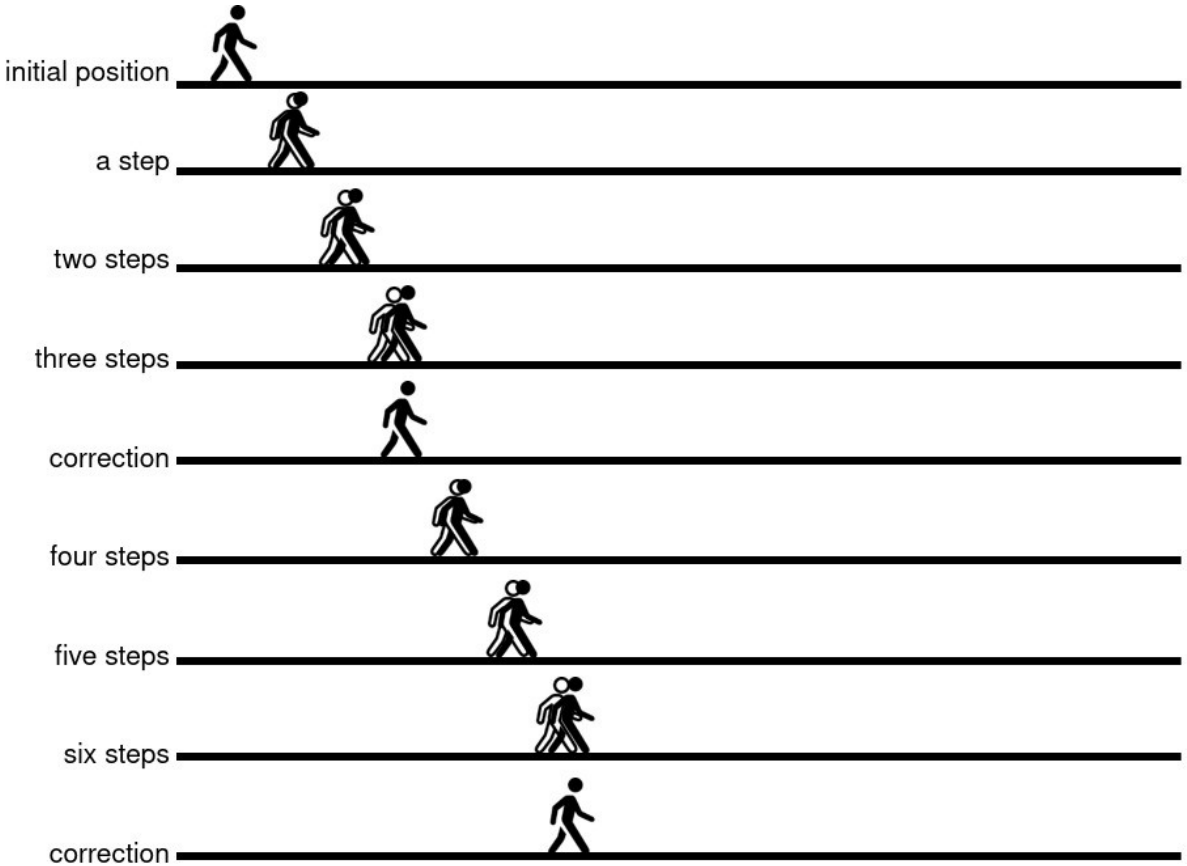
Take human walking in the dark as an example:



Localization

Multi-sensor fusion

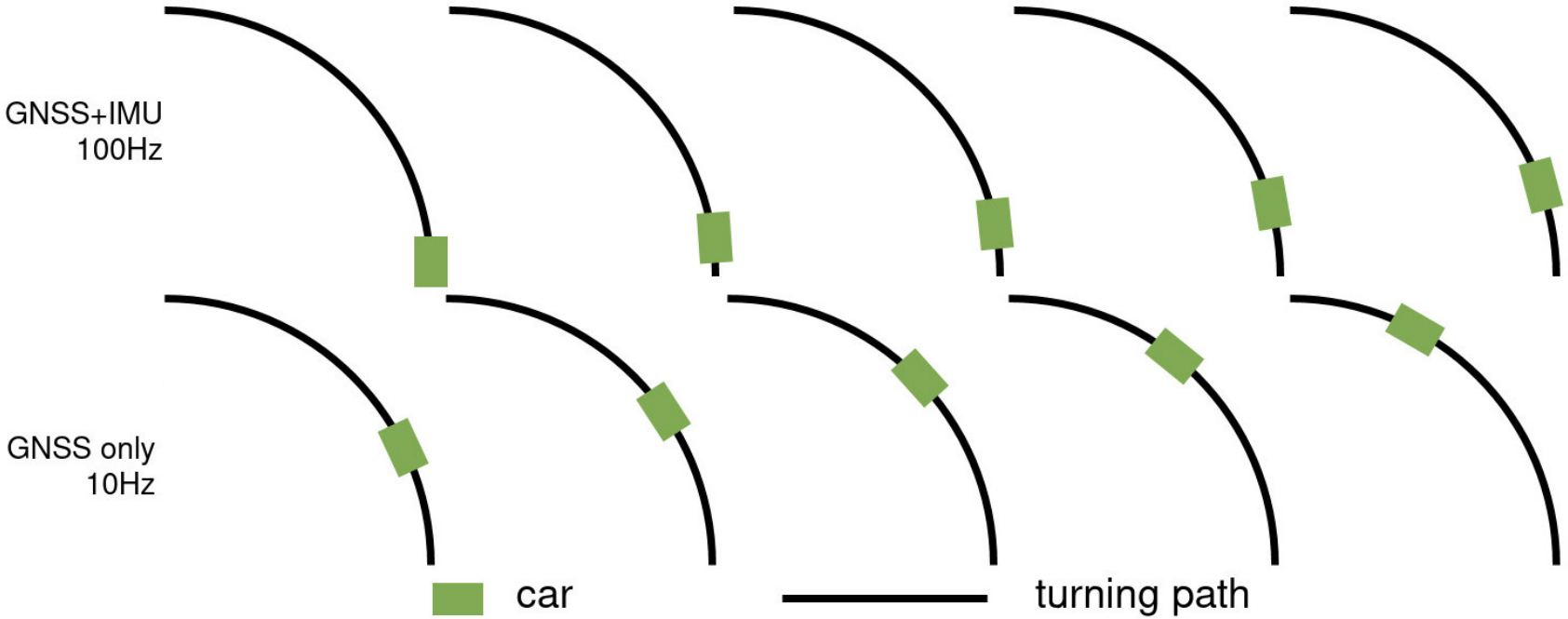
Additional sources of information (e.g. by touch) can improve a person's self-localization accuracy:



Localization

GNSS vs GNSS+IMU

Take self-driving cars as an example:



Localization

Self-estimation

- The robot estimates its position in the environment based on the observations (i.e. exteroception) and some known information (e.g. maps).
- A typical idea is for a robot to determine its position in the environment by matching the perceived local environment information with the environment in a known map.
- (accurate) Self-estimation is challenging, not only because of the noise/error from the sensors but also from the actuators.
- Another problem that affects self-localization is known as **sensor aliasing**: e.g. environments with the same appearance in different locations result in very similar sensor readings (i.e. lack of features).
- Proprioceptive sensors (e.g. wheel encoder, IMU) can help!

Localization

Self-estimation

At the operational level, the following components may need to be considered:

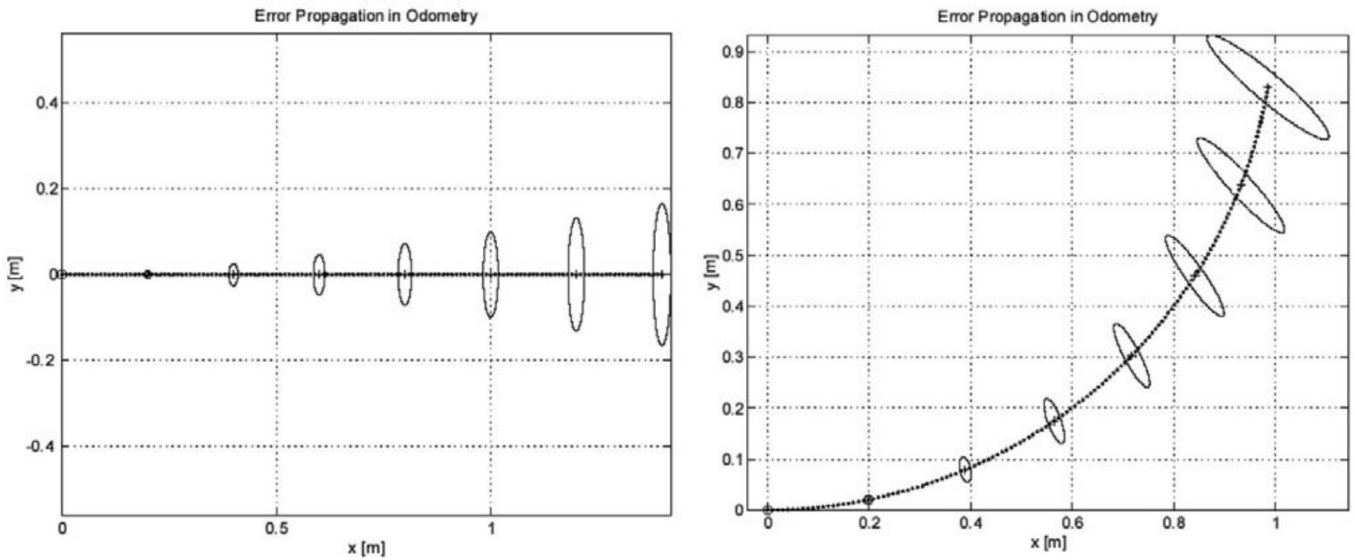
- Models of the deployed robot, including its geometric model, kinematic model, noise model, etc.
- Models about the work environment, such as maps.
- Methods for environmental feature extraction (via sensor readings).
- Strategies for matching features to environment models.
- Methods for updating robot pose estimation.

Localization

Self-estimation

To gain a further understanding of actuator noise and error accumulation, let's look at odometric position estimation over time (from a known position by integrating the movement) for a differential-drive robot:

- Left: growth of the pose uncertainty for linear movement.
- Right: growth of the pose uncertainty for circular movement.

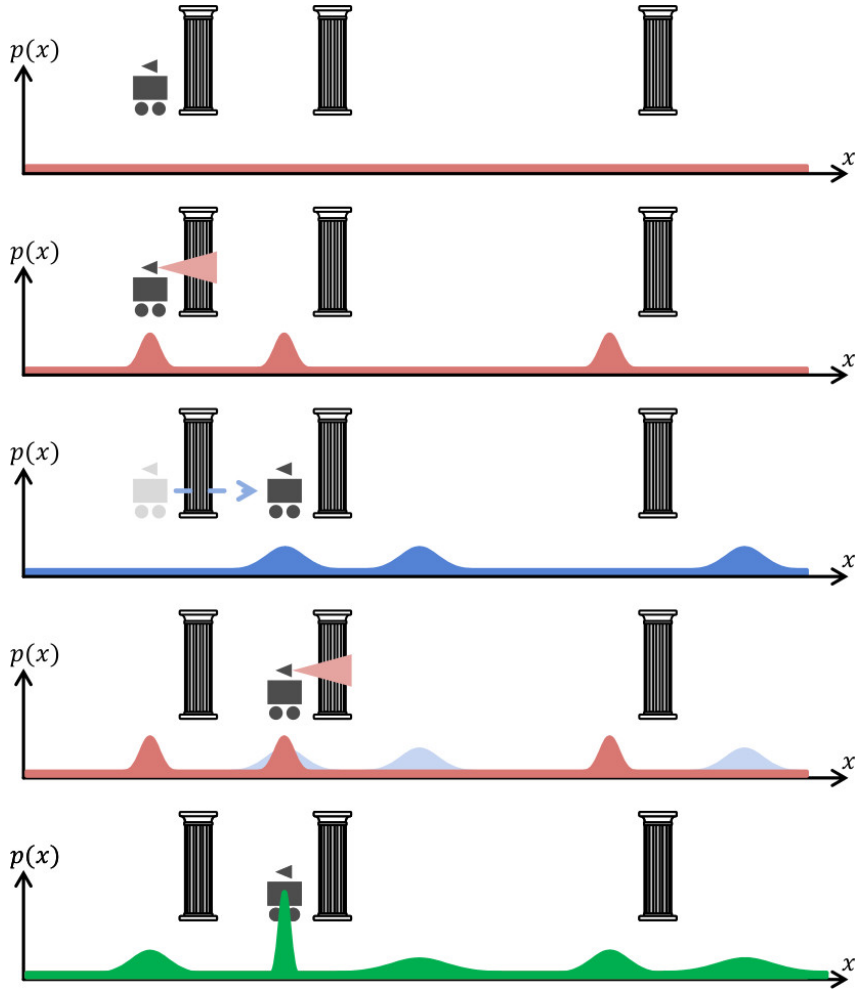


Localization

Self-estimation

"act-see" cycle for localization (by R. Siegwart *et al.* @ ETH Zurich)

- A robot is in an environment (a simple 1D example)
- "see": the robot queries its sensors => finds itself next to a pillar
- "act": robot moves 1m forward
  - motion estimated by wheel encoders
  - accumulation of uncertainty
- "see": the robot queries its sensors again => finds itself next to a pillar
- Belief update (information fusion with uncertainty models)

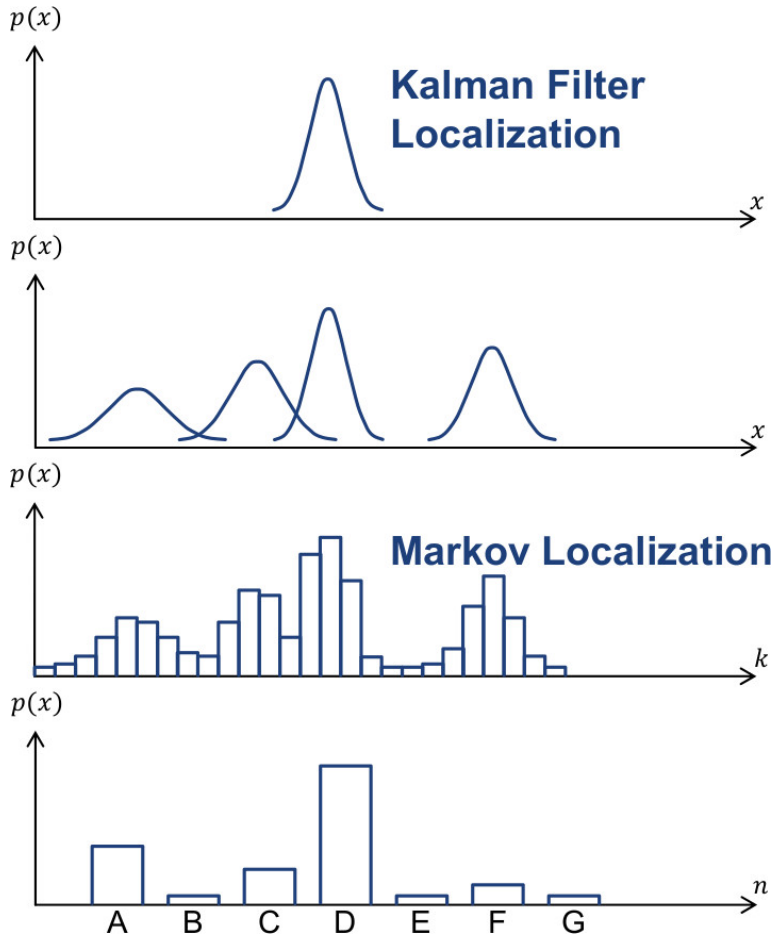


Localization

Self-estimation

Belief representation (uncertainty modeling):

- **Continuous map** with single hypothesis probability distribution $p(x)$ => powerless against uncertainty
- **Continuous map** with multiple hypotheses probability distribution $p(x)$
- **Discretized metric map** (grid k) with probability distribution $p(k)$
- **Discretized topological map** (nodes n) with probability distribution $p(n)$



Localization

Self-estimation based on probabilistic robotics

- Still remember Bayes' Theorem?

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

- This theorem is used by both **Kalman-filter** (continuous pose representation and Gaussian error model) and **Markov** (discretized pose representation) localization algorithms during the measurement update.

Localization

Self-estimation based on probabilistic robotics

- "**see**": probabilistic estimation of the robot's new belief state $bel(x_t)$ as a function of its measurement data z_t and its former belief state $\overline{bel}(x_t)$:

$$bel(x_t) = \eta P(z_t \mid x_t, M)\overline{bel}(x_t)$$

where $P(z_t \mid x_t, M)$ is the probabilistic measurement model (i.e. perception) that is, the probability of observing the measurement data z_t given the knowledge of the map M and the robot's position x_t. Thereby $\eta = P(y)^{-1}$ is the normalization factor so that $\Sigma P = 1$ (or $\int P = 1$ for continuous case).

Localization

Self-estimation based on probabilistic robotics

- Still remember the law of total probability?

$$\textbf{Continuous case}: P(A) = \int_{-\infty}^{\infty} P(A \mid X = x) f_X(x) dx$$

$$\textbf{Discrete case}: P(A) = \sum_n P(A \mid B_n) P(B_n)$$

- This law is used by both **Kalman-filter** and **Markov** localization algorithms during the prediction update.

Localization

Self-estimation based on probabilistic robotics

- "**act**": probabilistic estimation of the robot's new belief state $\overline{bel}(x_t)$ based on the previous location $bel(x_{t-1})$ and the probabilistic motion model $p(x_t \mid u_t, x_{t-1})$ with action u_t (i.e. control):

$$\textbf{Continuous case}: \overline{bel}(x_t) = \int P(x_t \mid u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

$$\textbf{Discrete case}: \overline{bel}(x_t) = \sum_{x_{t-1}} P(x_t \mid u_t, x_{t-1}) bel(x_{t-1})$$

Localization

Kalman Filter vs. Markov localization

Kalman filter localization:

- **Pros**: can track the robot from an initially known position and is inherently both precise and efficient.
- **Cons**: may fail to manage a large number of robot pose hypotheses (due to growing uncertainty) and end up getting lost irretrievably.

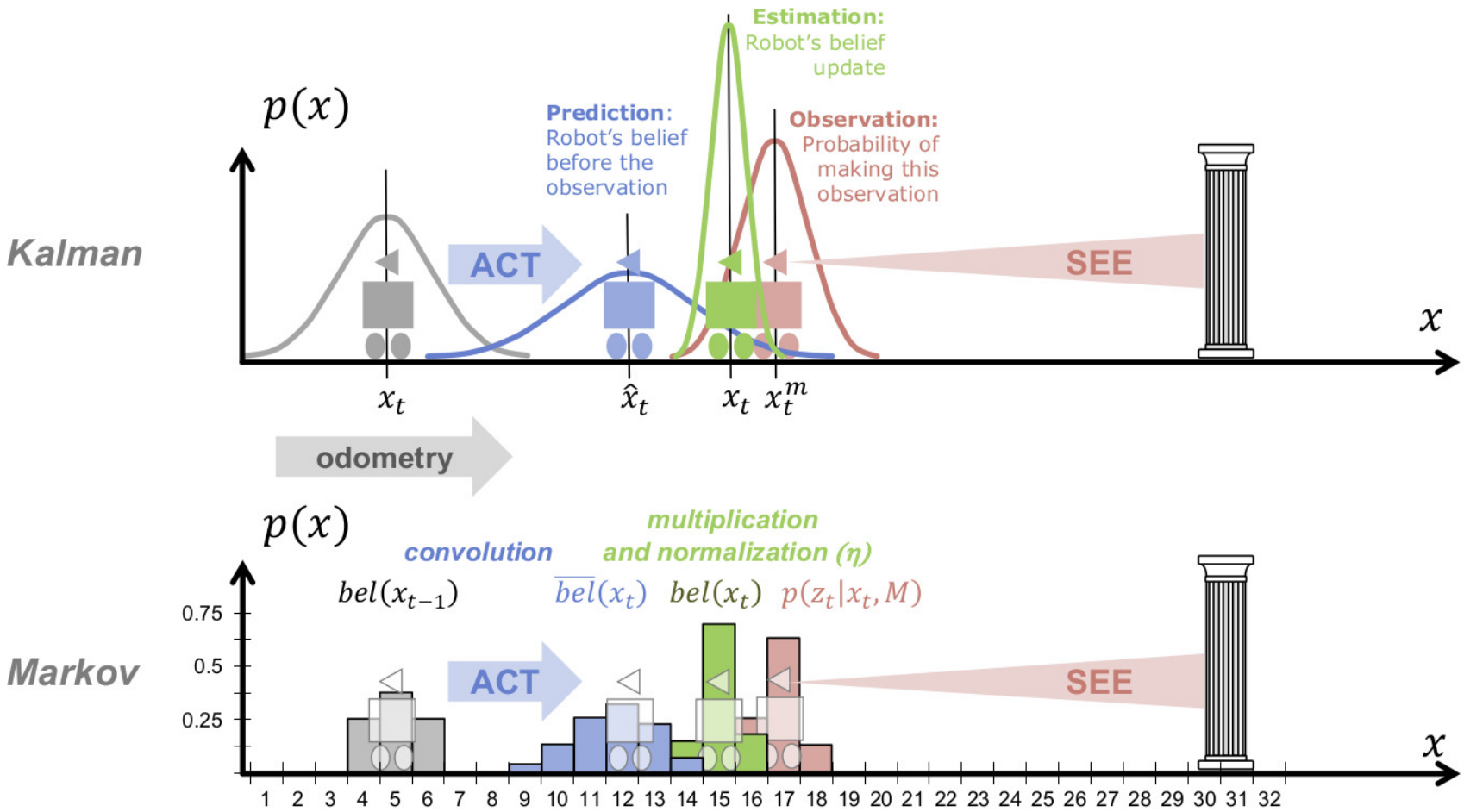
Markov localization:

- **Pros**: can track the robot from any unknown position and can recover from ambiguous situations.
- **Cons**: update the probabilities of all positions within the entire state space requires a discrete representation of the space, e.g. occupancy grid map, involving computational efficiency issues.

Localization

Kalman Filter vs. Markov localization

"act-see" cycle for localization (by R. Siegwart et al. @ ETH Zurich)

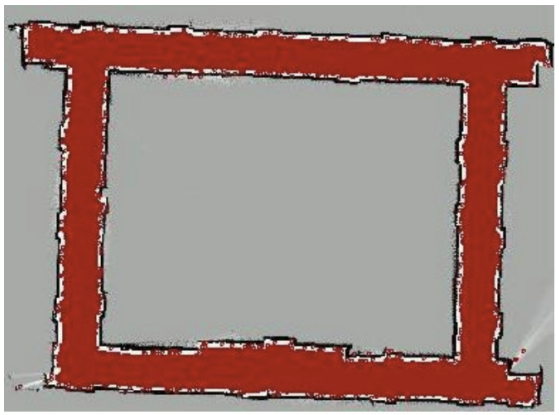


Localization

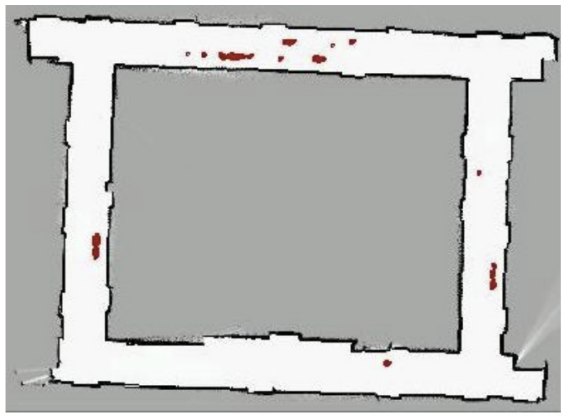
Monte Carlo localization (MCL)

Let's take a closer look at a Markov-based approach, i.e. **Monte Carlo localization**[1]:

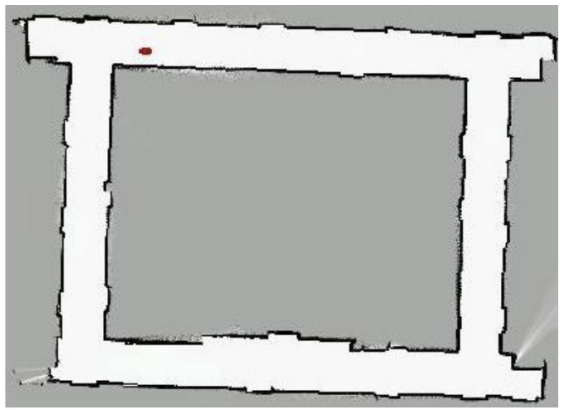
- Probability estimation based on particle filter (a sampling-based method).
- Computationally efficient, no need to compromise into coarse-grained localization.



t = 1 sec, approx 100,000 particles



t = 40 sec, approx 1,000 particles



t = 80 sec, approx 100 particles

[1] ROS implementation: `http://wiki.ros.org/amcl`

Localization

Monte Carlo localization (MCL)

1. The algorithm typically starts with a uniform random distribution of particles over the configuration space, meaning the robot has no information about where it is and assumes it is equally likely to be at any point in space.
2. Whenever the robot moves, it shifts the particles to predict its new state after the movement.
3. Whenever the robot senses something, the particles are resampled based on recursive Bayesian estimation, i.e., how well the actual sensed data correlate with the predicted state.
4. Ultimately, the particles should converge towards the actual position of the robot.

Localization

Monte Carlo localization (MCL)

- Randomly generate M particles
- Estimate the pose x_t of each particle based on x_{t-1} according to the motion model of the robot.
- The weight of each particle is calculated based on the sensor readings.
- Update the state of each particle.
- Resampling to get M particles from all particles according to the new weight value.

1: **Algorithm MCL**$(\mathcal{X}_{t-1}, u_t, z_t, m)$**:**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Localization

Adaptive Monte Carlo localization (AMCL)

Adaptive:

- Dynamically adjust the number of particles in the filter:
  - When the robot's pose is highly uncertain, the number of particles is increased.
  - When the robot's pose is well determined, the number of particles is decreased.
- A trade-off between processing speed and localization accuracy.

Algorithm Augmented_MCL$(\mathcal{X}_{t-1}, u_t, z_t, m)$**:**

static $w_{\text{slow}},\ w_{\text{fast}}$
$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
for $m = 1$ to M do
  $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
  $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
  $w_{\text{avg}} = w_{\text{avg}} + \frac{1}{M}\, w_t^{[m]}$
endfor
$w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$
$w_{\text{fast}} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$
for $m = 1$ to M do
  with probability $\max(0.0,\ 1.0 - w_{\text{fast}}/w_{\text{slow}})$ do
    add random pose to $\mathcal{X}_t$
  else
    draw $i \in \{1, \ldots, N\}$ with probability $\propto w_t^{[i]}$
    add $x_t^{[i]}$ to $\mathcal{X}_t$
  endwith
endfor
return $\mathcal{X}_t$

Summary

- Navigation for outdoor vehicles (e.g. self-driving cars).
- Typical localization methods in mobile robotics: self-estimation based on probability theory.
- Monte Carlo localization (MCL).
- Further reading: error modeling for odometric position estimation, Kalman-filter localization.

The end

Thank you for your attention!

Any questions?