Sistemi Intelligenti Avanzati
Corso di Laurea in Informatica, A.A. 2021-2022
Università degli Studi di Milano

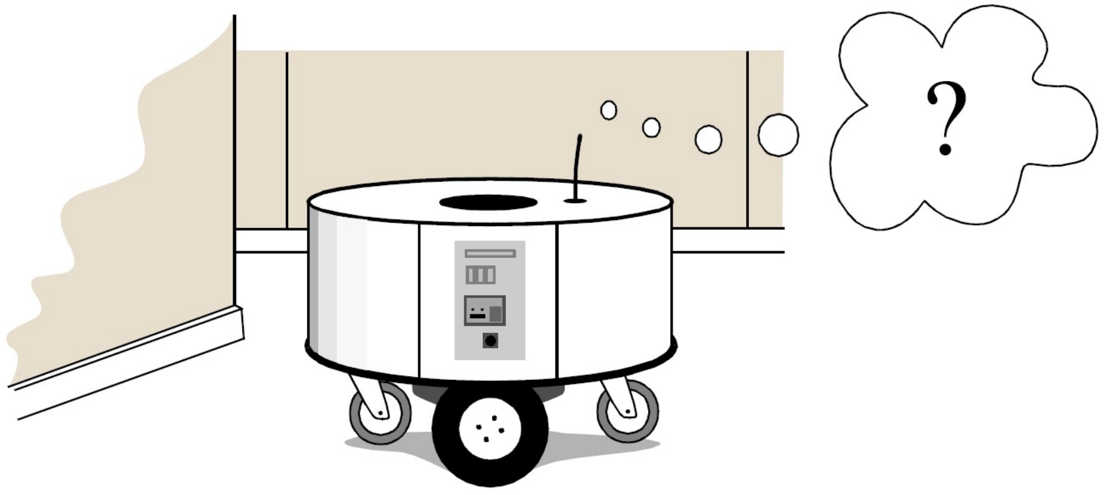
Introduction to Autonomous Mobile Robotics #2

Matteo Luperto
Dipartimento di Informatica
matteo.luperto@unimi.it

Outline

Overview of core concepts:

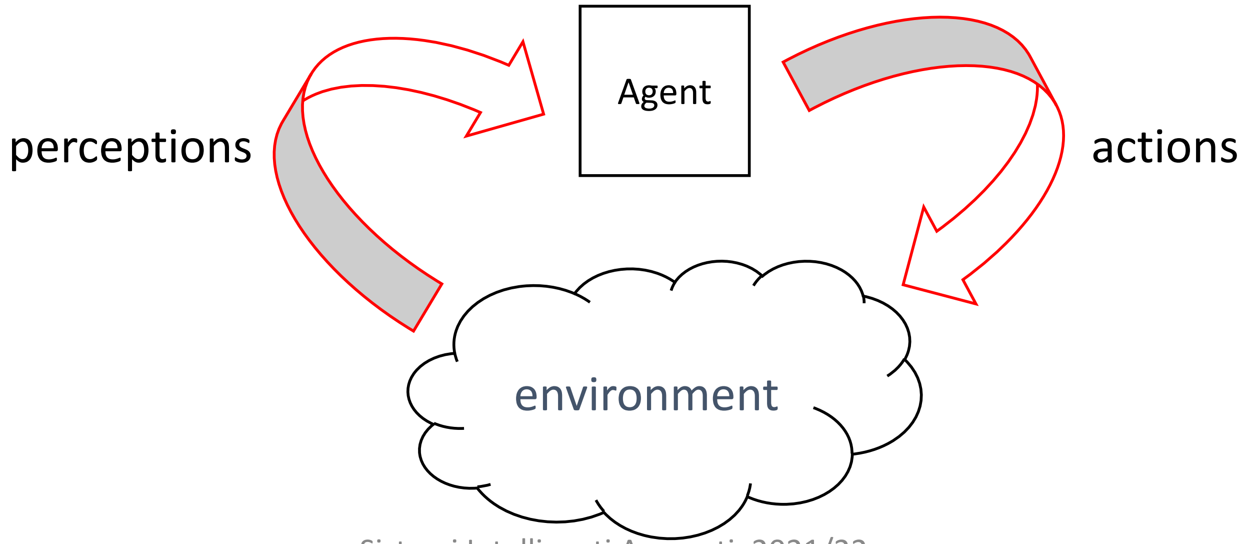
- Robot Motion
- <u>Perception</u>
- Localization and Mapping
- Navigation



Assumption: let's talk about the simplest type of mobile robots, wheeled ground vehicles

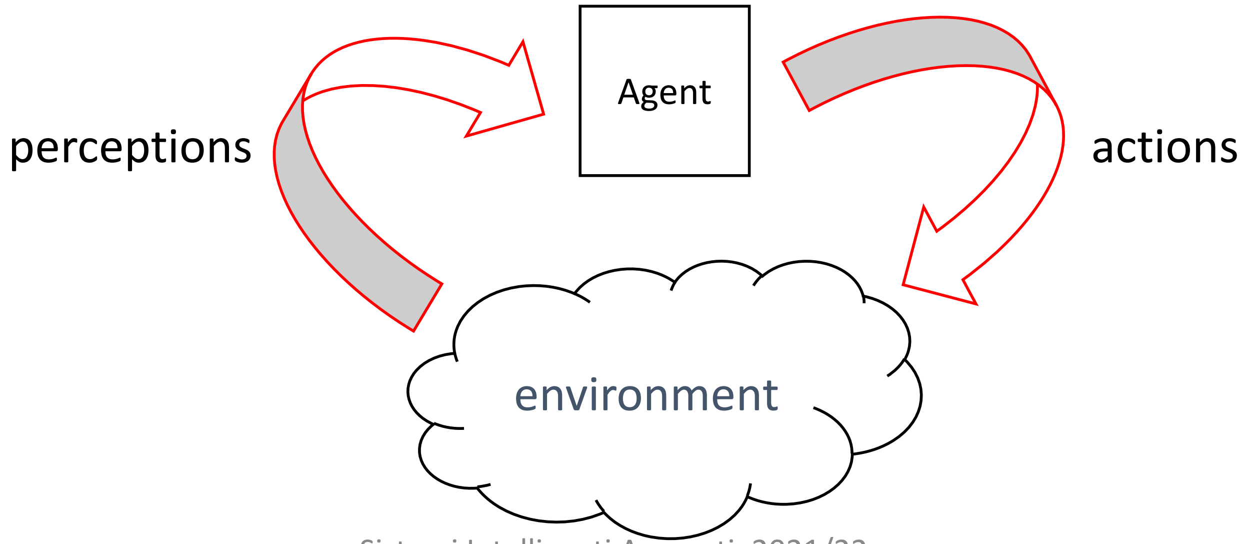
Perception: sensor types

- Proprioceptive
  - Sensor measure values internal to the system as motor speed, wheel load, robot arm joint angles, battery voltage
- Exteroceptive
  - Sensors acquire information from the robot's environment as distance measurement, light intensity, sound amplitude = meaningful environmental features



Perception: sensor types

- Passive sensors
  - Measure ambient environmental energy entering the sensors, as microphones, temperature probes, cameras
- Active sensors
  - Emit energy into the environment, then measure the environmental reaction. More control, more accuracy, but interference issues (and sometimes power)



Perception: sensor types

What to measure? What is the robot task?

- Vision
- Obstacle distance
- Position
- Environmental monitoring (ASV)
- Olfaction (e.g. inspection of chemical plants)
- Temperature (e.g. inspection of a server farm)

The most important sensors are those involved in the robots' mobility

Sensors performance characterization

- Dynamic Range
  ratio between maximum and minimum input value – usually in dB
- Resolution
  minimum difference between two values that can be detected
- Linearity
  how the sensor respond to changing inputs
- Bandwidth or Frequency
  speed with which a sensor can provide a stream of readings
  number of measurements per seconds (in Hz)

These specs of the sensors are usually measured in labs – controlled environments;
however, often we need to identify how the sensor performs in its real-world deployment

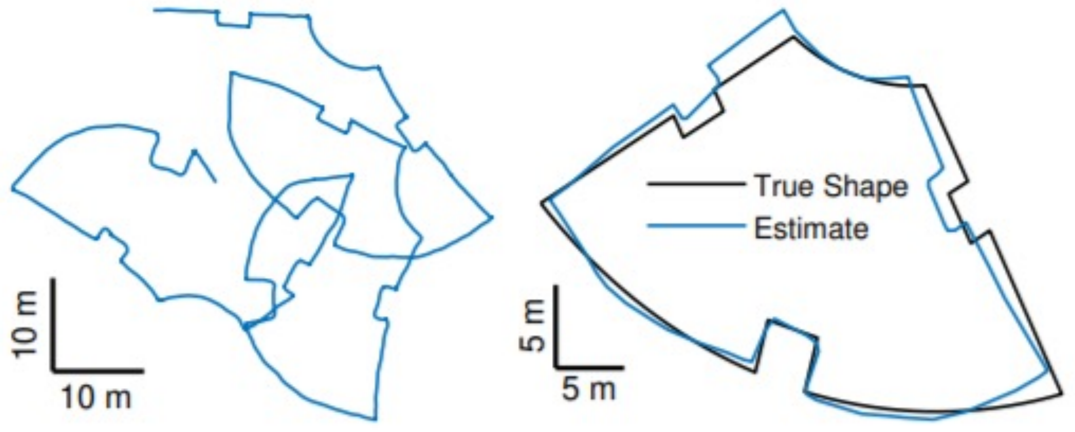
In Situ Sensors performance characterization

- Sensitivity:
  measures how incremental change in the target input changes the output signal
- Cross Sensitivity
  sensitivity to environmental external parameters that are orthogonal to the target parameter; high cross-sensitivity is task-related and unwanted
- Error
  difference between output and true value
- Accuracy
  degree of conformity between sensor’s measurement and true value (usually %)
- Systematic Error
  errors caused by factors that, theoretically, can be modeled; deterministic; example: calibration errors, slopes, …
- Random Error
  errors that cannot be predicted using a model nor can be mitigated; modeled as probabilistic process (stochastically)
- Precision
  not to be confused with accuracy; reproducibility of the results: if the phenomena is the same, the measured value should be the same (this holds if I use several different sensors of the same type: I expect the same results from all of them)

Challenges in Sensors modeling

- Blurring of systematic and random errors
  active ranging sensors tend to have failures that are triggered by specific relative position of the sensor and of the environment (e.g., glass surfaces, mirrors, …)
  During motion this happens at stochastic intervals
  Moreover, robot usually have different and concurrent sensors
  This, combined, is used to model error and to smooth their impact wrt the robot activity

(a) The courtyard of our Institute. We used the inner lawn area for testing the proposed mapping method.



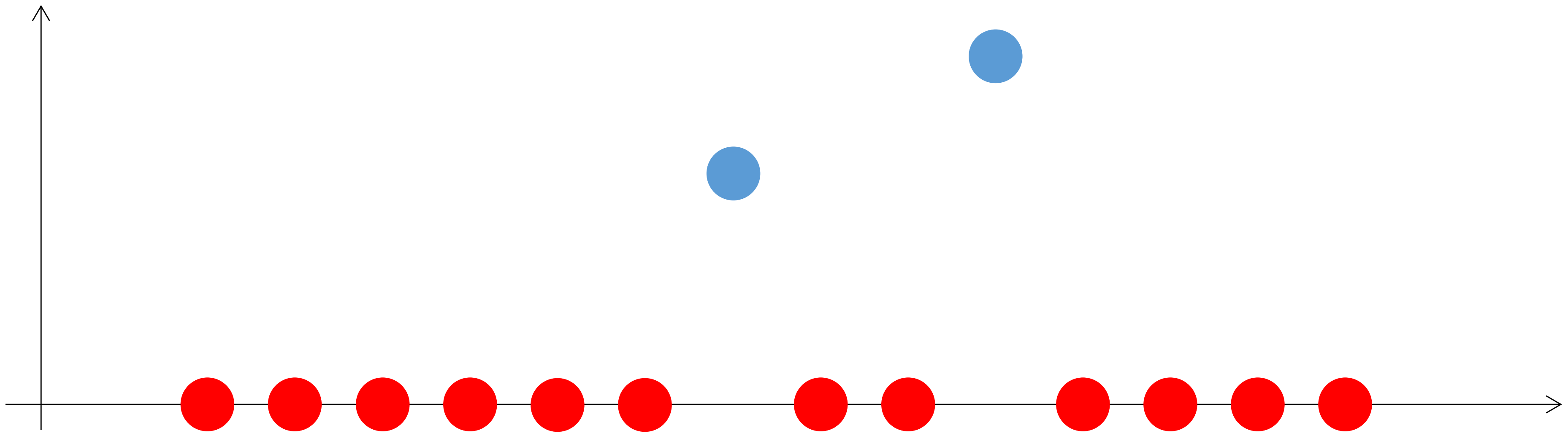
(b) The left panel shows the estimated path of the robot generated from its wheel odometry and the right panel the estimated map and the true shape of the test environment.

Fig. 9. The real courtyard depicted in (a) and the collected odometry data together with the map estimate shown in (b).

Example: [Rottman et al, ECMR 2019]

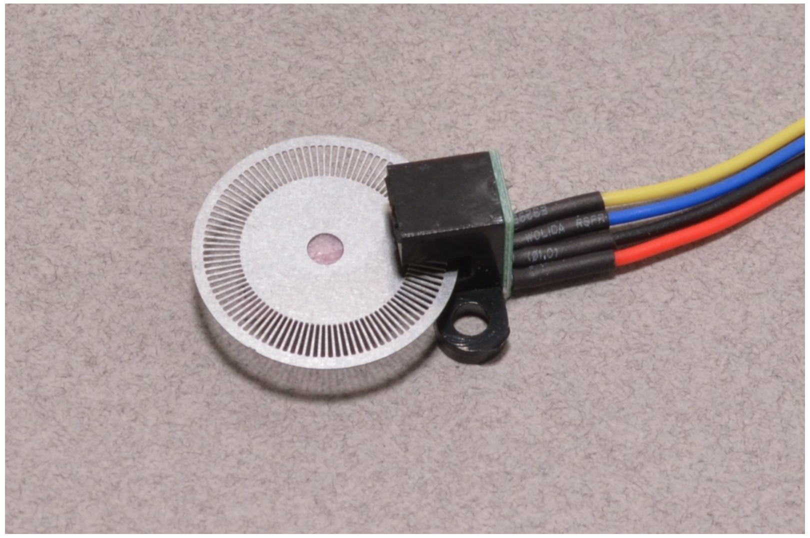
Challenges in Sensors modeling

- Multimodal error distribution
  a common choice is to characterize the behavior of a sensor’s random error in terms of a probability distribution over various output values; diverging from the model can help to detect errors (measuring the correct value is most probable)



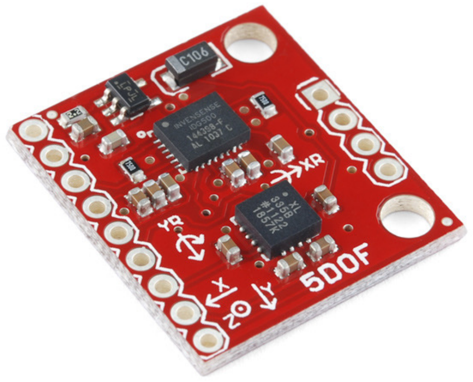
Wheel/Motor sensors

- Proprioceptive sensors used to measure the internal state and dynamics of the robot
- Optical encoders: measure the angular speed and position within a motor drive, or shaft of a wheel or steering mechanism
- Used for localization and to estimate the robot movements
- While the sensor itself could be accurate, the measure is inherently inaccurate (odometry) and needs integration (it measures the motor itself, what if a wheel slips? or if there is a slope?)



Heading Sensors

- Compasses
  - outdoor
- Ground-based-beacons
  - **GPS** is a good choice for outdoor robots, but performs poorly indoor
  - For Indoor robots, we can have similar solutions indoor that usually requires other complex sensing capabilities (vision) or detection of NFC or RFID tags installed in the environment
  - We require a resolution of a few centimeters
- Gyroscope
  - Usually combined with accelerometers in an IMU Inertial Measurements Unit



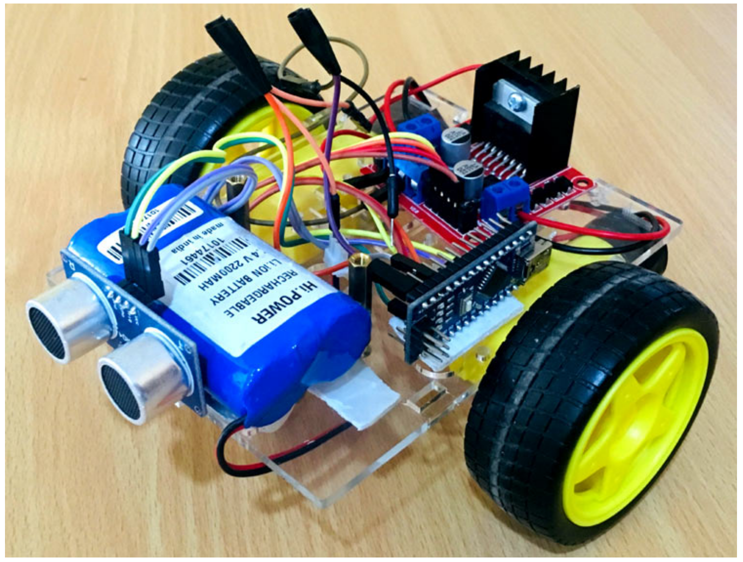
Active ranging

- Most popular sensors in mobile robotics
- Usually have a low price point and easily interpreted outputs
- Among them, *time-of-flight* sensors are those commonly used
  - $d = c \cdot t$
  - d distance travelled
  - c speed of wave propagation
  - t time of flight

1. Sonars
2. Laser Range Finder

Ultrasonic Sensors - Sonars

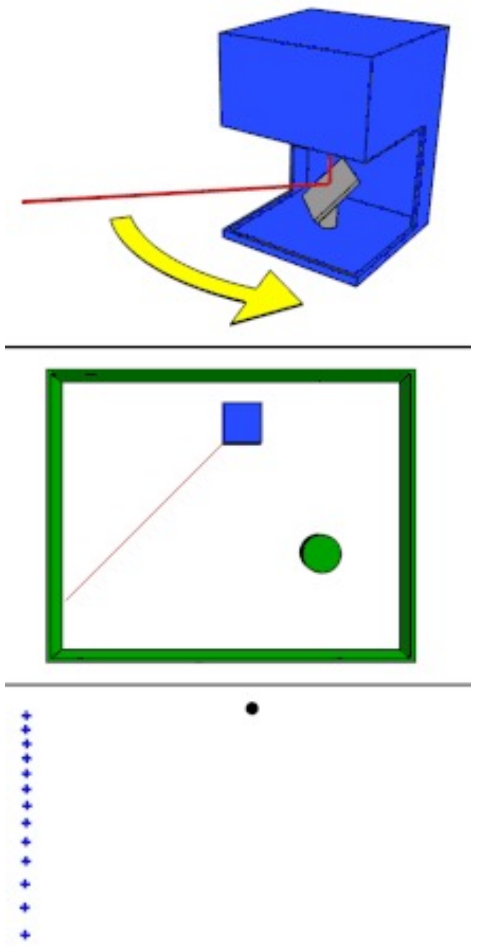
- Cheap
- Not particularly accurate
- Simple and interpretable measurements
- Good for proximity – obstacle avoidance
- Low range



Laser range finders - Lidars

Time of Flight (ToF) sensor which is used to *scan* the surrounding of the robot. Parameters:

- Range = max perceivable distance (1-100m)
- Field of View (FOV) = degrees of a scan, from 180° to 270°, 360°
- Angular resolution = how many points for each degree in a scan
- Frequency = how many scan per second (1hz-50hz)
- 2D or 3D

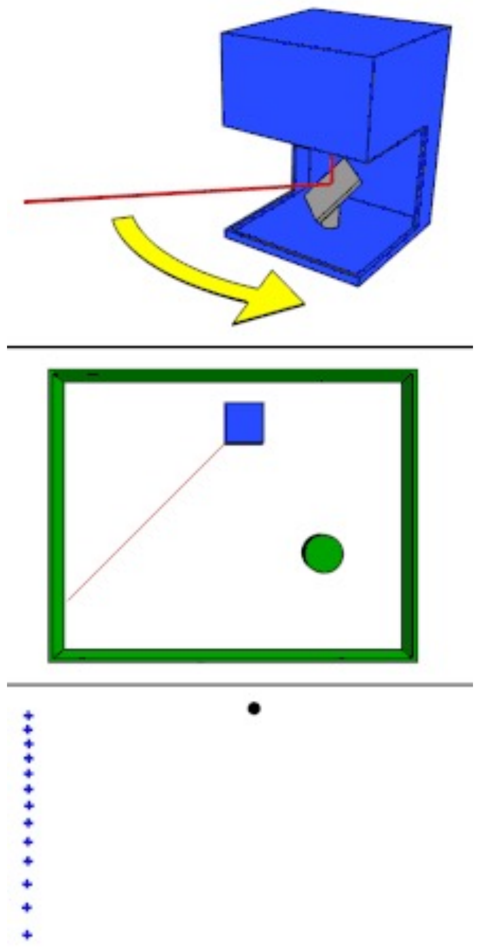


Laser range finders - Lidars

Widely used in most indoor and outdoor robot applications as they:

- Are relatively cheap
- Easy to use and provide interpretable measures
- Robust wrt environmental changes (e.g. day, night, different seasons)

Laser range scanner are the most important sensor for most autonomous mobile robots



Laser range finders - Lidars

Different tasks – different environment – different lidar types:

Indoor lidars have a range from 3-5m to 10-20m, with a FOV of 180-270°.

They are relatively cheap (250€ for unreliable entry level lidars, 1000-5000€ for reliable models).

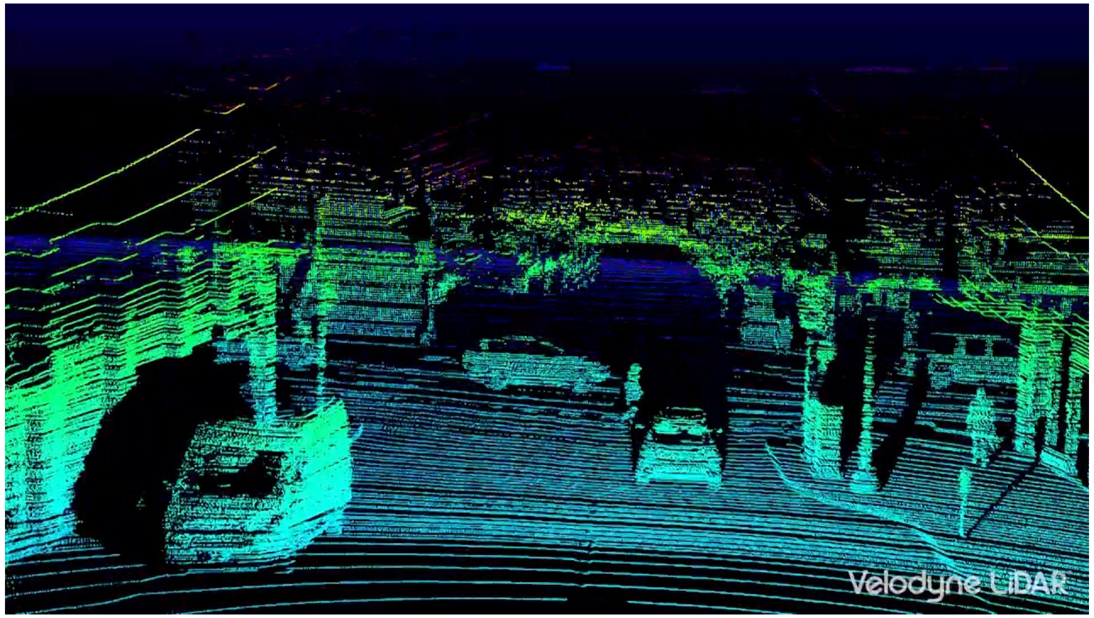
Outdoor lidars have a range from 10 to 30-50m, with a FOV of 180-360°.

Price is higher (5-15k €) but still reasonable, performance are good.

Laser range finders - Lidars

In outdoor applications (autonomous vehicles), 3D lidars are a popular choice:

- Multi-layered lidar, not really 3D (from a single source)
- Usually 360° FOV
- Usually longer ranges (up to 200m)
- Expensive (10k-100k€)
- More data but also more complex to interpret



Vision

With the increasing success of Deep Learning, vision has becoming more and more important in robotics

Cameras provide a lot of data, are *relatively cheap*, but their output is also much more complex to interpret than the one of LIDARS.

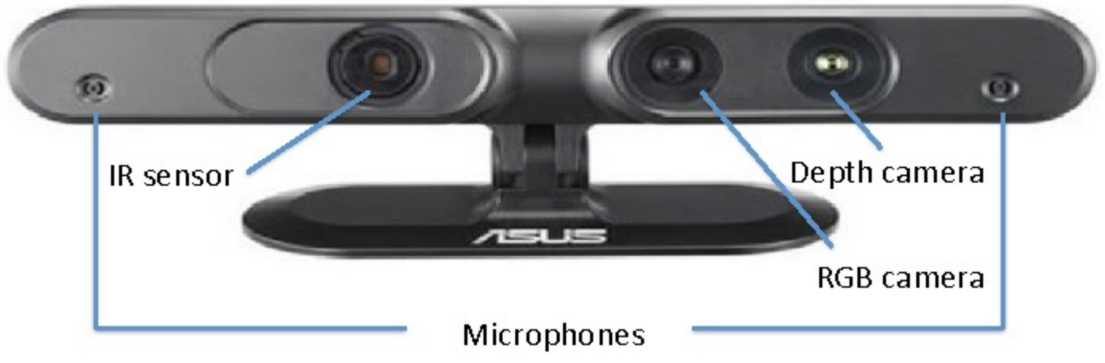
- Limited range
- Distortion
- *Reliability* (day-night or light changes)
- Calibration

Lidar VS Camera

Lidar	Camera
• Cheap	• Cheap
• Long range	• Close range
• Up to 360° FOV	• Limited FOV
• Usually 2D	• 3D
• Simple output	• Complex output
• Subject to reflections	• Subject to distortions, changing light conditions, …
• Measure the spatial surrounding of the robot	• Measure the appearance of the surrounding of the robot
• Good to infer spatial occupancy, difficult to infer semantics	• Could be used to infer semantic knowledge

RGBD Cameras

- camera + depth information using an active sensor
- easy to reconstruct 3D image of the environment
- good for a lot of sensing tasks (e.g., human detection, obstacle)
- widely used and useful, especially indoor
- limited range - depth (usable range < 3/5m)
- distortion
- cheap (100€→1000€)
- do not replace vision
  (poor camera quality)



(b) Asus Xtion Sensor

Other sensors types

As robot can perform several different tasks, robots could be equipped with different type of sensors:

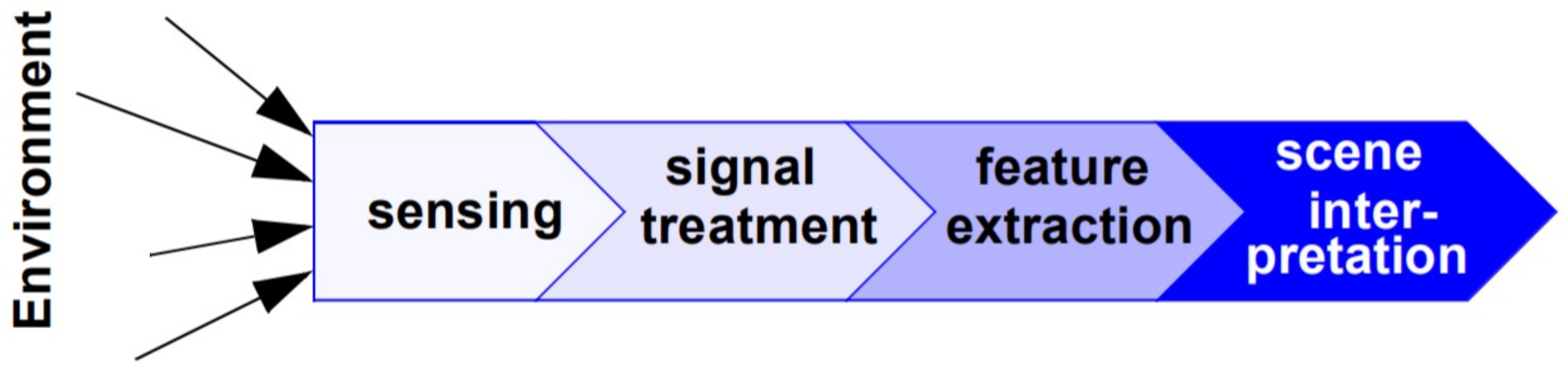
- Bumpers
- Olfactometry
- Chemicals
- Temperature sensors
- NFC readers
- RFID readers
- Radio – or other communication mechanisms…
- …

While you can expect to find one or more lidar/cameras on robot, other type of sensors are relative to the robot type/task.

Sensors wrap-up

- Robot usually have *several* sensors that are used sometimes for acquiring data related to the same subproblem, sometimes for different subproblems
- Laser range finders and cameras are usually combined
- More sensors = more data = more computational capacity required and more complexity (especially for vision)
- RGBD data are often a good compromise between data quality and complexity, but are rarely used as primary source of sensors
- There is a shift towards pure vision-based systems due also to the popularity of computer vision and deep learning, (however, this might be a trend)
- All robot data are defined by <u>errors</u> and <u>uncertainty</u> that have to be modeled (this is “easy” for lidars, but what about vision?)

Perception and Feature Extraction



To reduce the impact of inaccurate sensor readings, an idea is to extract features from one (or more) sensor data:

- Low level features: geometric primitives (lines, edges, corners)
- High level features: semantic labeling (object detection, people detection, …)

This depends on the sensors type / data quality, the environment, and the computational power / frequency required to process data

From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2011

Vision-related tasks

- Structure from stereo:
  Reconstruct the depth (the distance of objects/obstacles) from two or more cameras, images obtained at the same time
  How: matching common features in both images (left, right)
- Structure from motion:
  Reconstruct the depth and structure of objects from a flow of images obtained from one/multiple cameras in a sequence
- Visual odometry:
  Estimate the motion of the robot/vehicle from visual input alone
- Colour tracking:
  Following an input with a predefined colour easy to identify (e.g., an orange ball, a line on the ground). Used in low-cost or simple platforms (e.g., teaching robots, RoboCup). Allows a reliable and fast recognition of landmarks with a low computational effort – works also with cheap sensors.

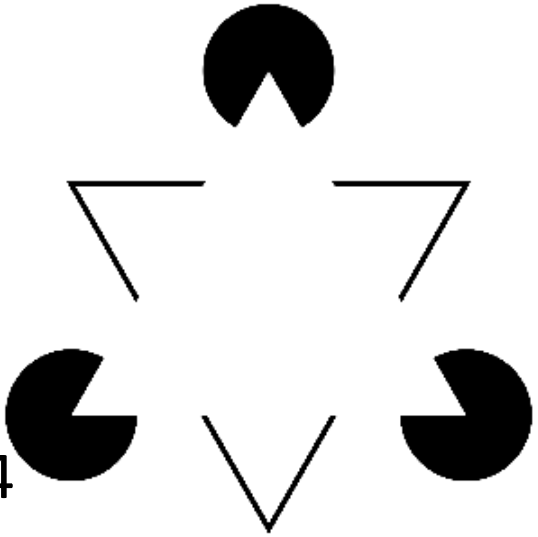
Features extraction and computation: two issues

Features extraction is a way to extract *a few meaningful information* from a *dense and complex* input

- We are able to identify features/pattern easily, robots don’t

E.g. an 8K video streams has a bandwidth of 300Mb/s

- How much of that info is needed by the robot? E.g., lines are stable
- How much of that info can be processed in real time? E.g. ResNet18 processes images of size 224x224



Desired properties of features in robotics

- Features are recognizable structures of elements in the environment.
- Extracted from measurements and mathematically described. (Model based – you select the features to search)
- Recent trends as deep learning allows a black-box extraction of features. (The networks computes its own features)
- Good features are always *perceivable* and easily *detectable* from the environment (by us)
- Raw sensor data provide a large volume of data, but with low distinctiveness of each individual quantum of data. No loss of information.
- Low-level to high-level features are abstractions of raw data, and as such they provide a lower volume of data while increasing the distinctiveness of each feature.
- Ideally, the goal is to filter out «useless» data, keeping all good ones.

More properties of features

- Localization accuracy: features should be easily identifiable and localizable wrt the world model
- Quantity of features: the «right amount»;
  E.g., edge detection, 3D reconstruction.
- Invariance: not affected by changes of viewpoint, illumination, scale
- Computationally efficient
- Robustness: image noise, artifacts, blur, distortions should not affect the feature detection.

High-level features: semantic knowledge

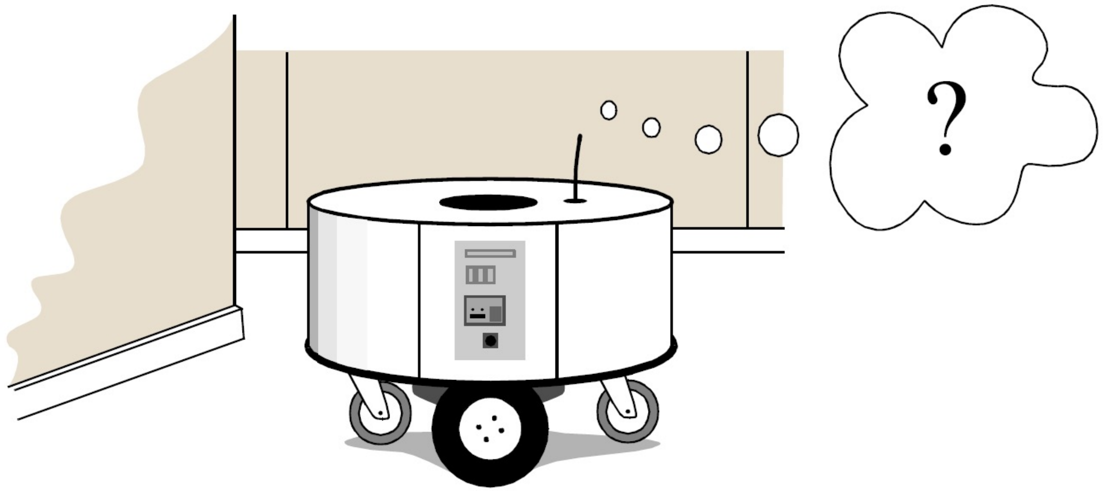
Semantic regards the task of giving a «meaning» to perception.
Usually done using vision (+ depth).
Deep learning improved (a lot) these abilities in the last years.

- Place recognition:
  Identification of rooms/locations (corridor, kitchen, office, …)
- Object detection:
  Identify an object in front of the robot
- Semantic segmentation:
  Identify which region in an image is which object
- Loop closure:
  Identify that the robot has already observed the same location (perhaps from a different point of view).

Outline

Overview of core concepts involving robot mobility:

- Robot Motion
- Perception
- <u>Localization and Mapping</u>
- Navigation



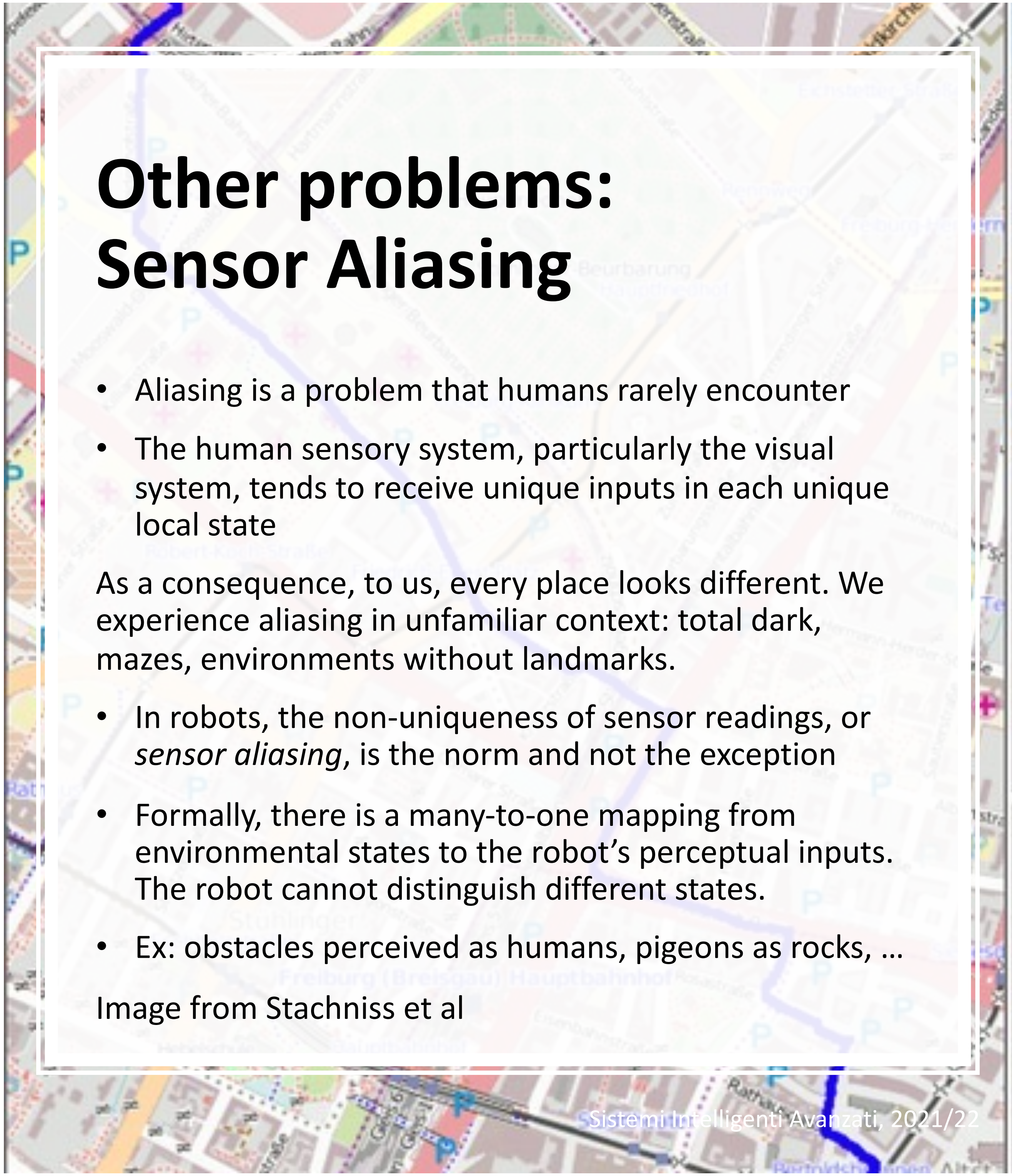
From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2011

Assumption: let’s talk about the simplest type of mobile robots, wheeled ground vehicles

Localization and Mapping issues

Robot mobility requires addressing a key property: ***uncertainty***

- Environment: the world is unpredictable
- Sensors: sensors have limits, are subject to physical laws, and are subjects to noise and errors
- Robots: actuation is unpredictable, an action can not have the desired effect
- Models: we can model certain components of the robot, and also uncertainty; but models are inherently inaccurate, models are an abstraction
- Computation: robot acts as real-time system, but usually are not real-time system.
  Real time computation is often approximated (quasi real-time).



Other problems: Sensor Aliasing

- Aliasing is a problem that humans rarely encounter
- The human sensory system, particularly the visual system, tends to receive unique inputs in each unique local state

As a consequence, to us, every place looks different. We experience aliasing in unfamiliar context: total dark, mazes, environments without landmarks.

- In robots, the non-uniqueness of sensor readings, or *sensor aliasing*, is the norm and not the exception
- Formally, there is a many-to-one mapping from environmental states to the robot's perceptual inputs. The robot cannot distinguish different states.
- Ex: obstacles perceived as humans, pigeons as rocks, …

Image from Stachniss et al

Other problems: effector noise

- Similarly to robot sensors that are noisy, robot effectors are also noisy
- As a consequence, the effects of a robot actions are not entirely observable and uncertain
- If not handled properly, errors accumulates over time

Ex: wheeled robots effector noise:

- Limited resolution of odometry
- Misalignment of wheels
- Unequal wheel diameter
- Variation in the contact point of the wheels
- Tire pressure
- …

Localization

Identifying the position of the robot in a **known** environment:

It needs:

- A model of the robot
- A map of the environment,
- Perception (sensor readings)

Localization is not a one-shot task: we need to maintain the robot localized and update its position while it is moving.

Localization is usually seen as an estimation problem, where we infer the robot position from available data modelling the robot.

Localization vs Dead-reckoning

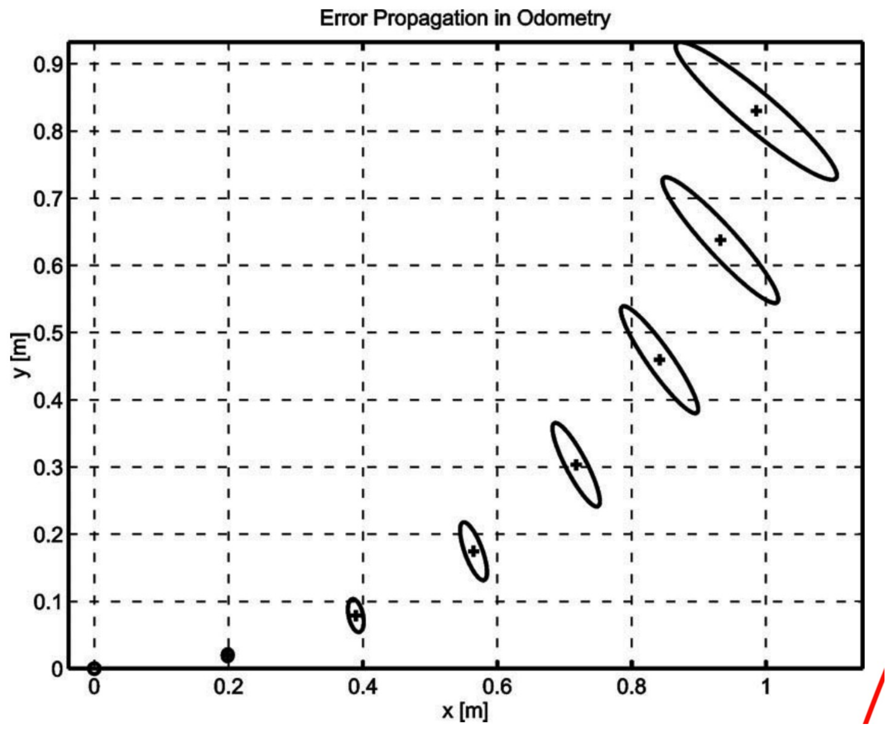
Odometry:

open loop estimation of the robot motion from sensor readings.

Dead-reckoning:
From an initial pose, open loop estimation of the robot position using odometry.

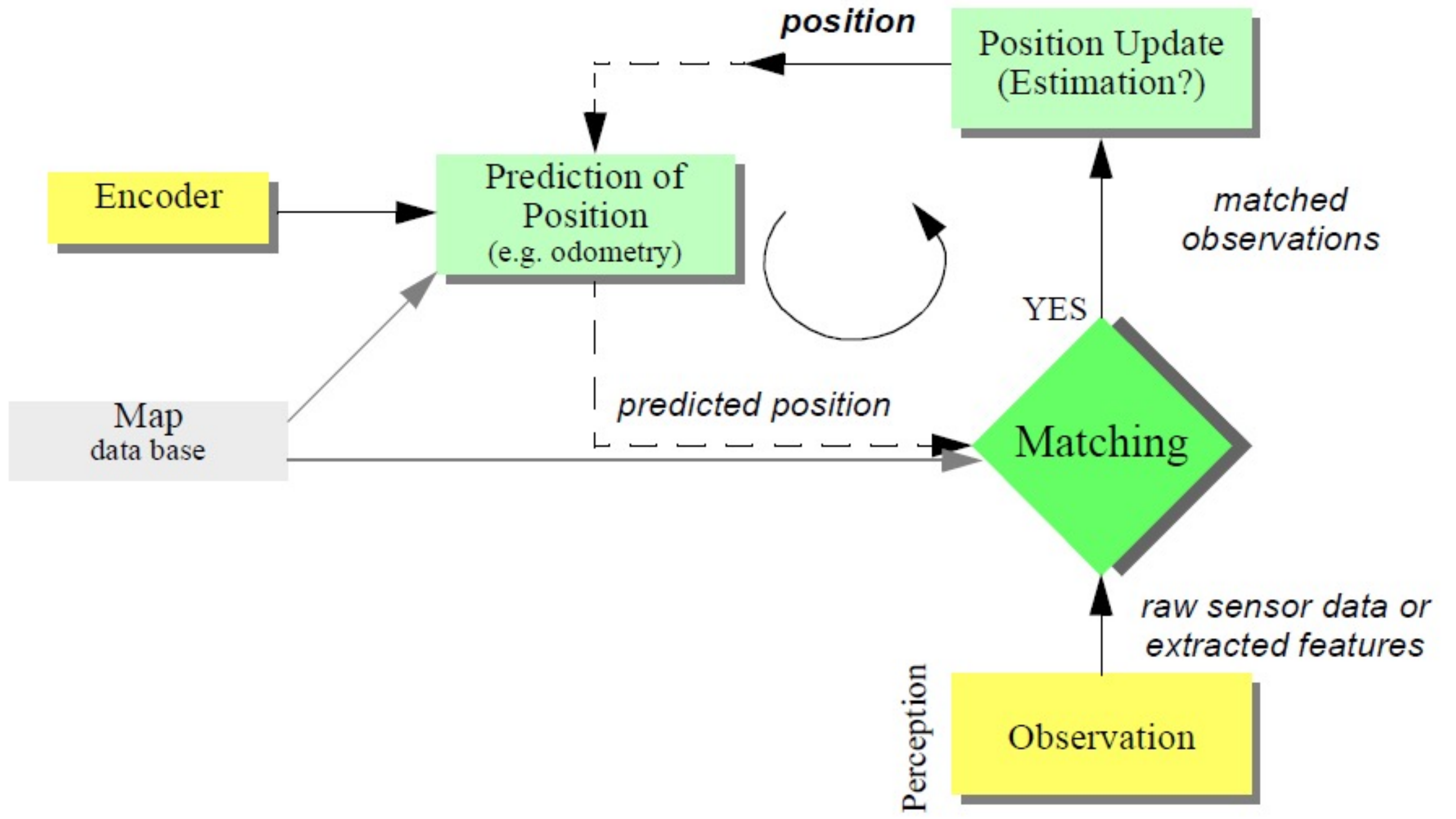
Localization:

Usually closed loop integration of the robot position



Localization

Identifying the position of the robot in a **known** environment


position
Position Update (Estimation?)
Encoder
Prediction of Position (e.g. odometry)
matched observations
YES
Map data base
predicted position
Matching
raw sensor data or extracted features
Perception
Observation

Localization is usually seen as an estimation problem, where we infer the robot position from available data modelling the robot

Localization as estimation

Continuous estimate the robot position from data:

- Motion information:
  - Proprioceptive sensors, odometry
- Environmental Measurements :
  - Exteroceptive sensors as lidars, sonar, …

$$x_r = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Robot pose

$x_{r,0:t} = \{x_{r,0}, x_{r,1}, \ldots, x_{r,t}\}$ Robot poses from time 0 to time t

$z_{1:t} = \{z_1, z_2, \ldots, z_t\}$ Robot exteroceptive measurements from time 1 to time t

$u_{0:t} = \{u_0, u_1, \ldots, u_t\}$ Motion commands (or proprioceptive measurements) from time 0 to time t

Usually solved as using probabilistic filtering

Motion model

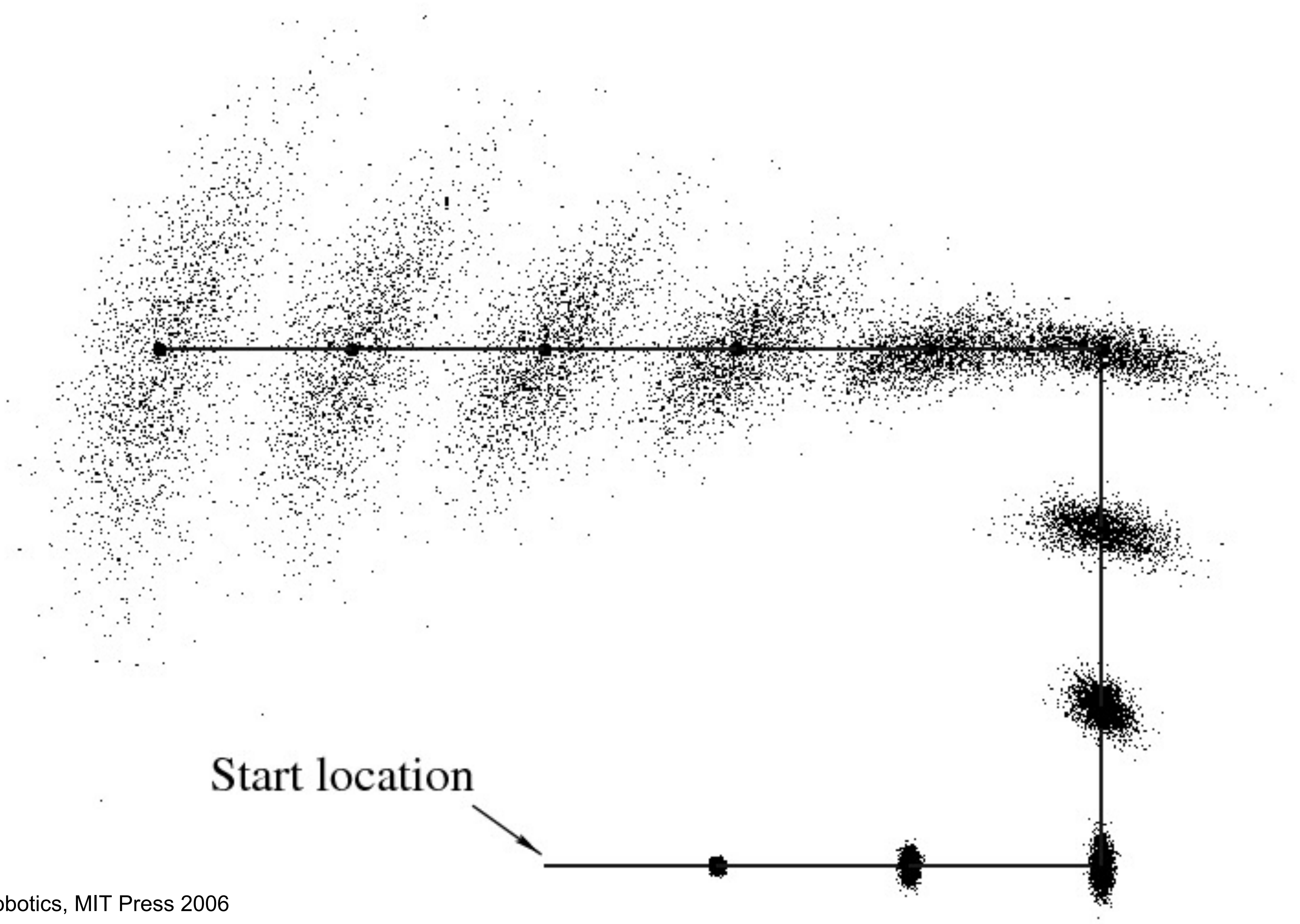
The robot motion model is the probability distribution of the robot pose at time t+1 given the robot pose and the expected robot movement, measured using motion or proprioceptive sensors:

$$p\left(x_{r,t+1} \middle| x_{r,t}, u_t\right)$$

Assuming that the robot is at $x_{r,t}$ and the control u_t is applied, we estimate the expected robot position

Motion model

Using only proprioceptive measurements, pose estimation error increases



From Thrun Burgard Fox, Probabilistic Robotics, MIT Press 2006

Measurement model

It describes the probability of a robot measurement z_t

$$p(z_t | x_{r,t})$$

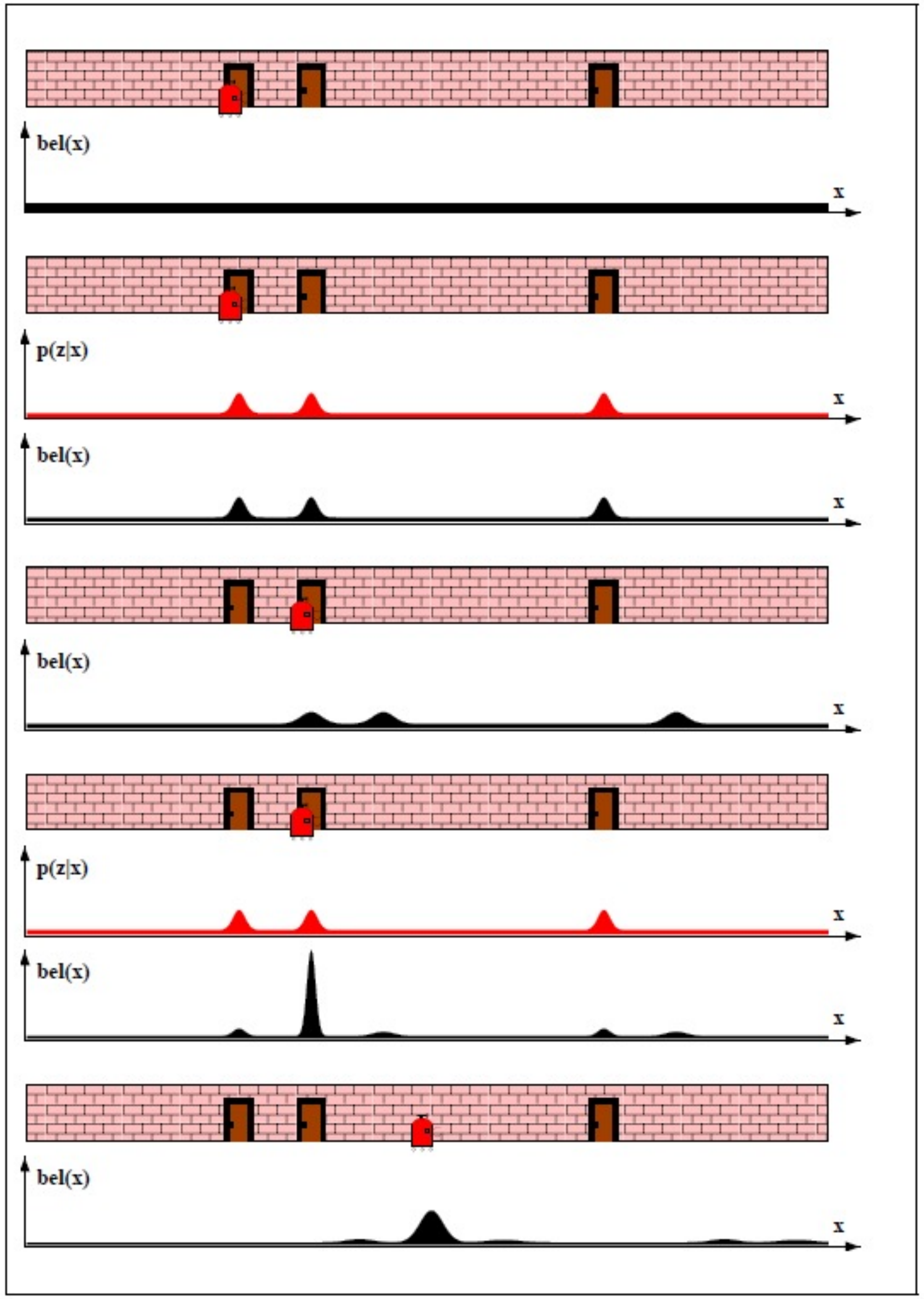
given a robot pose $x_{r,t}$ considering possible noise regarding sensors.

This is used to update the robot belief at time t

$$bel_t(x_r) = p(x_{r,t} = x_r | z_{1:t}, u_{0:t-1})$$

The robot belief is a probability distribution over the space of all possible locations of the current robot pose

Localization example



From Thrun Burgard Fox, Probabilistic Robotics, MIT Press 2006

Robot localization algorithms

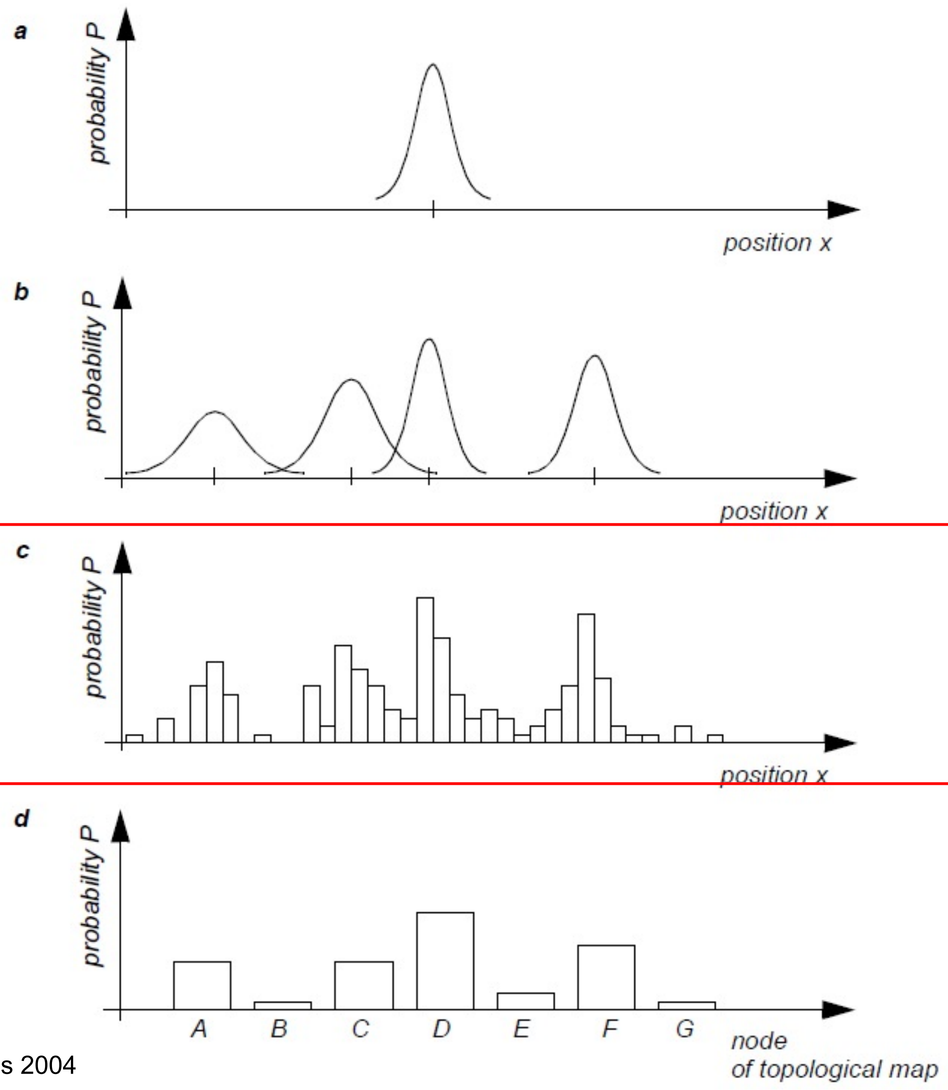
The localization model assumes so to predict the current position from movement, to observe if the measurements are coherent with the estimated position, and to close the loop by updating the robot belief.

This is done usually by exploiting probabilistic filters:

Gaussian Filters:	**Extended Kalman Filter (EKF),**
	Unscented Kalman Filter (UKF),
	Extended Information Filter (EIF)
Non Parametric Filters:	**Histogram Filter (HF),**
	Particle Filter (PF)

Belief representation

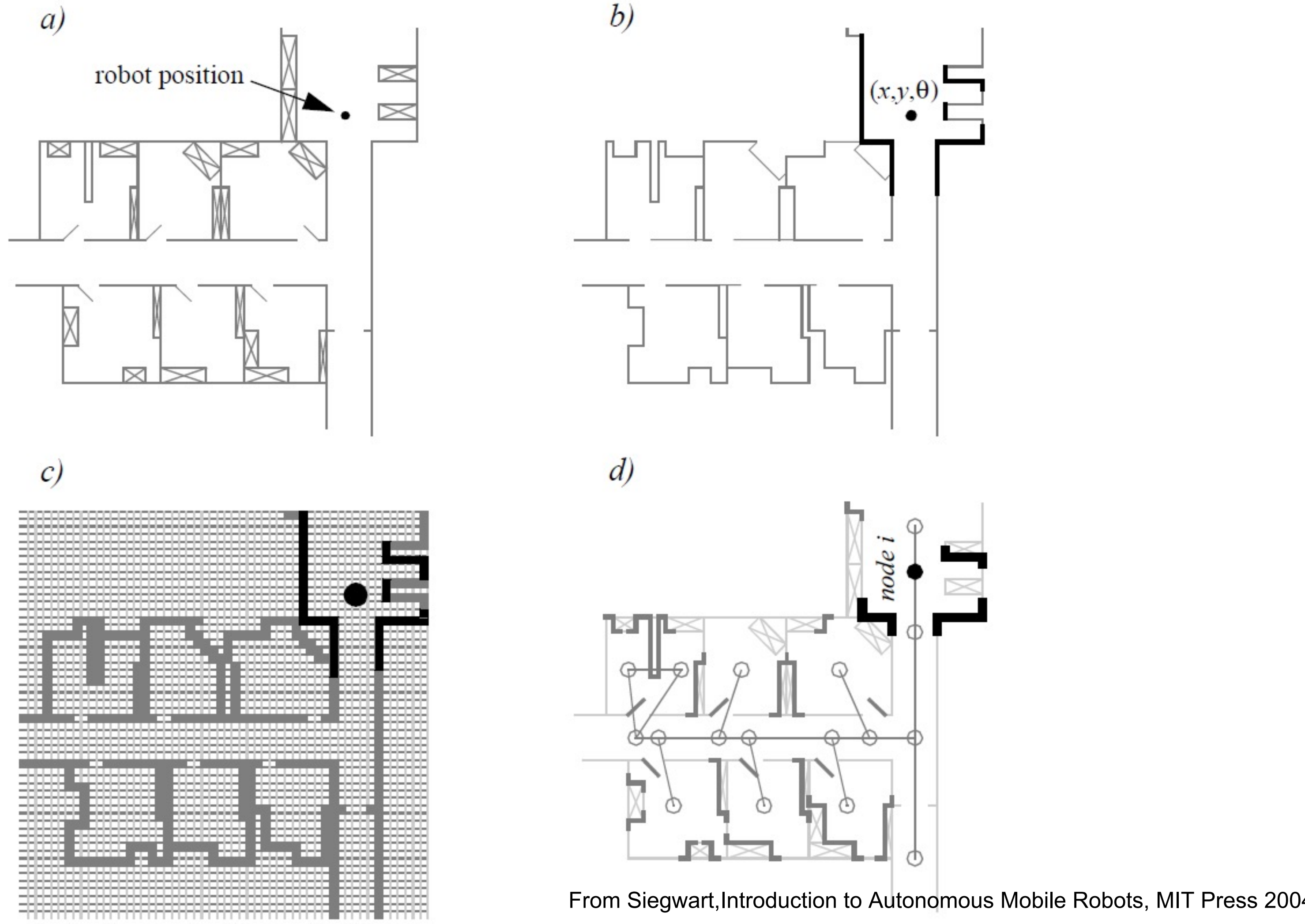
We can represent the belief as a single hypothesis of by have a distribution probability
over multiple hypotesis



From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

Map representation

According to the algorithm used for localization, the type of belief distribution, we can have multiple type of map representations



From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

Map representation

Usually a robot has different maps, at different level of abstraction; one of them is the one used for localization.

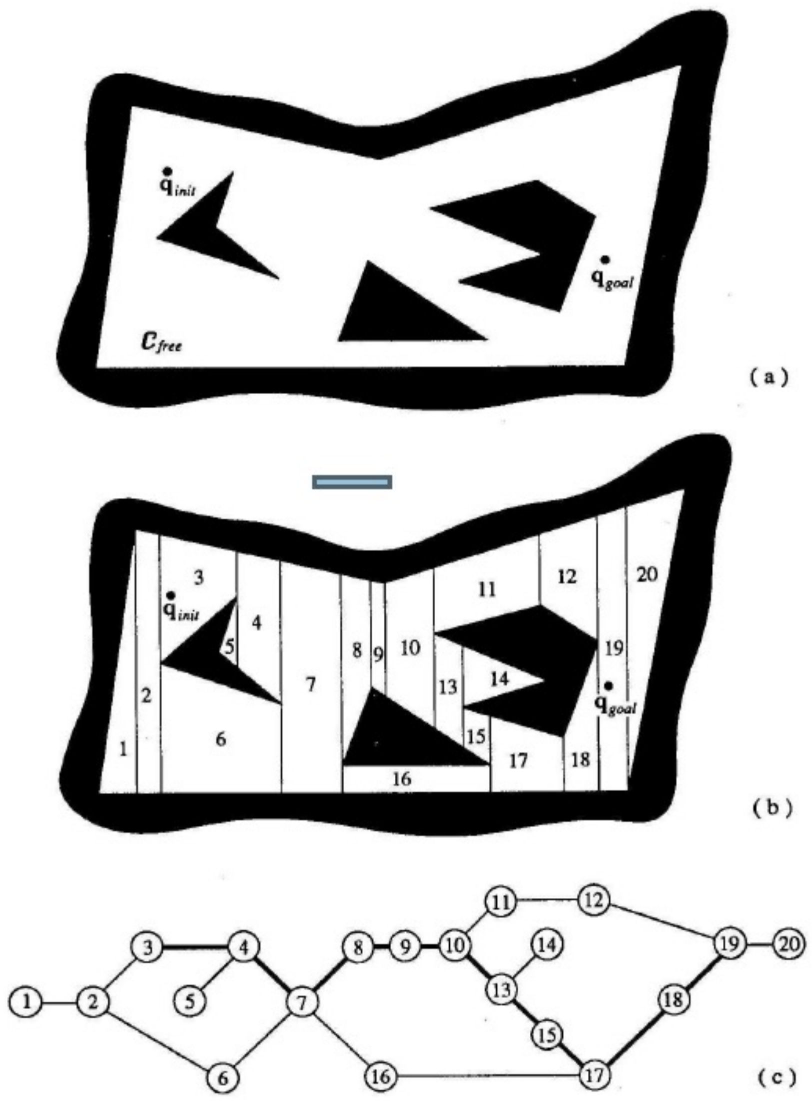
- Continuous vs discrete representation
- Occupancy vs topological maps
- Closed world assumption: only what there is in the map exists
- <u>Static</u> vs dynamic
- 2D or 3D

A map, overall, is an approximation of the environment.

Exact Cell Decomposition

This method use critical points to tesselate environment, obtaining a discrete topological map from a continuous one.

Assumption: the particular position of the robot in one of the area belonging to one node of the map does not matter, that matter is the ability of the robot to move from area to area.



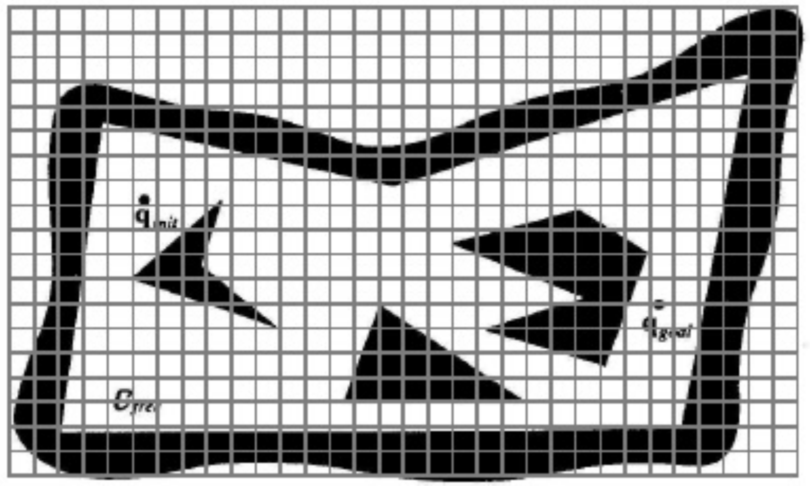
From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

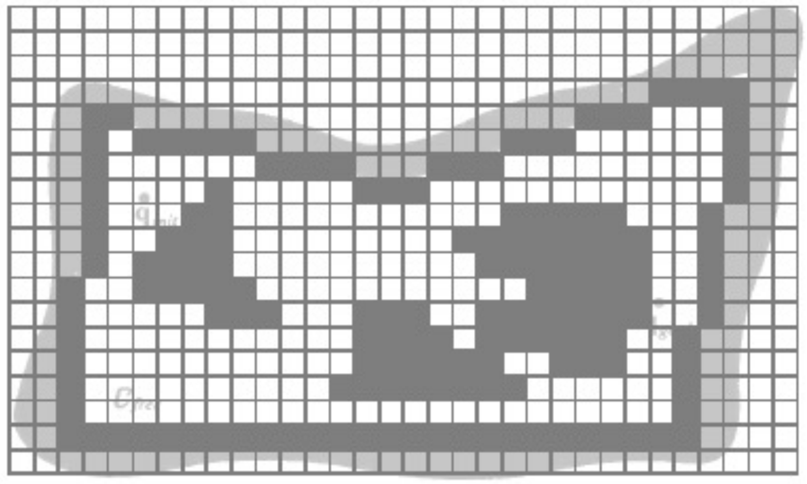
Fixed decomposition maps

Discretization of the map into cells of the same size, representing occupancy.

Narrow passages disappear, but each cell has the same representation.

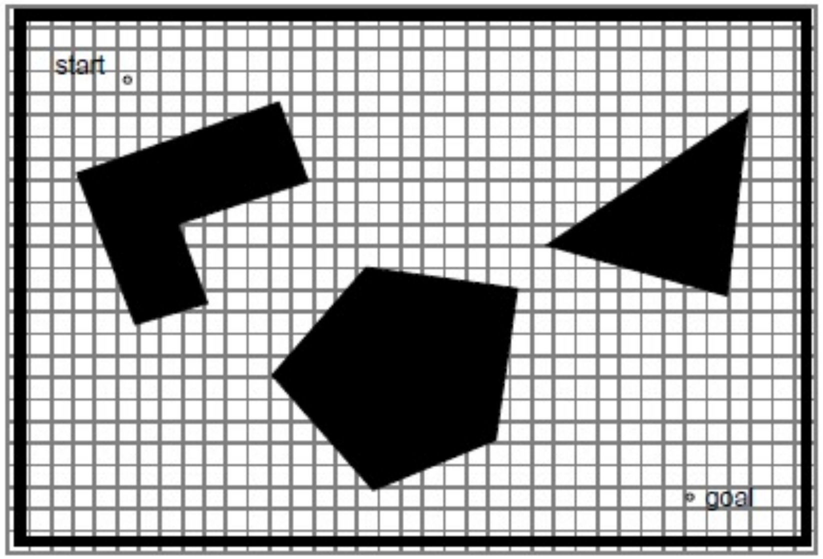
We obtain a grid map.
We can also assign different type of values to each cell (instead of 1-0, e.g. occupancy probability)

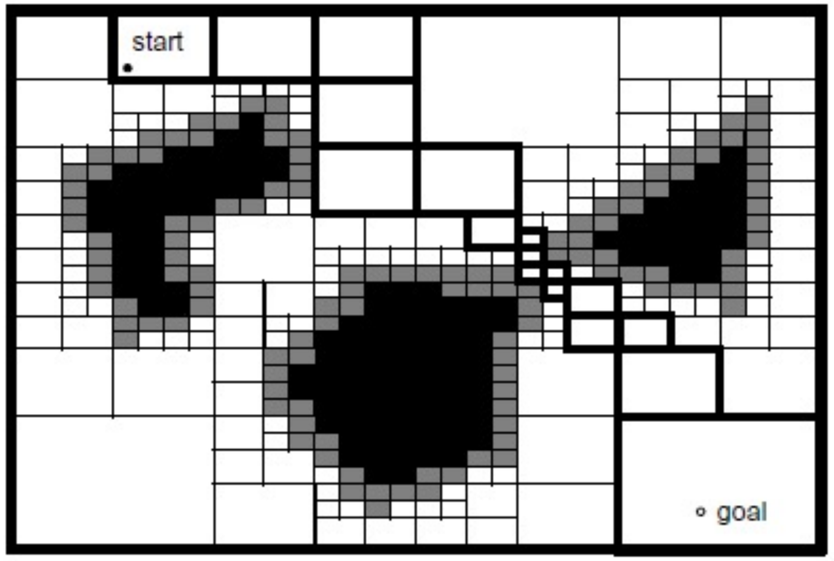


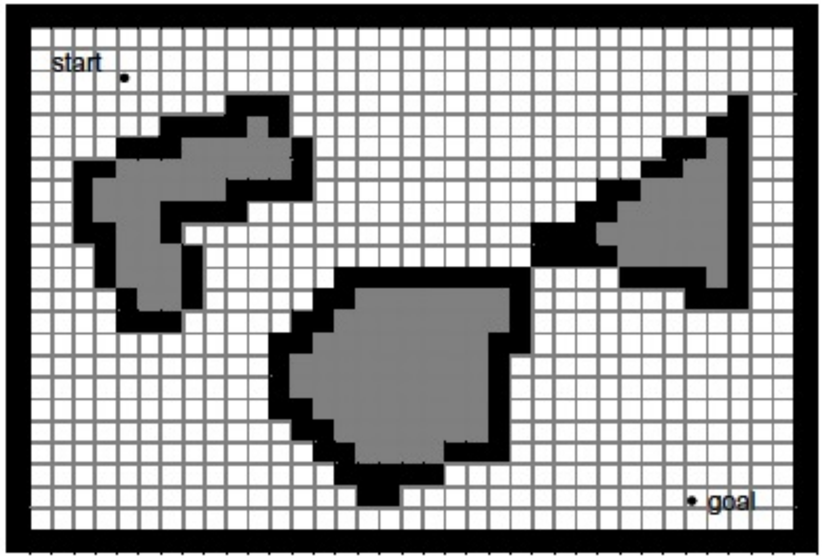


From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

Grid maps





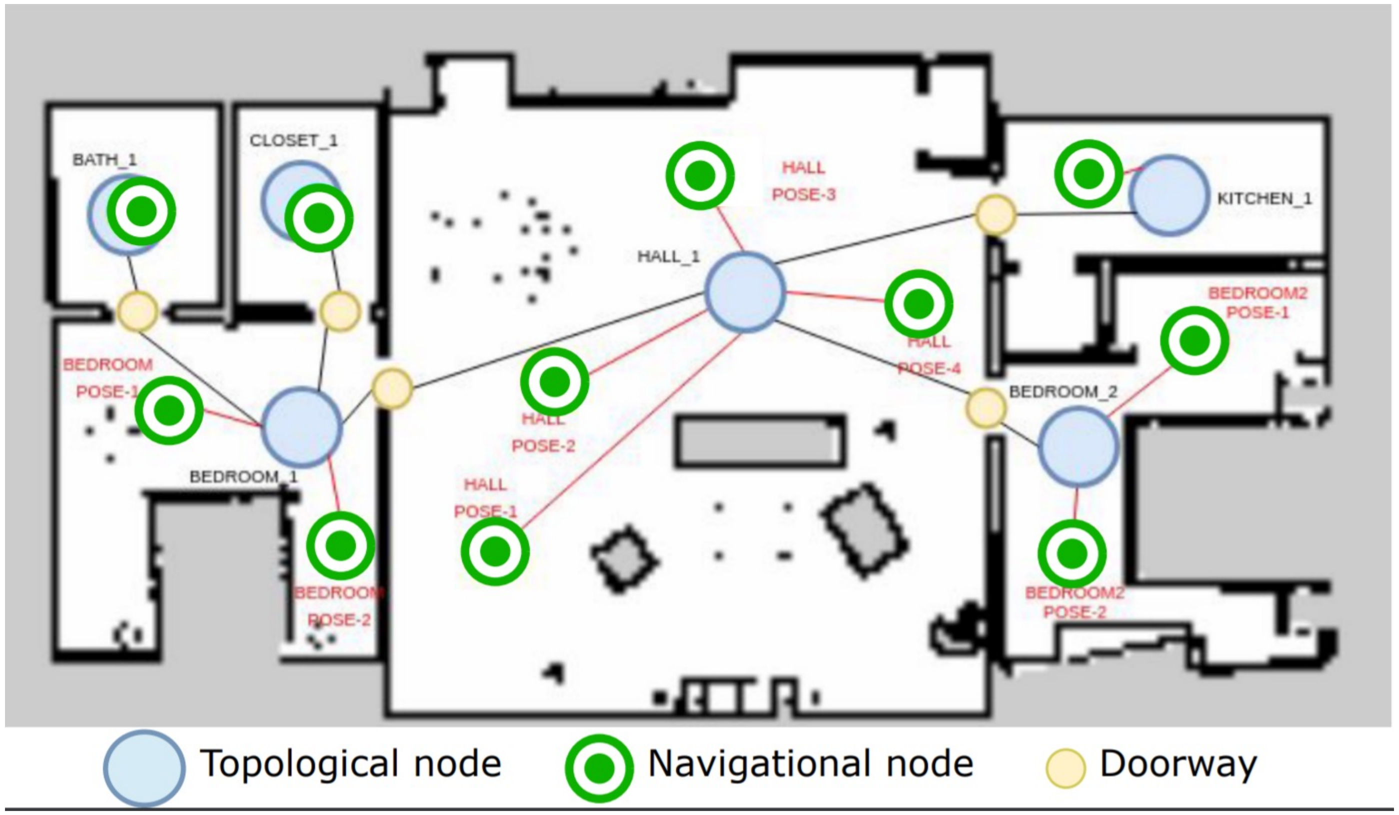


Grid map are a popular approach widely adopted.
As free space is not interesting, we can use cells of different size to optimize the grid map representation.

From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

Hybrid maps

We can have different layered maps, as topological and grid maps, combined, to allow the robot to do different tasks.



SLAM

In all the previous examples, we have considered the map as known. However, what if the robot is placed in an unknown environment?

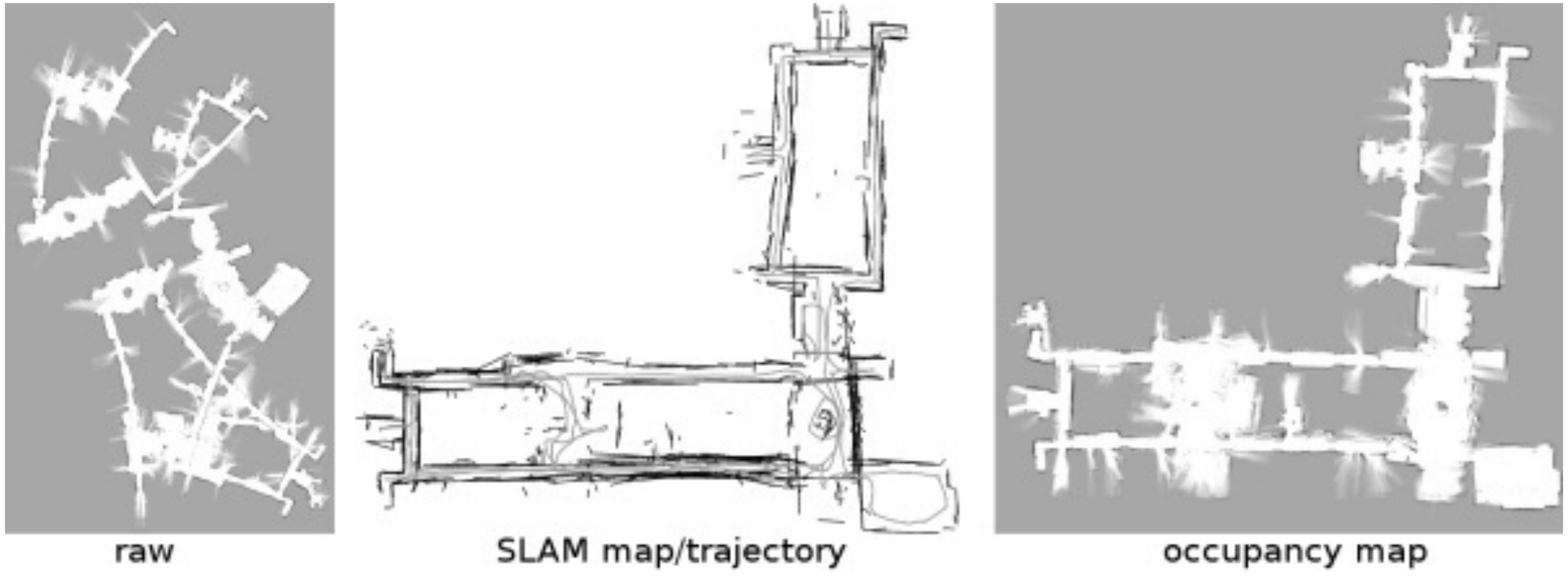
How the map is done in the first place?

The robot needs at the same time to:

1. Map incrementally the environment integrating new observations
2. Localize itself its in the map

This is called Simultaneous Localization and Mapping (SLAM), a joint estimate of both the environment map and the robot pose.

SLAM 101



How to build a map?

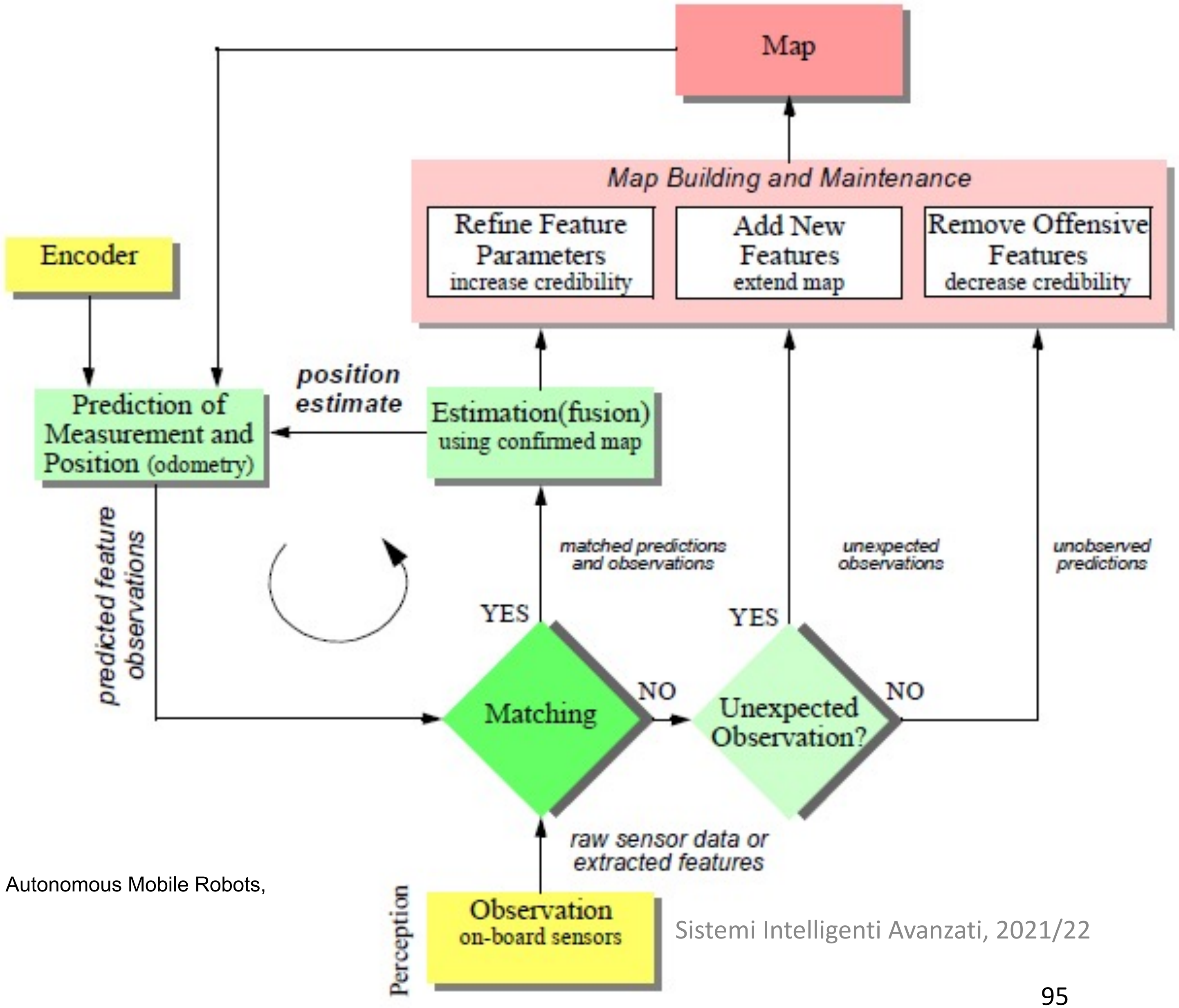
Incrementally, by fusing together sensor readings while correcting sensor error.

When the robot has observed (through sensors) the entire environment, the mapping process is over)

SLAM 101

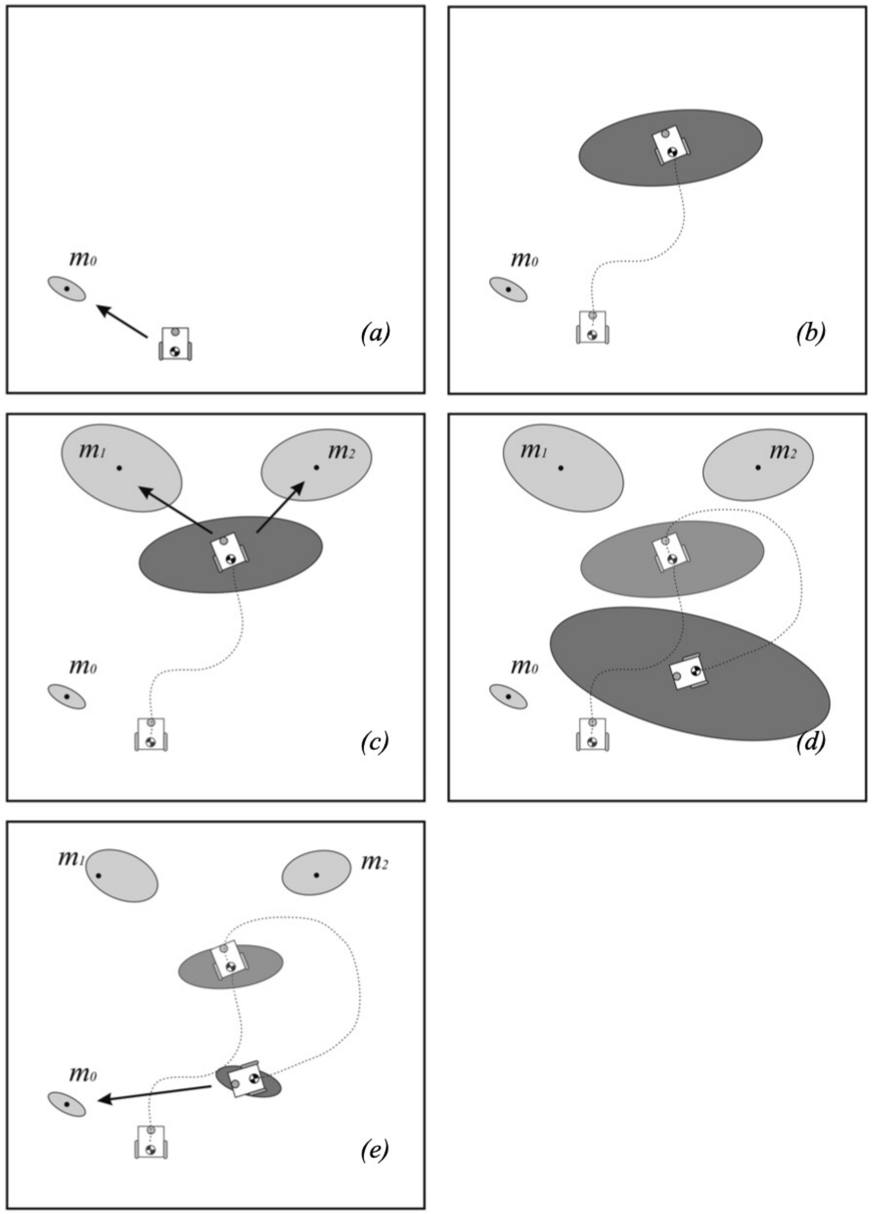
During SLAM the robot integrates sensorial input by correcting odometry and sensing error to provide an estimate of the environment.

At the same time it estimates its pose in it.


Map
Map Building and Maintenance
Refine Feature Parameters increase credibility
Add New Features extend map
Remove Offensive Features decrease credibility
Encoder
Prediction of Measurement and Position (odometry)
position estimate
Estimation(fusion) using confirmed map
predicted feature observations
matched predictions and observations
unexpected observations
unobserved predictions
YES
YES
NO
NO
Matching
Unexpected Observation?
raw sensor data or extracted features
Observation on-board sensors
Perception

From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

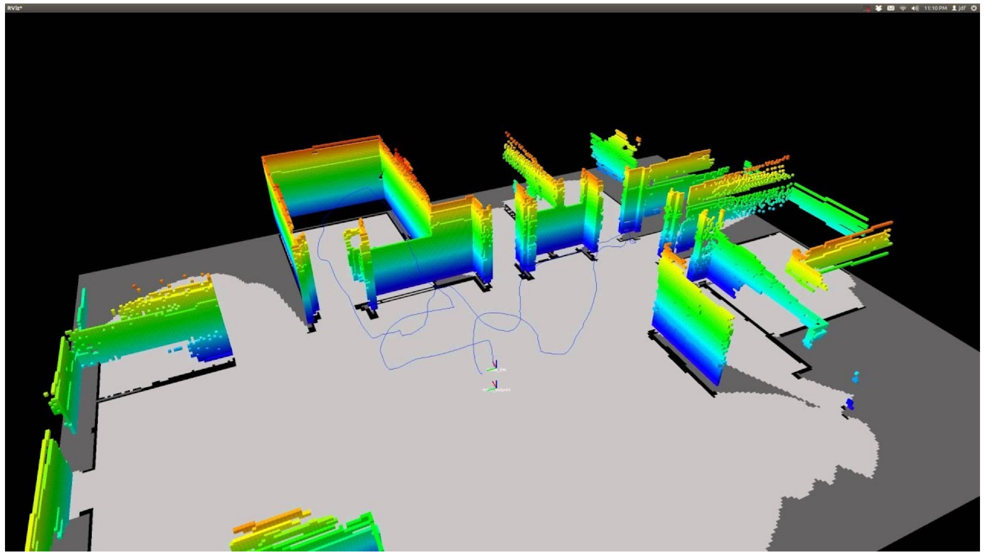
SLAM Example

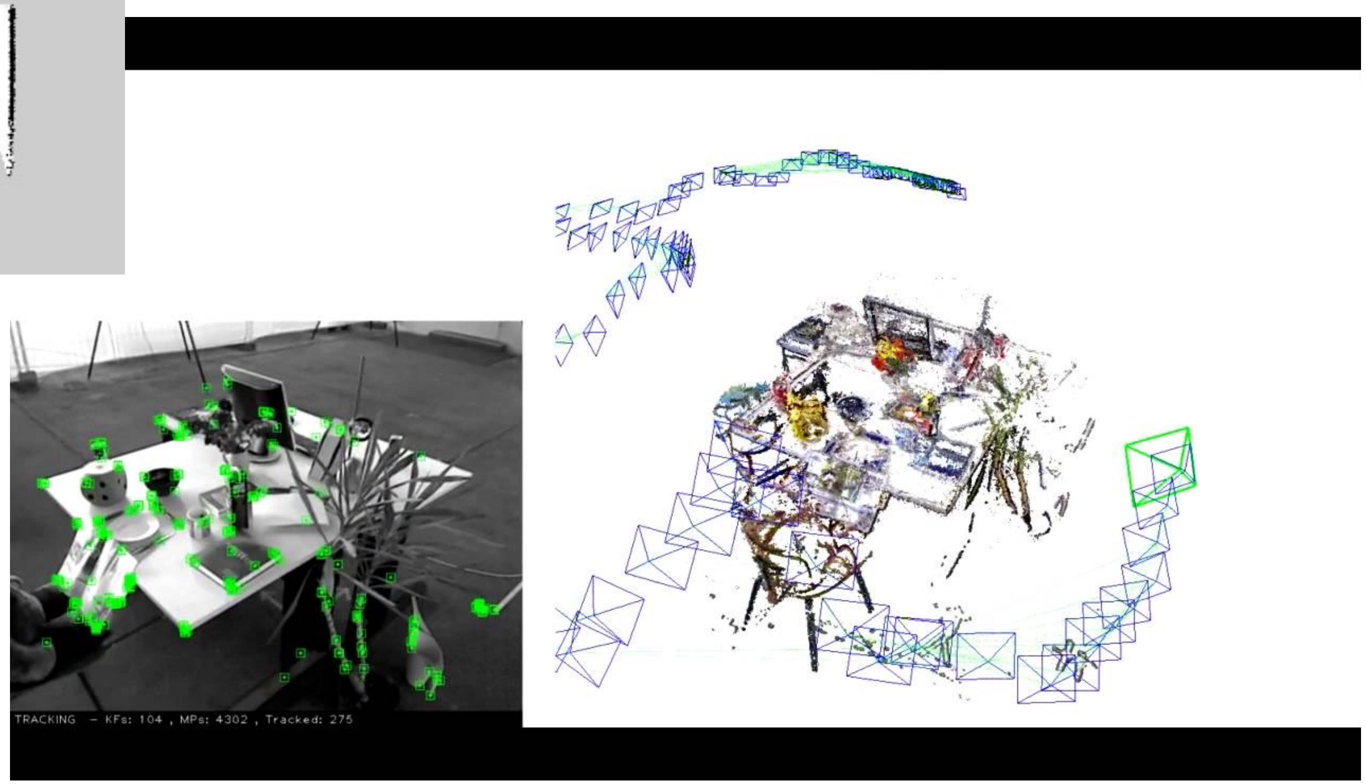


From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2011

Loop closure detection: the robot observes
the same feature twice,
reducing pose uncertainty

SLAM

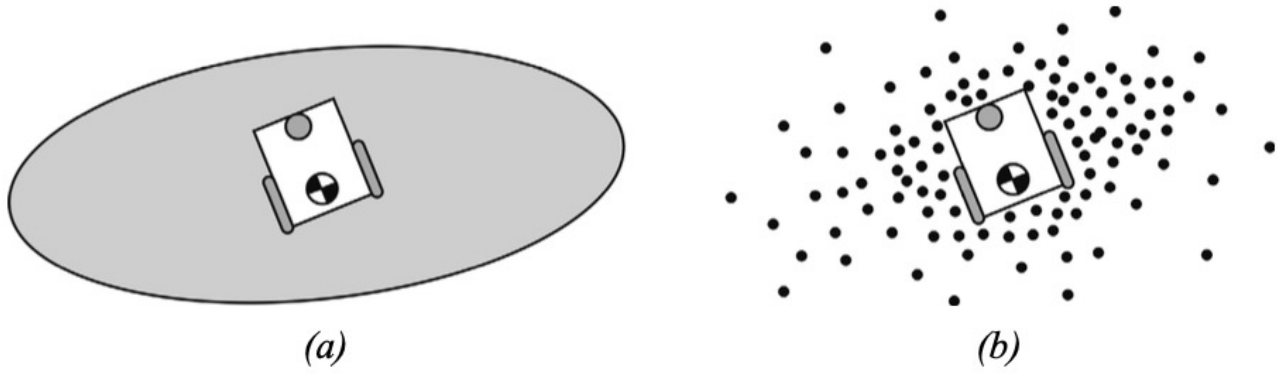




```
TRACKING  - KFs: 104 , MPs: 4302 , Tracked: 275
```

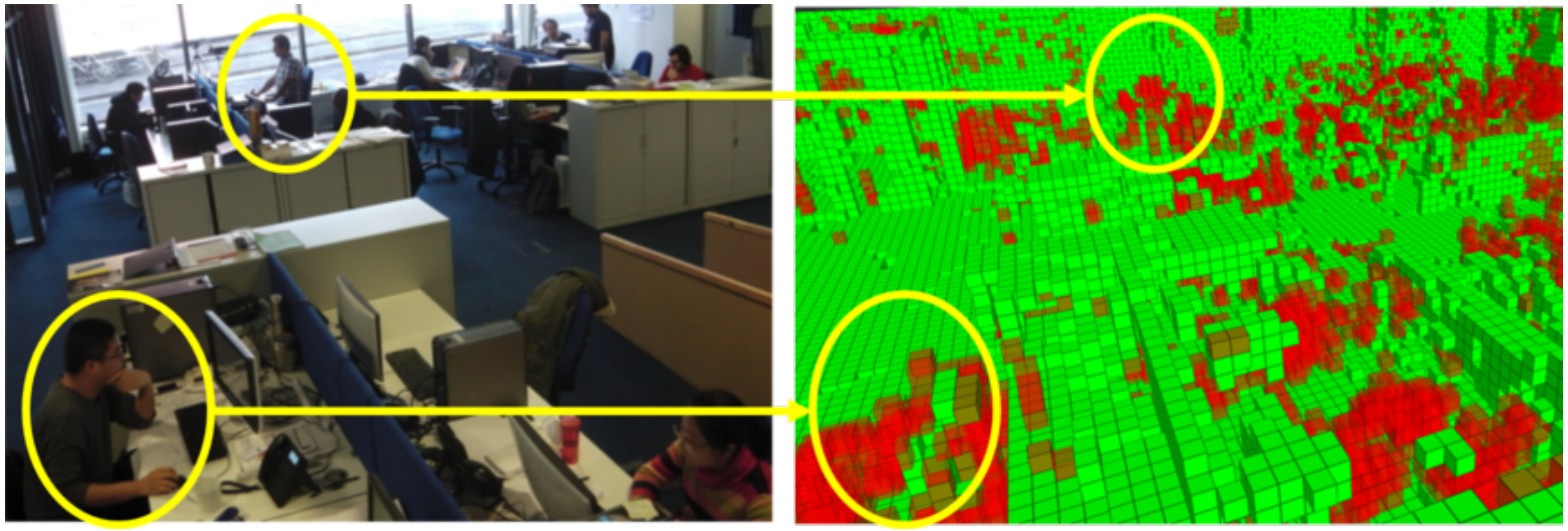
SLAM belief state

- Some SLAM methods represent the probability distribution of the robot location as a parametric form (e.g., a Gaussian)
- Other method (particle filter SLAM) represent it as a set of randomly drawn (and resampled) sampes.
- In this case, the density of the particles is higher towards the center and decreases with distance.
- When a new observation arrives, low-probability samples are discarted and new are randomly sampled.



Open Challenges in Mapping

- The world is dynamic, humans are moving around, the robot is moving, objects can be moved around.
- Maps are represented as static environments. Dynamic objects are not represented (e.g., people moving) and obstacles are represented in a fixed position (e.g., chairs)
- Large-scale and open spaces are difficult to represent
  - How to «segment» a parking lot into different areas in an objective way?
- Closed-world assumption



Alternatives: Landmark-based navigation

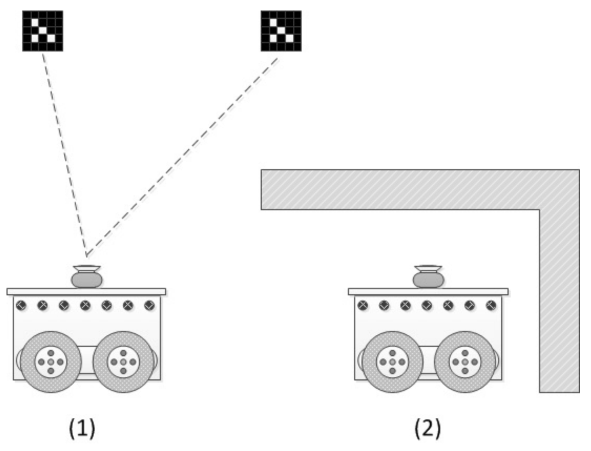
An alternative is to put a set of easily recognizable landmarks in the environment:

- The robot moves in open loop by doing dead-reckoning from landmark A to landmark B
- When the robot detects landmark B, it localize itself there

Advantages: it's a simple yet effective method, robust to changes

Disadvantages: landmark should be put in advance in the environment (that should be modified), the distance between two landmark should be little to reduce the chance of failure during movement while doing dead-reckoning

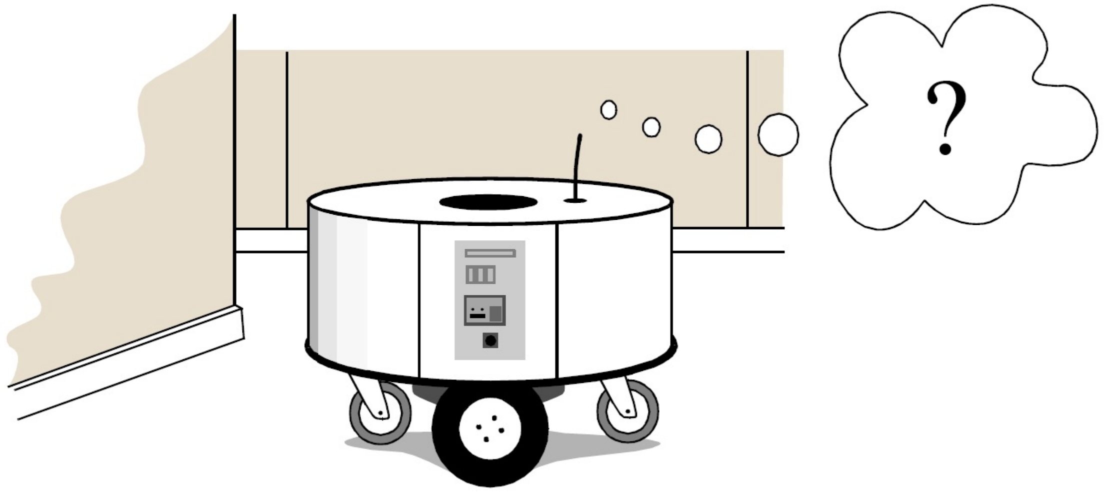
Landmarks: beacons, QR codes or visual markers, usually in placed without obstruction (ceiling)



Outline

Overview of core concepts:

- Robot Motion
- Perception
- Localization and Mapping
- Navigation



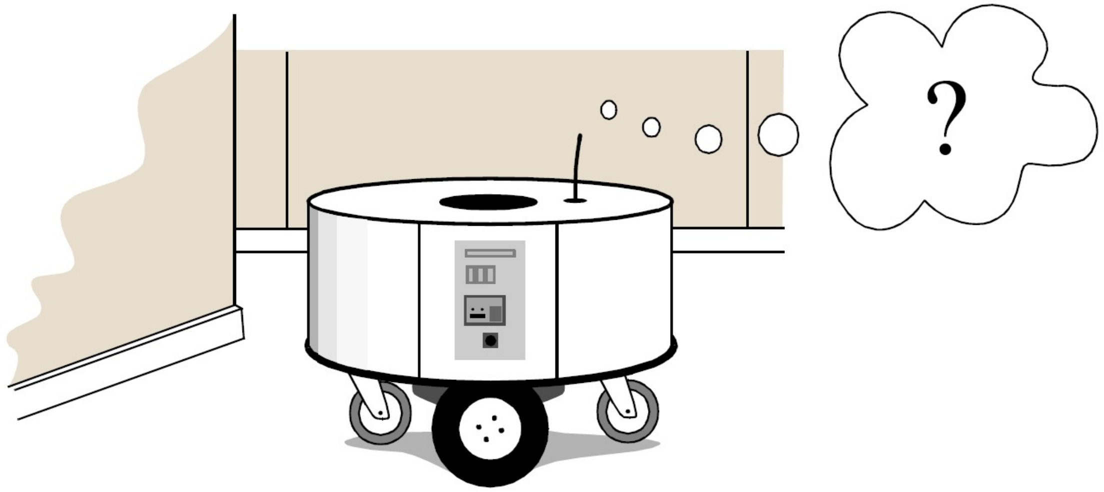
From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

Assumption: let’s talk about the simplest type of mobile robots, wheeled ground vehicles

Navigation

After we have a map, and the robot position, how to go from A to B?

- Path Planning
  ”how to go from A to B”
- Obstacle Avoidance
  “how to avoid obstacles
  while going to A and b”
- Navigation Architecture
  How to integrate everything together



Path Planning approaches

Once we have the map, we have to compute a set of states for finding the path that the robot can execute.

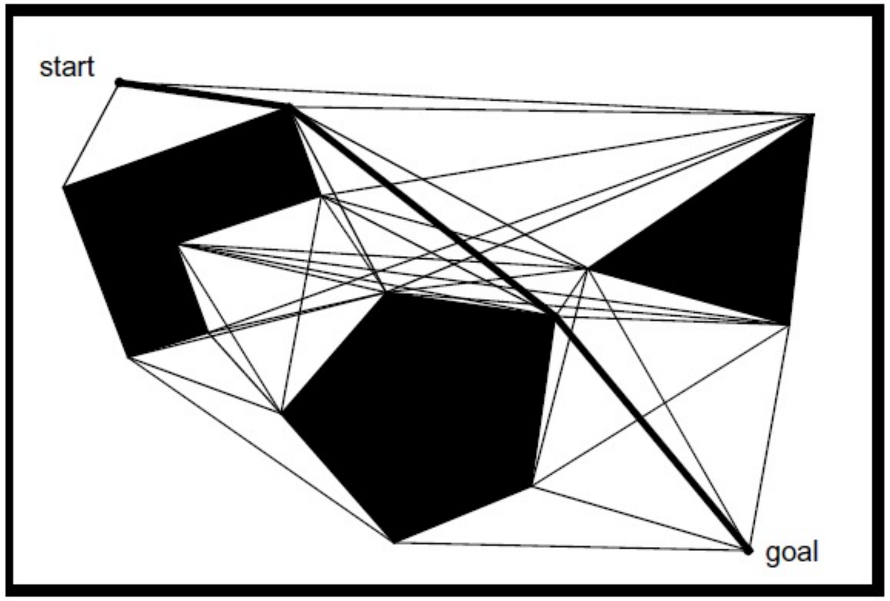
However, as we’ve seen, we have to provide a proper formulation for this problem:

- Road map: identify a set of routes within the free space
- Potential field: impose a mathematical function over the space
- Cell decomposition: discriminate between free and occupied cells

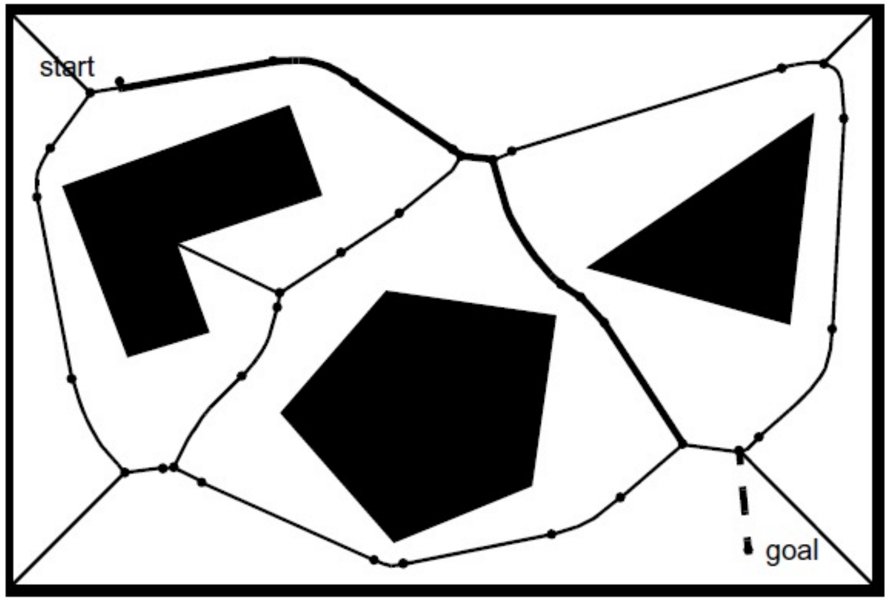
Road map path planning

Idea: develop a network of roads / paths along the environment using a decomposition of the robot traversable free space.

Method used for computing paths:
Voronoi decomposition, direct visibility, …



Visibility Graph

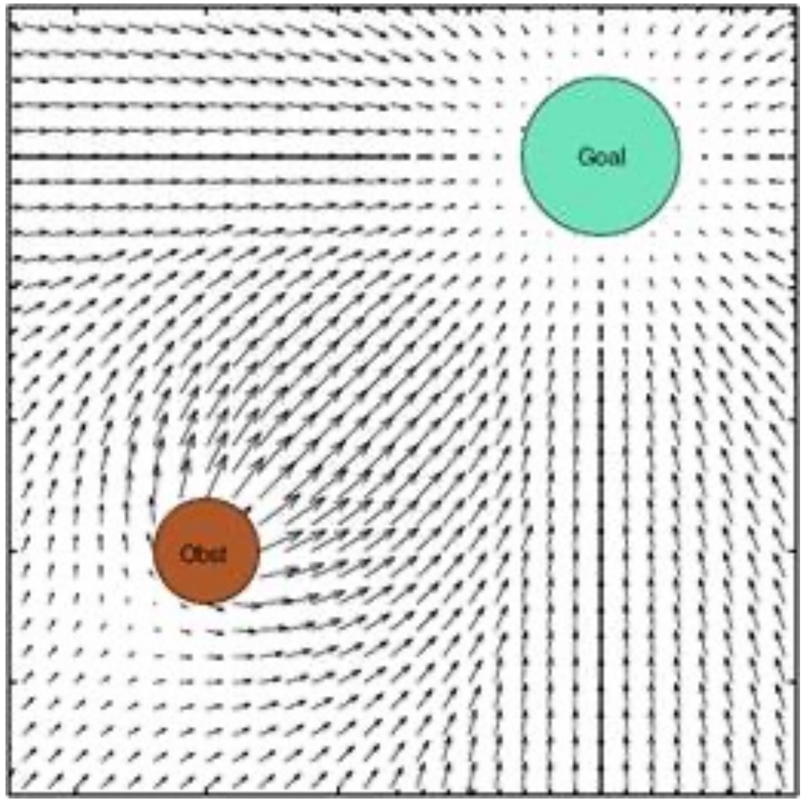


Voronoi Decomposition

From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

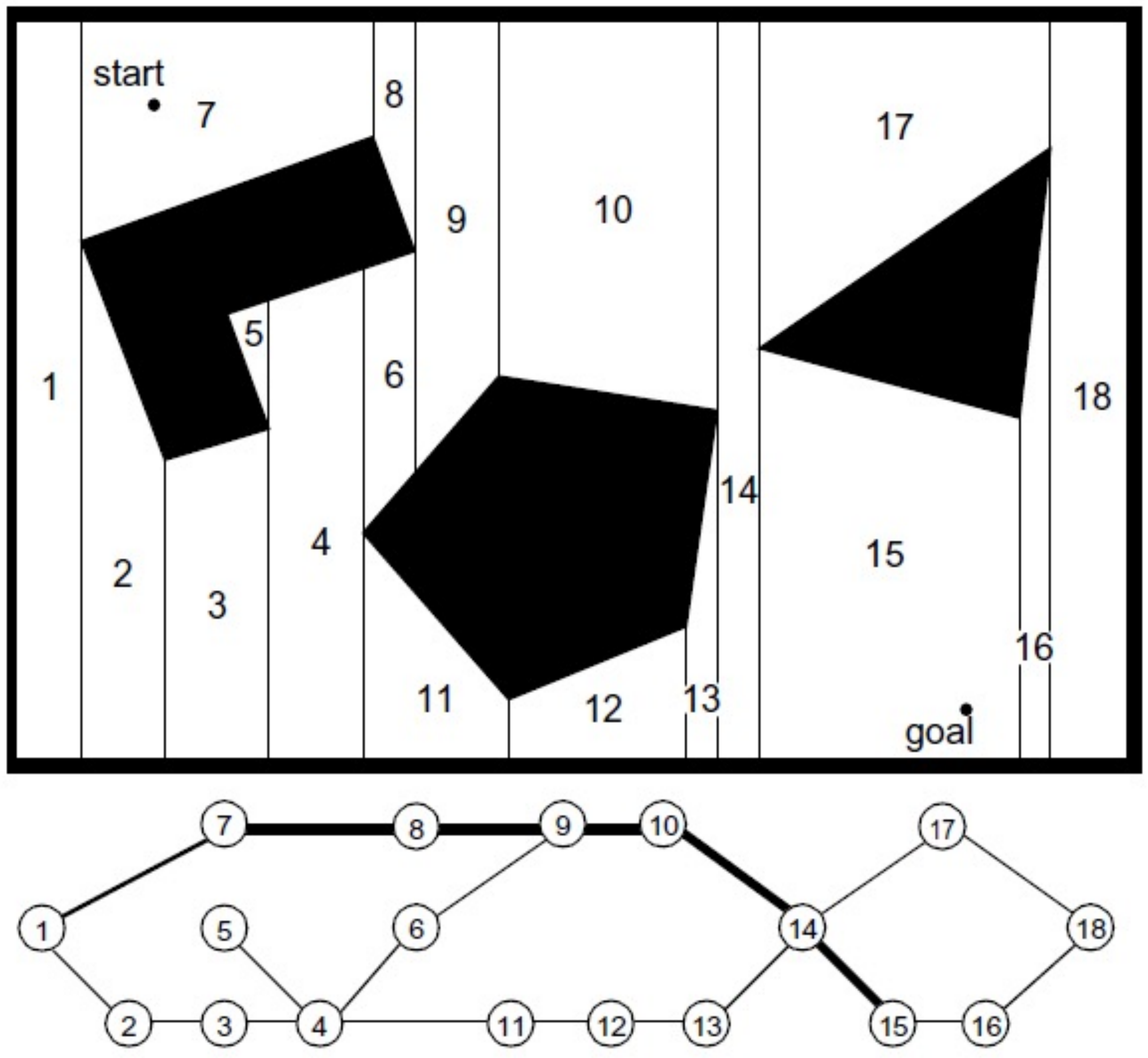
Potential field path planning

Idea: put an attractive artificial potential field on the goal, a repulsive one on obstacles, let the robot follow these simulated forces



from https://www.cs.mcgill.ca/~hsafad/robotics/

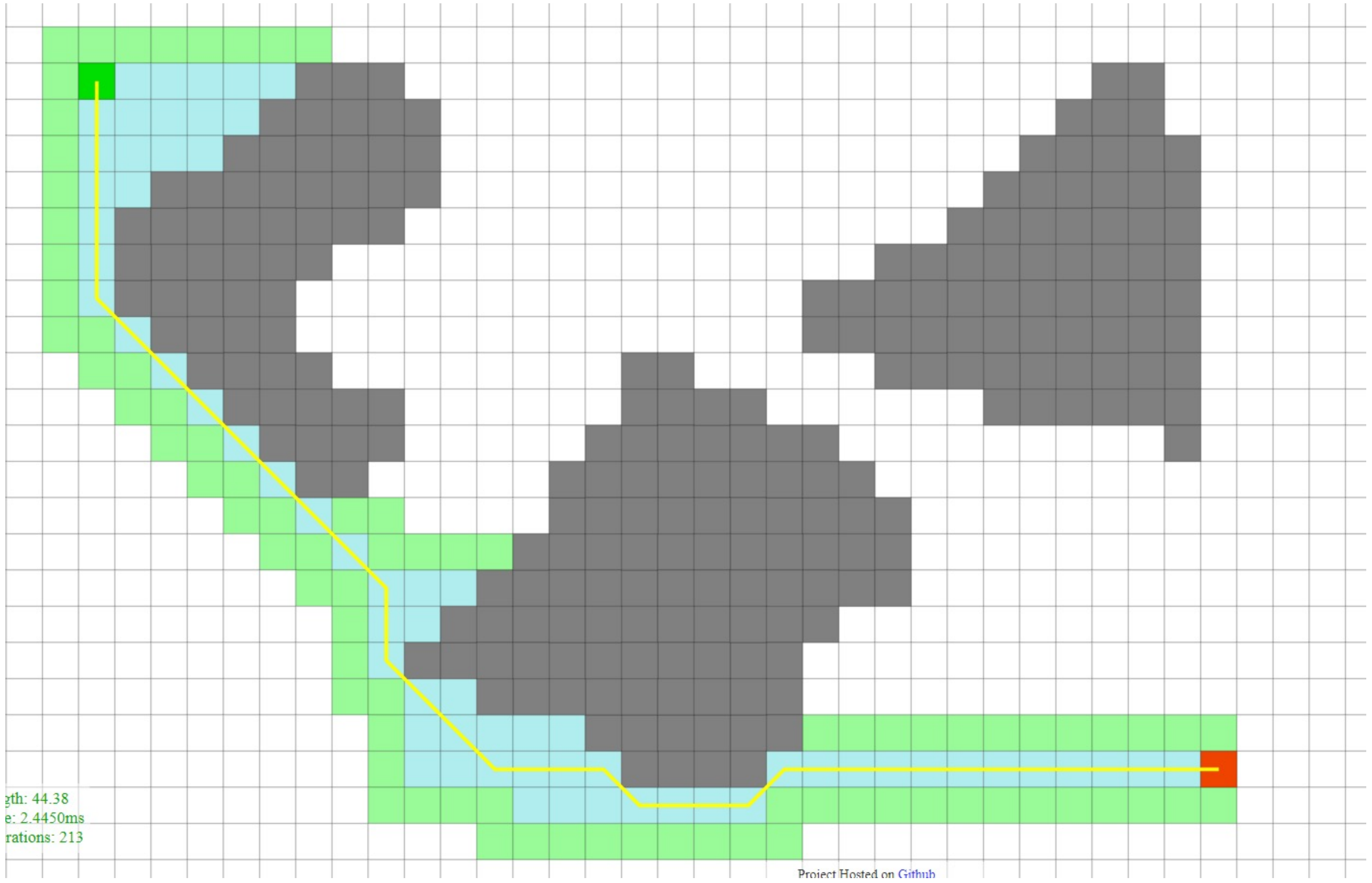
Cell decomposition path planning



From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

Approximate cell decomposition

This is what is usually done: path planning on a grid map



```
gth: 44.38
e: 2.4450ms
rations: 213
```

Project Hosted on Github

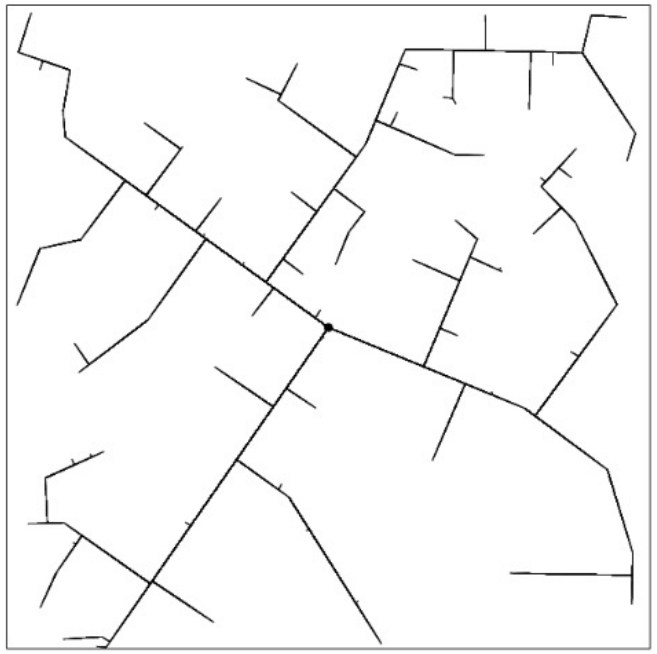
Here: A* for solving the search problem, Manhattan distance as h()

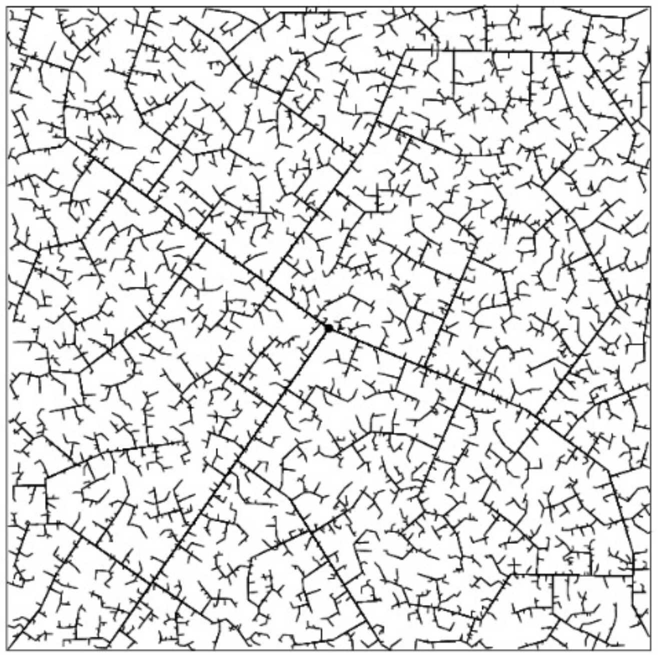
Path Planning Algorithms

- Path planning is usually modeled as a tree-search problem (examples here: https://qiao.github.io/PathFinding.js/visual/)

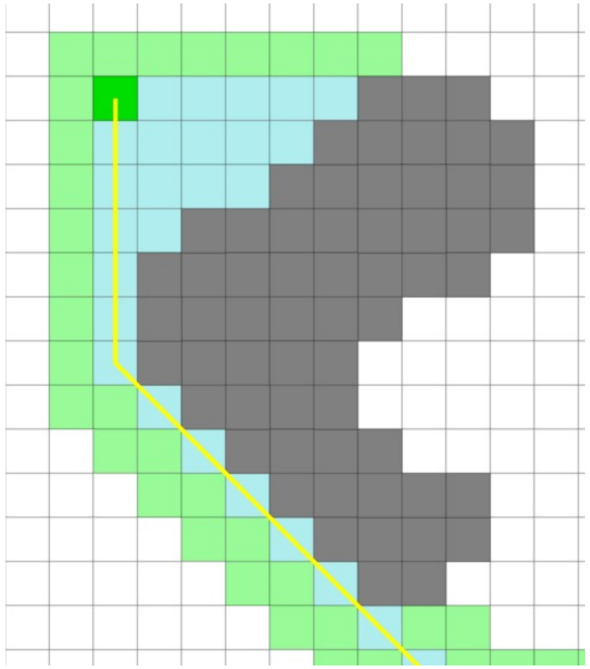
Examples of algorithms used :

- A* (more on this later in the course)
- RRT (Rapidly Exploring Random Trees) (image from LaValle)


45 iterations

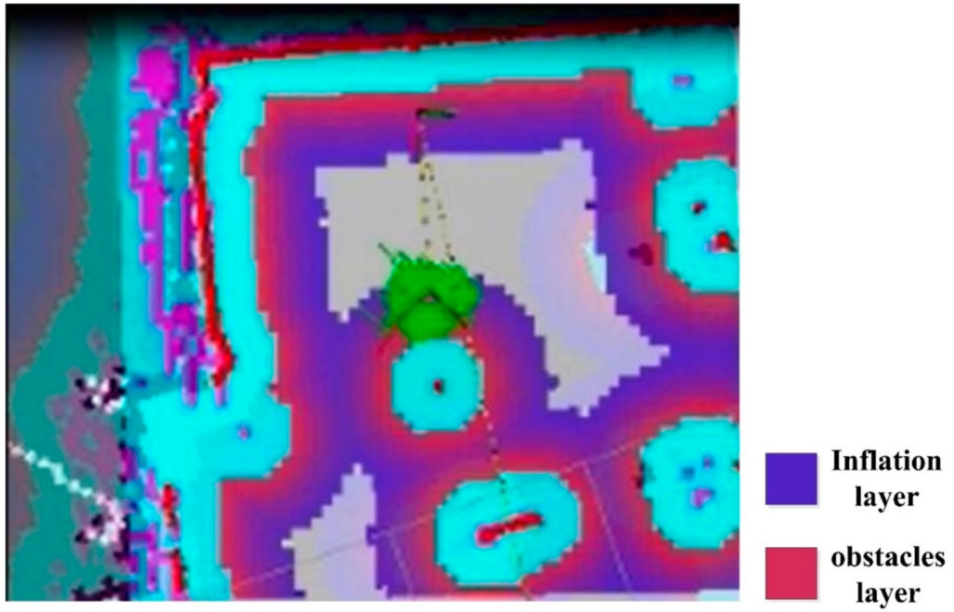

2345 iterations

Obstacle avoidance

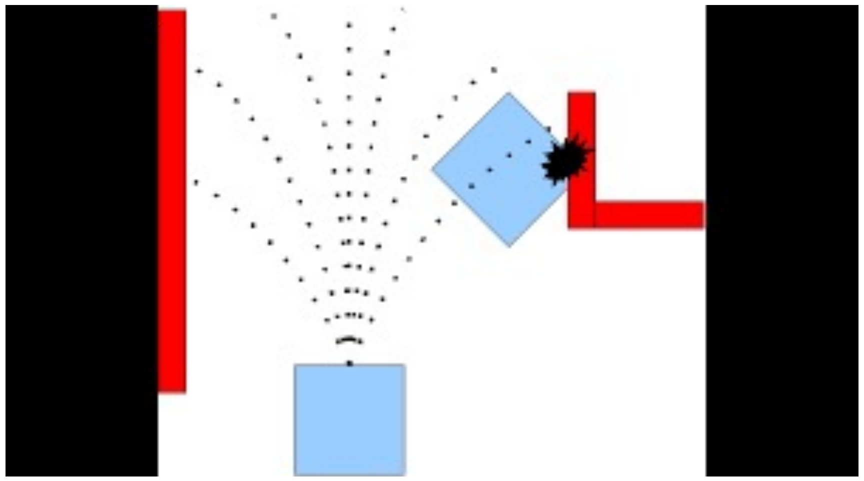


What happens if the robot is bigger than 1 cell (e.g., a 2x2 cell)? Shall we allow trajectories that are that close to the obstacles?

Several techniques are used for performing obstacle avoidance. Examples: inflating either the obstacle (considering the robot as a point) or the robot (allowing the robot to plan trajectories that goes across the obstacle.

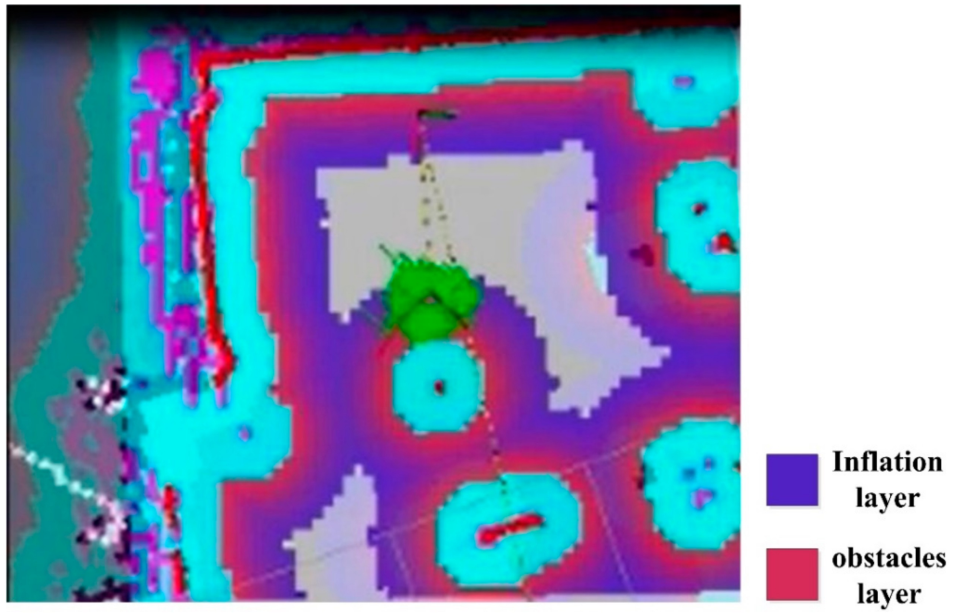


Obstacle avoidance



Usually a two-later architecture is used:

- Global path planning computes the trajectory towards the goal
- Local path planning executes locally the trajectory avoiding obstacles



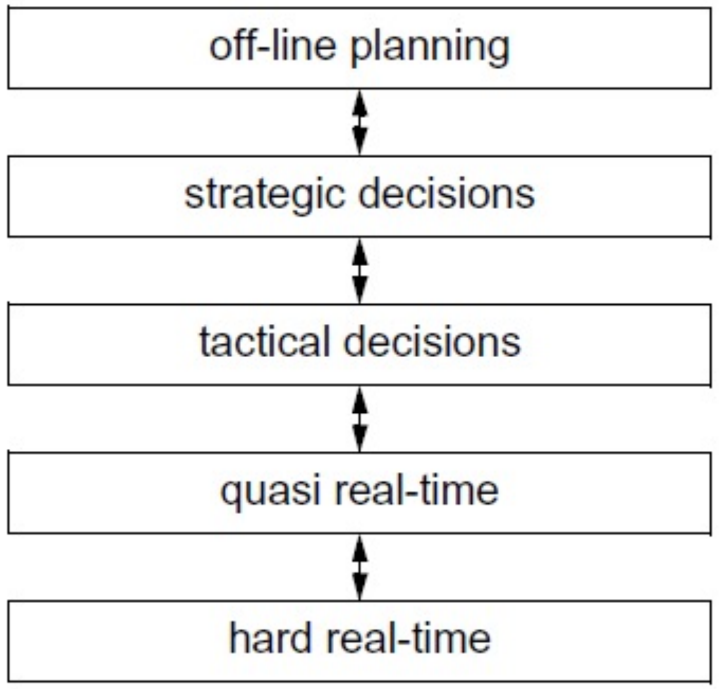
Local path planning uses:

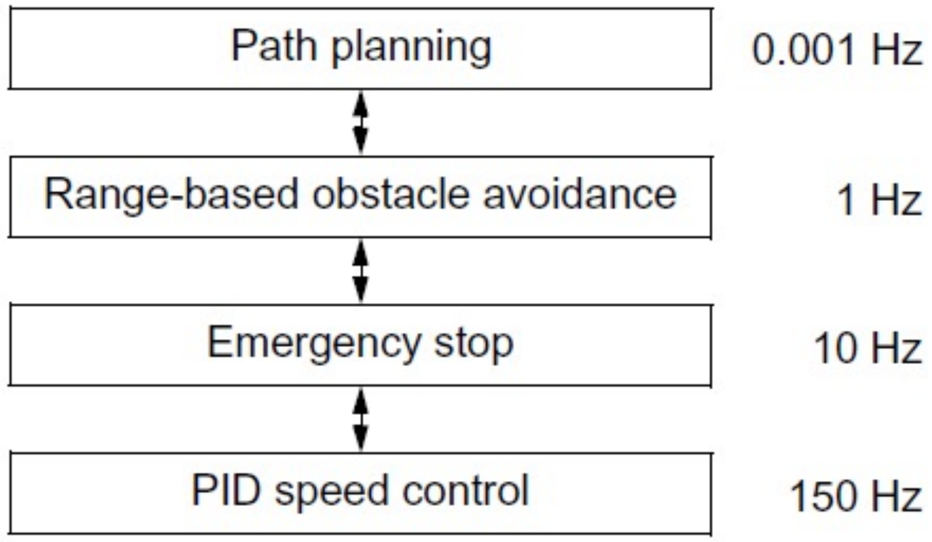
- Potentials fields
- Dynamic windows approaches

Navigation Architecture

Navigation is a task that requires both hi-level planning and low-level control, reacting to changes in the environment.

We can organize modules of the robot according to different hierarchies, as performing a temporal decomposition:



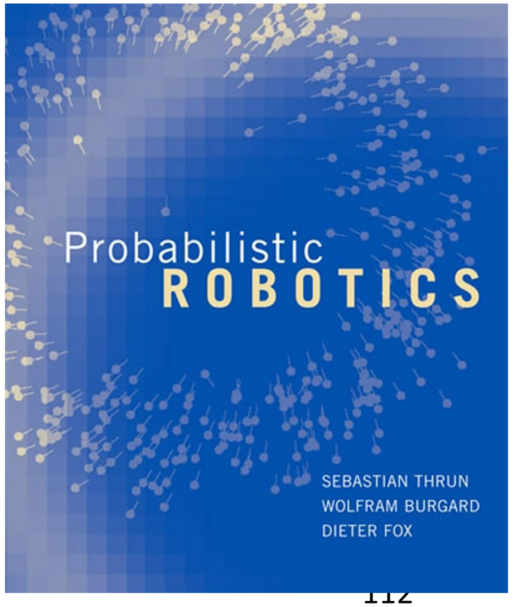


From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

Sources

Roland Siegwart, Illah Nourbakhsh,
Davide Scaramuzza
Introduction to Autonomous Mobile Robots, II ED,
MIT Press, 2011

Sebastian Thrun, Wolfram Burgard, Dieter Fox,
Probabilistic Robotics,
MIT Press, 2006



Sistemi Intelligenti Avanzati
Corso di Laurea in Informatica, A.A. 2021-2022
Università degli Studi di Milano

Matteo Luperto
Dipartimento di Informatica
matteo.luperto@unimi.it