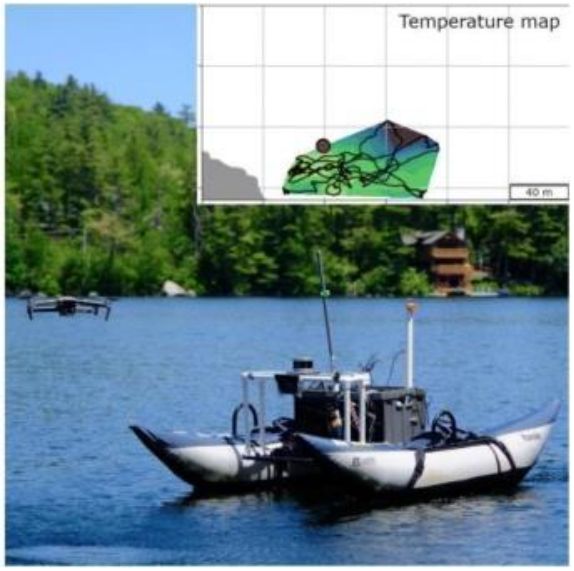
Sistemi Intelligenti Avanzati – 2025/2026
Università degli Studi di Milano

Introduction to Autonomous Mobile Robotics

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

Autonomous mobile robots



What defines an autonomous mobile robot?

- Its architecture / configuration
  - Wheeled or legged
  - Humanoid
  - Fling – UAV, fixed wing
  - Water – ASV, underwater
  - …
- Its environment
  - Indoor (house, office, logistic, hospitals)
  - Outdoor (Field, marine, flying, space)
- Its tasks
  - Assistive / Collaborative (cleaning)
  - Patrolling / Surveillance
  - Urban Search and Rescue
- Its interaction with humans
  - Autonomous vs semi-autonomous
- Multi-robot

Autonomous mobile robots

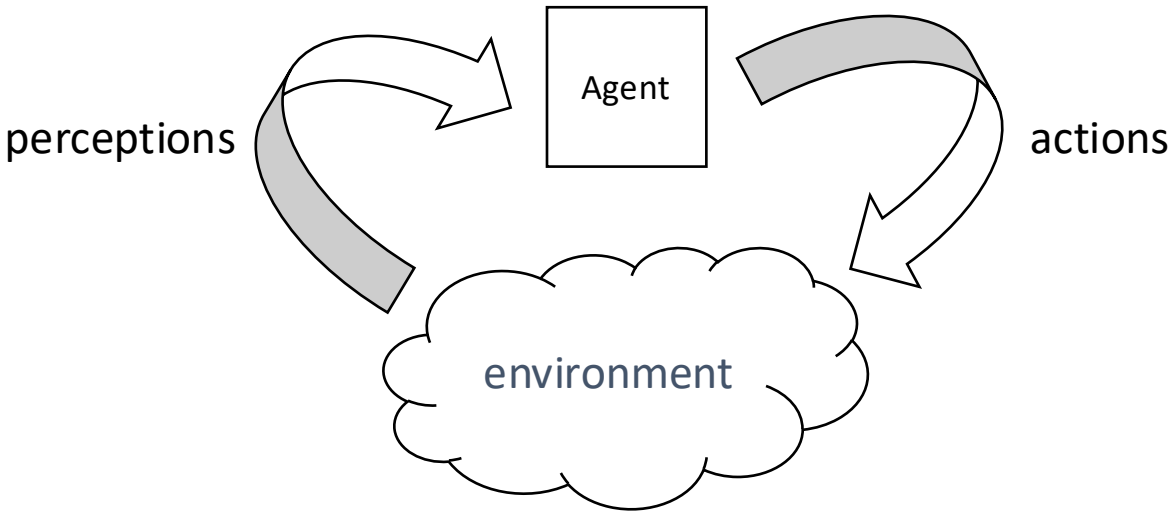
What is an autonomous mobile robot?

An agent that autonomously moves inside a given environment, to perform a given task

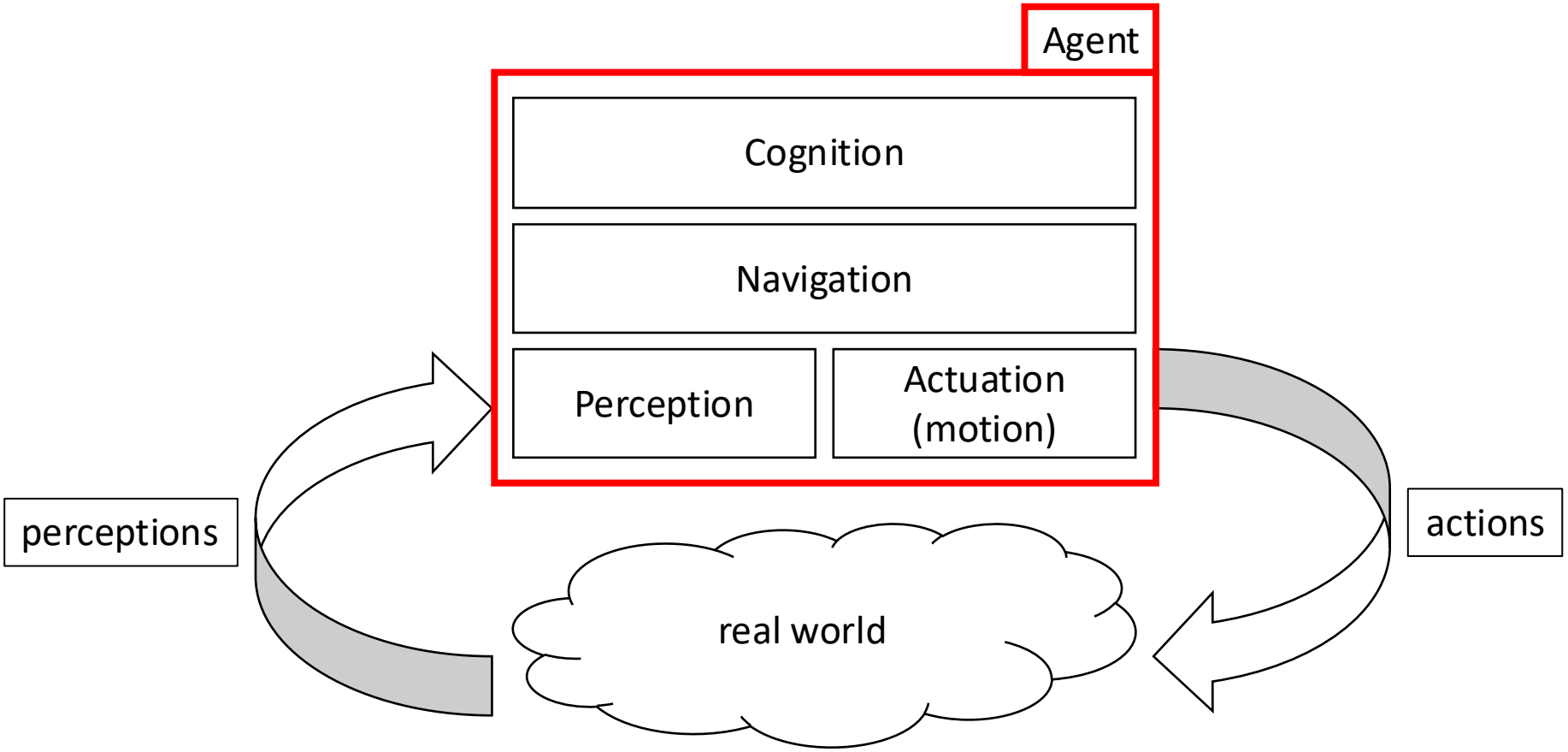
Robots as Agents

- *"[…] anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators." [Russel, Norvig 1995]*

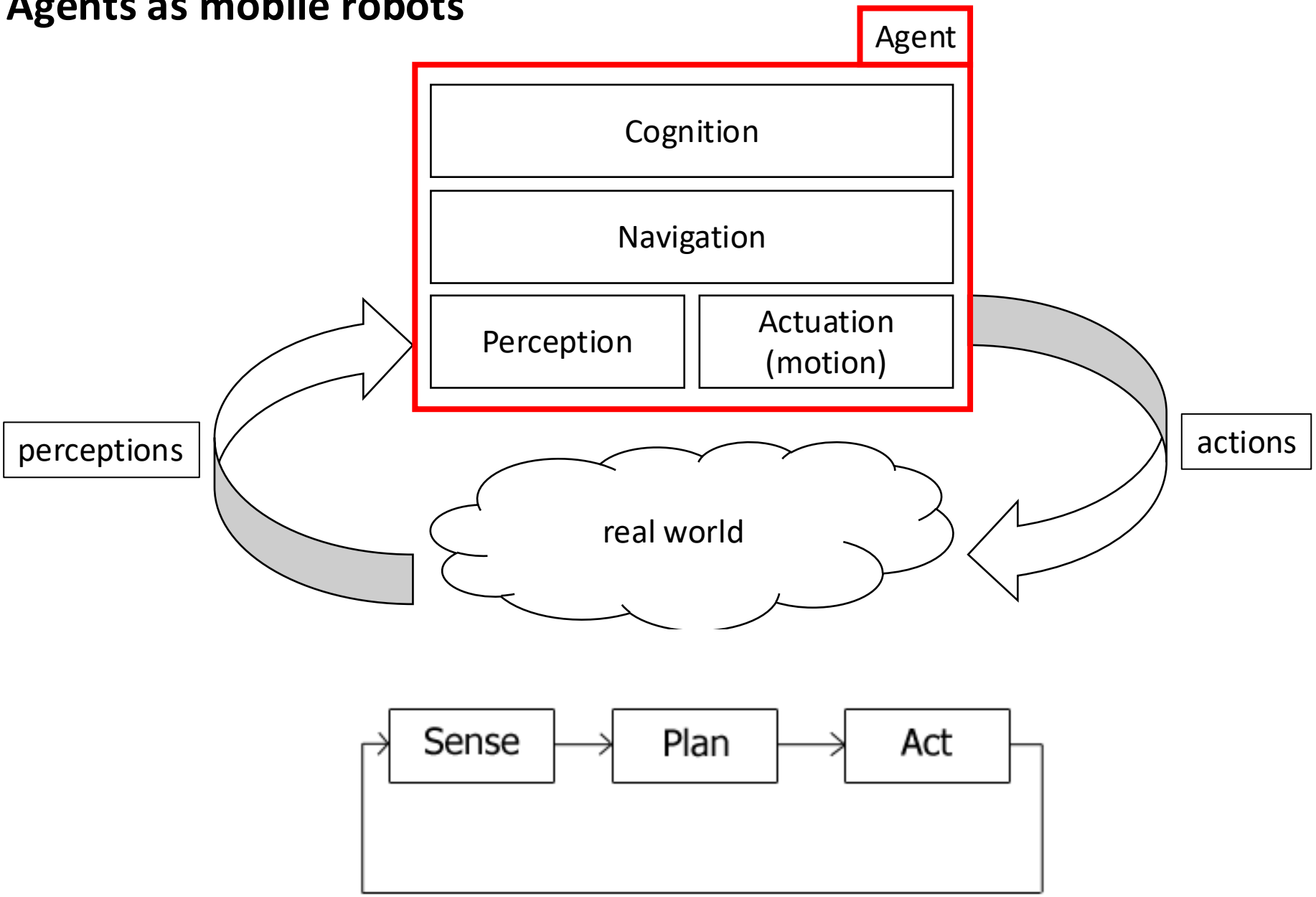
- *"[…] a computer system that is situated in some **environment**, and that is capable of **autonomous action** in this environment in order to meet its delegated objectives." [Wooldridge, 2009]*



Agents as mobile robots



Agents as mobile robots



Environments and tasks

What we want robot to do? tedious, boring, hazardous, costly tasks that we do not want to do (or to help us in doing so)

Environments and tasks

What we currently have are robots that can perform repetitive simple tasks into controlled environments (e.g., industrial robots).

What we want is a sci-fi general AI robot capable of interacting with us and adapt to new challenges and tasks

Environments and tasks for robots

Despite costs (still quite high) manipulators are “commonly” used in manufacturing, but for performing repetitive and preprogrammed tasks…

Environments and tasks for robots

Despite costs (still quite high) manipulators are “commonly” used in manufacturing, but for performing repetitive and preprogrammed tasks…

…however their generalization to different settings (e.g. logistics, small manufacturing, …)

Environments and tasks for robots

Domestic robots are slowly becoming a common item in our homes, but even in this case they have limited capabilities and they can perform only simple tasks (vacuum cleaners, lawnmowers, …)

Environments and tasks for robots

Autonomous driving cars are “almost” here, however:

- They can drive only “easy” contexts; it is easy to find their limits
- How to do the last mile towards *really* having autonomous road vehicles is still unknown
  (a lot of effort, and money, since 2010, no results)

autoblog

NEWS REVIEWS CAR BUYING CAR RESEARCH AND TOOLS DEALS MAINTENANCE FEATURES MAKES

News

Waymo Uses Remote Workers in the Philippines to Assist Its Self-Driving Cars

A recent Senate hearing grilled Waymo for using remote workers abroad to intervene when needed, raising security concerns

Feb 7, 2026 9:45 AM EST

By Anton Andres News Editor

1 Add us on G

Waymo

https://www.autoblog.com/news/waymo-uses-remote-workers-in-the-philippines-to-assist-its-self-driving-cars

Environments and tasks for robots

Broadly speaking: if we simplify the environment enough, and we simplify the robot's tasks enough, we can *have* autonomous robots…

…but there are still major limitations that prevents the widespread adoption of such machines.

(on the other side, general AI sci-fi robots are still sci-fi)

Limitations of Autonomous Robots

An agent that autonomously moves inside a given environment, to perform a given task

The major limitations of modern robots are due to the fact that robots need to make *decisions* to adapt their behaviour to the *environment* towards reaching theirs *tasks*…so where is the limitations?

- *Embodiment* = is it related to limitation in the robot HW?
- *Cognition* = is it related to limitation in the robot reasoning / SW?

What is missing to Autonomous Robots

[Pieter Abbeel, 2008]

Limitations of Autonomous Robots

An agent that autonomously moves inside a given environment, to perform a given task

It seems that, while we still have major limitations in terms of robots' actuation (wheels, arms, grippers) sensorial perception (sensors), and computational power (CPU/GPU,Memory), the main limitation is still related to their cognition level, i.e., how to make decisions.

Limitations of Autonomous Robots

An agent that autonomously moves inside a given environment, to perform a given task

If we have to pick one major issues about modern autonomous robots, the main one is *perception*, as it involves the *interpretation* of sensed data in a meaningful way.

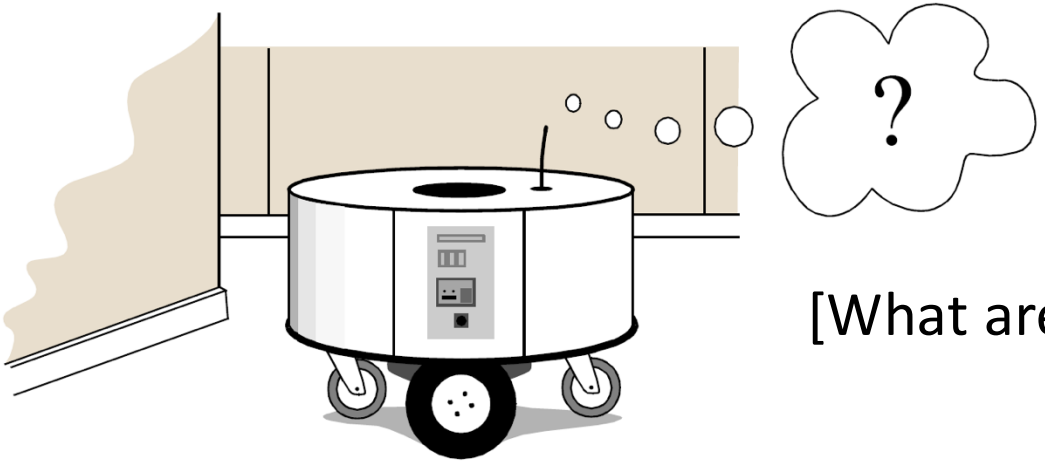
Thus, *mobility* is a critical aspects as it depends on perception and interpretation (while, manipulators, have less strict requirements)

Towards Autonomous Robots

[Where to go next?]

[What is that?]

[Where am I?]



[What's my battery status?]

[What are my tasks?]

[How do I go there?]

[What task is next?]

[What plan to do all my tasks for today?]

[How to accomplish this task?]

An autonomous mobile robot needs to solve different concurrent tasks

Towards Autonomous Robots

[Where am I?]

[Where to go next?]

[How do I go there?]

[How to accomplish this task?]

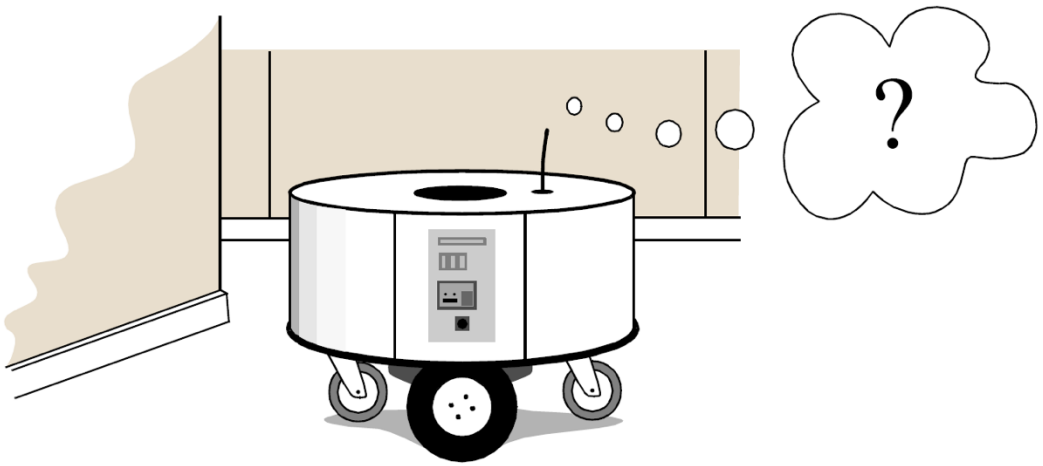
[What task is next?]

[What plan to do all my tasks for today?]

[What are my tasks?]

[What's my battery status?]

[What is that?]



Divide et impera: divide robot functionalities in sub-problems, organize them at different level of abstraction, solve them separately, integrate

Towards Autonomous Robots

[Where am I?] [What's my battery status?] [What is that?] →

[Where to go next?]

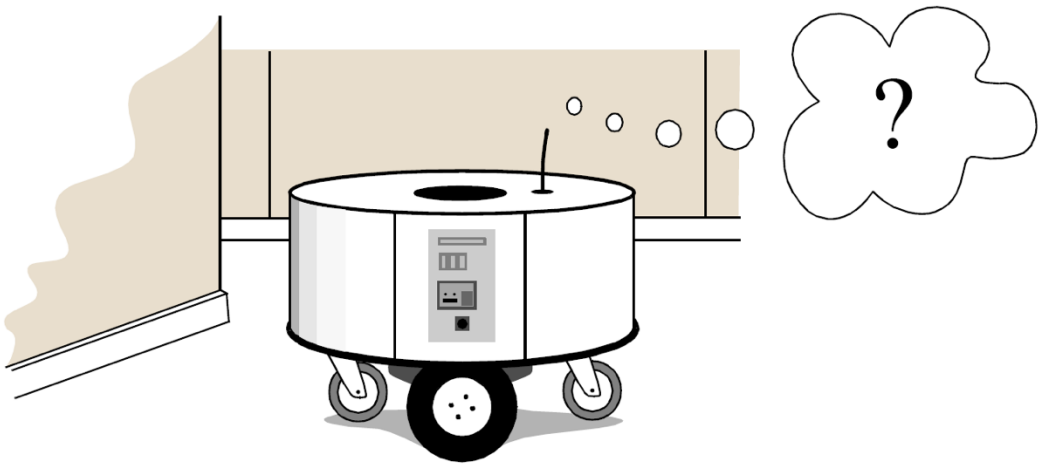
[How do I go there?]

[How to accomplish this task?]

[What task is next?]

[What plan to do all my tasks for today?]

[What are my tasks?]



Adapt the execution to environmental changes, unexpected events, make robust solutions (e.g., self-driving cars)

Towards Autonomous Robots

[Where am I?]

[Where to go next?]

[How do I go there?]

Navigation and mapping and their subproblems:

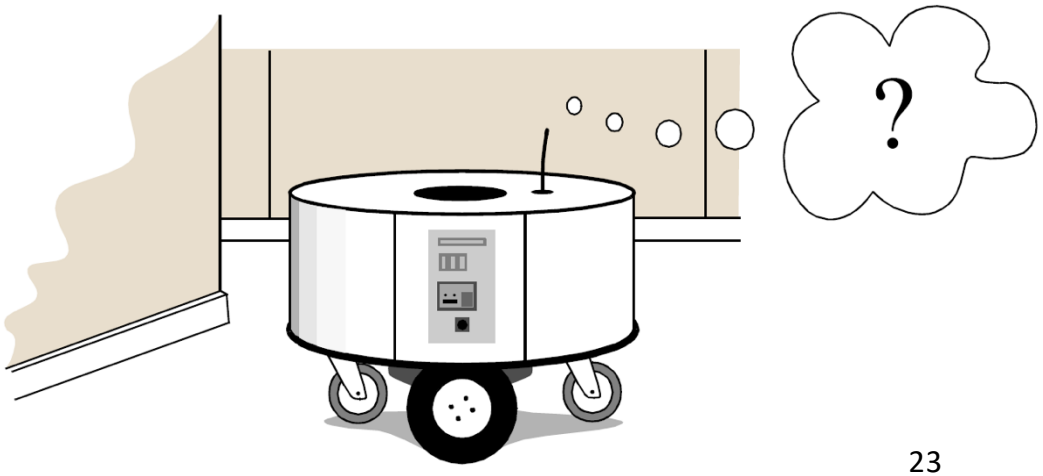
Motion, mapping, localization, path planning, path execution

[How to accomplish this task?]

[What task is next?]

[What plan to do all my tasks for today?]

[What are my tasks?]



Towards Autonomous Robots

[Where am I?]

[Where to go next?]

[How do I go there?]

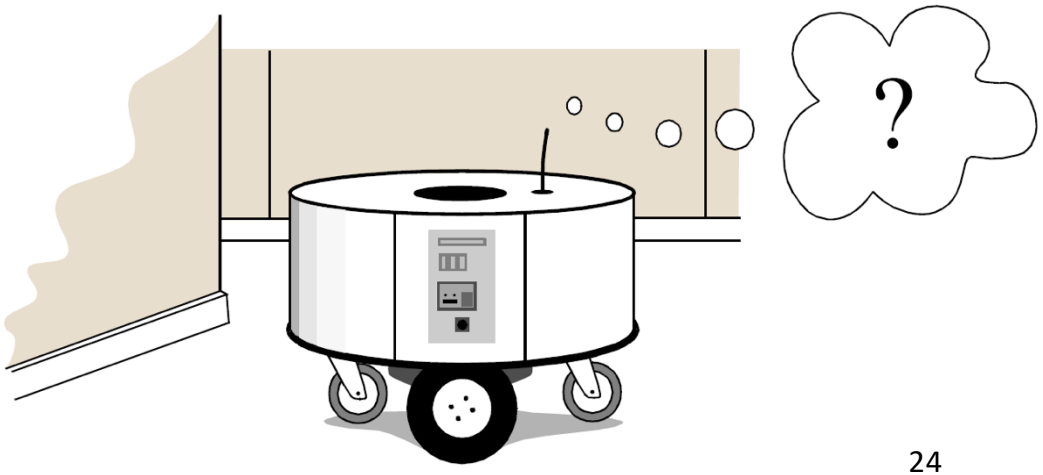
[How to accomplish this task?]

[What task is next?]

[What plan to do all my tasks for today?]

[What are my tasks?]

Task-related problems: manipulation, grasping, human-robot interaction, cleaning, patrolling, …



Towards Autonomous Robots

[Where am I?]

[Where to go next?]

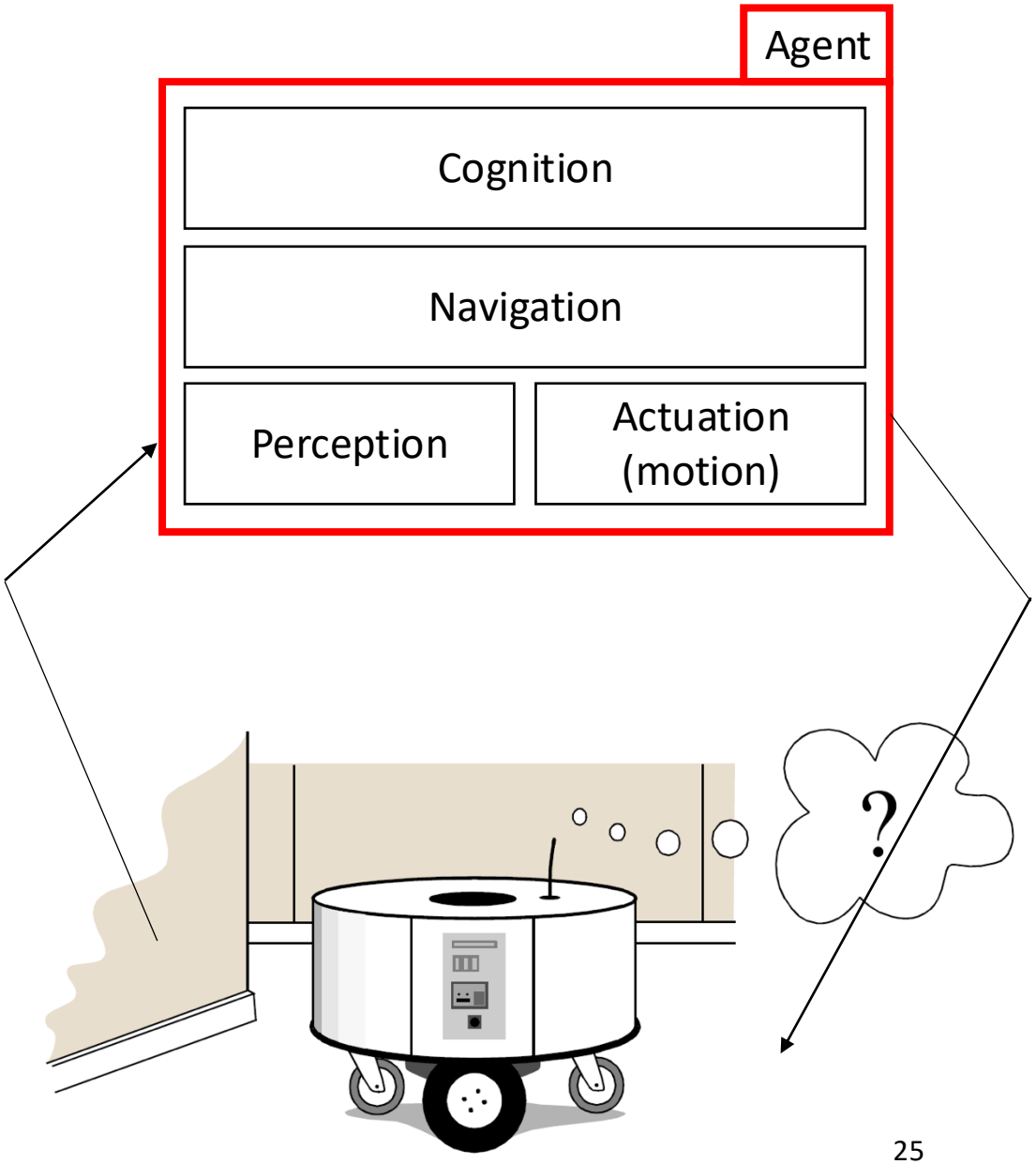
[How do I go there?]

[How to accomplish this task?]

[What task is next?]

[What plan to do all my tasks for today?]

[What are my tasks?]



Towards Autonomous Robots

[Where am I?]

[Where to go next?]

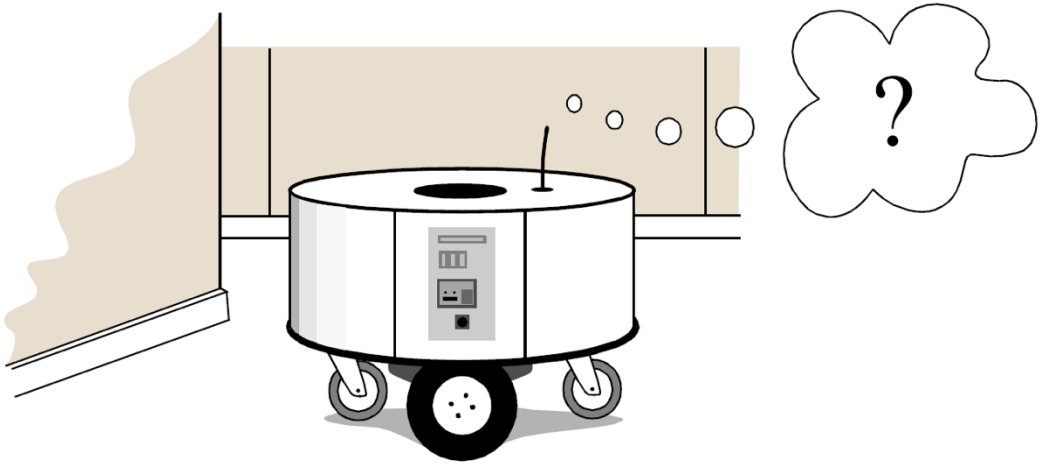
[How do I go there?]

[How to accomplish this task?]

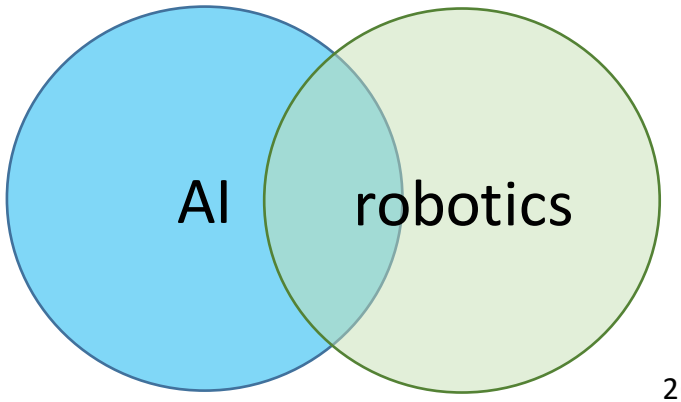
[What task is next?]

[What plan to do all my tasks for today?]

[What are my tasks?]



Planning problems, AI for robotics



Different robots, different level of complexity

[Kunze et al., 2018]

SURVEYED AI-ENABLED LONG-TERM AUTONOMY ROBOT SYSTEMS.

Domain	Application Features								Duration	AI Areas						System
	Environment Variability	Task Diversity	Semantics	Dynamics	Partial Observability	Cost & Criticality	Interaction & Cooperation	Level of Autonomy		Navigation & Mapping	Perception	KR & Reasoning	Planning	Interaction	Learning	
Space	L	L	L	L	H	H	L	M	Years	○	●	–	●	○	–	Opportunity [9], [10]
									Years	○	●	–	●	○	–	IPEX [11]
Marine	M	L	L	M	H	H	L	H	Days	○	●	○	●	–	○	AUVs [12], [13]
									Months	○	○	–	○	–	–	Gliders [14]
Air	M	M	M	H	H	H	M	M	Days	○	●	○	○	–	–	AtlantikFlyer [15]
Field	H	M	L	M	H	M	M	M	Days	●	●	○	–	○	○	VT&R2 [16]
									Years	●	●	○	–	–	○	BearNav [17], [18]
Road	M	L	M	H	M	H	M	L	Days	○	●	●	○	–	○	VaMP [19]
									Days	○	●	○	○	–	○	ARGO [20]
									Months	○	●	○	○	–	○	PANS [21]
									Months	○	●	○	○	–	○	VIAC [22]
Service	H	H	H	L	H	L	H	M	Days	●	○	○	●	●	○	Rhino [23]
									Days	●	○	○	●	●	○	Minerva [24]
									Days	●	○	○	○	●	○	Willow Garage [25]
									Months	●	●	●	●	●	●	STRANDS [26]
									Years	●	●	●	●	●	●	CoBot [27]

Legend: L *low*, M *medium*, H *high*, – *not integrated*, ○ *partially integrated*, ● *fully integrated*

Different robots, different level of complexity

[Kunze et al., 2018]

SURVEYED AI-ENABLED LONG-TERM AUTONOMY ROBOT SYSTEMS.

Domain	Application Features								Duration	AI Areas						System
	Environment Variability	Task Diversity	Semantics	Dynamics	Partial Observability	Cost & Criticality	Interaction & Cooperation	Level of Autonomy		Navigation & Mapping	Perception	KR & Reasoning	Planning	Interaction	Learning	
Space	L	L	L	L	H	H	L	M	Years	○	●	–	●	○	–	Opportunity [9], [10]
									Years	○	●	–	●	○	–	IPEX [11]
Marine	M	L	L	M	H	H	L	H	Days	○	●	○	●	–	○	AUVs [12], [13]
									Months	○	○	–	○	–	–	Gliders [14]
Air	M	M	M	H	H	H	M	M	Days	○	●	○	○	–	–	AtlantikFlyer [15]
Field	H	M	L	M	H	M	M	M	Days	●	●	○	–	○	○	VT&R2 [16]
									Years	●	●	○	–	–	○	BearNav [17], [18]
Road	M	L	M	H	M	H	M	L	Days	○	●	●	○	–	○	VaMP [19]
									Days	○	●	○	○	–	○	ARGO [20]
									Months	○	●	○	○	–	○	PANS [21]
									Months	○	●	○	○	–	○	VIAC [22]
Service	H	H	H	L	H	L	H	M	Days	●	○	○	●	●	○	Rhino [23]
									Days	●	○	○	●	●	○	Minerva [24]
									Days	●	○	○	○	●	○	Willow Garage [25]
									Months	●	●	●	●	●	●	STRANDS [26]
									Years	●	●	●	●	●	●	CoBot [27]

Legend: L *low*, M *medium*, H *high*, – *not integrated*, ○ *partially integrated*, ● *fully integrated*

Different robots, different level of complexity

[Kunze et al., 2018]

SURVEYED AI-ENABLED LONG-TERM AUTONOMY ROBOT SYSTEMS.

Domain	Application Features								Duration	AI Areas						System
	Environment Variability	Task Diversity	Semantics	Dynamics	Partial Observability	Cost & Criticality	Interaction & Cooperation	Level of Autonomy		Navigation & Mapping	Perception	KR & Reasoning	Planning	Interaction	Learning	
Space	L	L	L	L	H	H	L	M	Years	○	●	–	●	○	–	Opportunity [9], [10]
									Years	○	●	–	●	○	–	IPEX [11]
Marine	M	L	L	M	H	H	L	H	Days	○	●	○	●	–	○	AUVs [12], [13]
									Months	○	○	–	○	–	–	Gliders [14]
Air	M	M	M	H	H	H	M	M	Days	○	●	○	○	–	–	AtlantikFlyer [15]
Field	H	M	L	M	H	M	M	M	Days	●	●	○	–	○	○	VT&R2 [16]
									Years	●	●	○	–	–	○	BearNav [17], [18]
Road	M	L	M	H	M	H	M	L	Days	○	●	●	○	–	○	VaMP [19]
									Days	○	●	○	○	–	○	ARGO [20]
									Months	○	●	○	○	–	○	PANS [21]
									Months	○	●	○	○	–	○	VIAC [22]
Service	H	H	H	L	H	L	H	M	Days	●	○	○	●	●	○	Rhino [23]
									Days	●	○	○	●	●	○	Minerva [24]
									Days	●	○	○	○	●	○	Willow Garage [25]
									Months	●	●	●	●	●	●	STRANDS [26]
									Years	●	●	●	●	●	●	CoBot [27]

Legend: L *low*, M *medium*, H *high*, – *not integrated*, ○ *partially integrated*, ● *fully integrated*

Autonomous Robots at large

Mechanics

Kinematics

Control Theory

Signal Processing

Information Theory

Probability Theory

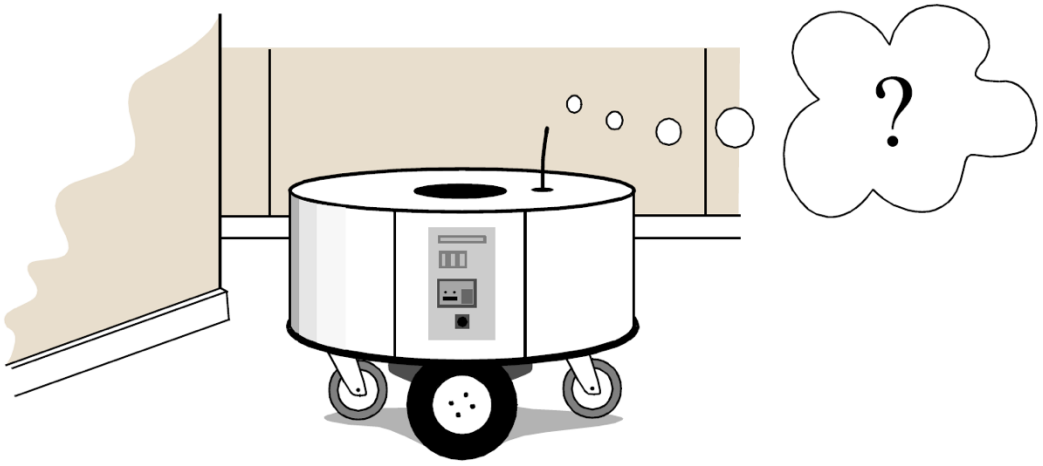
Artificial Intelligence

Computer Vision

Multi Agent Systems

Multiple perspectives and fields involved, from HW to SW

There is no single solution on how to address this problem
(robotics is still a young field)

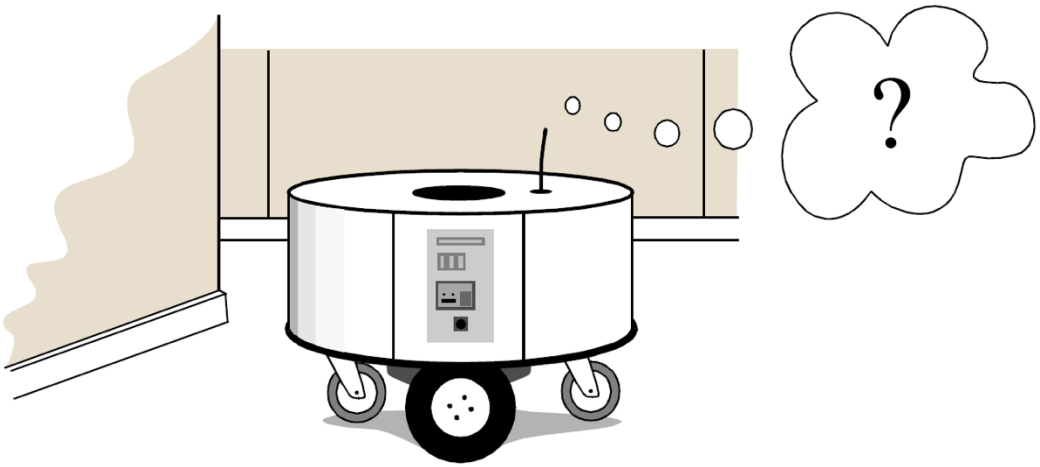


Focus of this lessons: why wheeled ground robots

Classic navigation approaches

Overview of core concepts:

- Robot Motion
- Perception
- Localization and Mapping
- Navigation

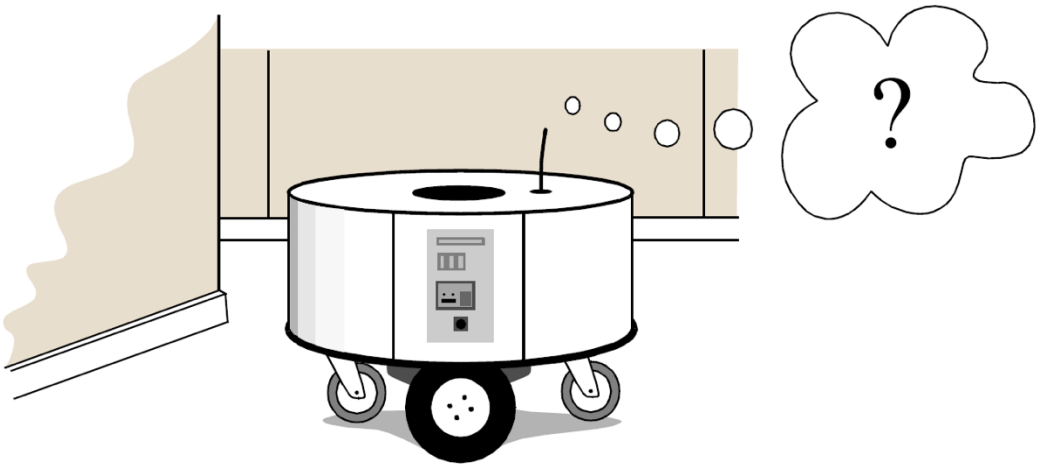


Assumption: let's talk about the simplest type of mobile robots, wheeled ground vehicles

Classic navigation approaches

Overview of core concepts:

- Robot Motion
- Perception
- Localization and Mapping
- Navigation



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

Robot Motion

Locomotion

Wheels

Configuration

Kinematics

Robot Wheels

1

2

3

4

Four main types of wheels:

1. Standard wheel - 2 DOF - rotation around the wheel axle
2. Castor wheel – 2 DOF – rotation around the steering joint
3. Mecanum wheel (Swedish or Omni Wheel) – 3DOF –
   rotation around wheel axle, rollers, contact point, 45° or 90°
4. Ball or Spherical Wheel

Mecanum wheel = omnidirectional

Wheel Configuration

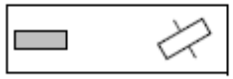
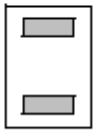
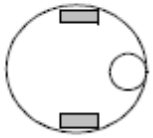
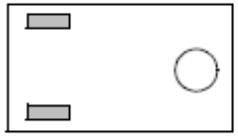
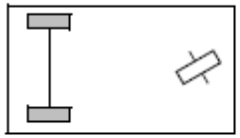
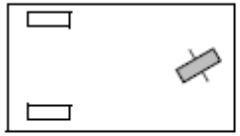
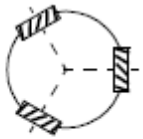
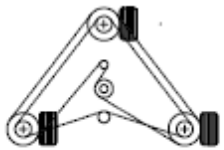
- How many wheels? 2,4,6,8?
- How many axes?
- What type of wheels?

Targets:

- Stability = robot does not fall → 2 wheels minimum, 3+ for “robust” solutions
- Maneuverability = do we have motion constraints? (e.g., car in parallel parking)
- Controllability = how difficult is to control movement?

Usually, maneuverability and controllability are inversely correlated

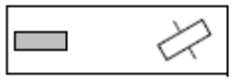
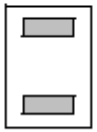
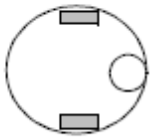
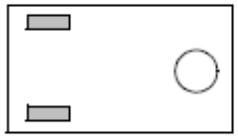
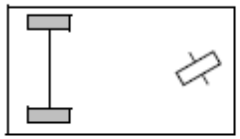
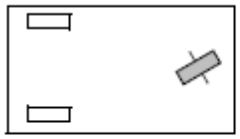
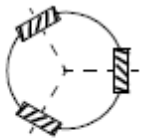
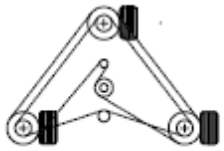
Wheel Configuration

# of wheels	Arrangement	Description	Typical examples
2		One steering wheel in the front, one traction wheel in the rear	Bicycle, motorcycle
		Two-wheel differential drive with the center of mass (COM) below the axle	Cye personal robot
3		Two-wheel centered differential drive with a third point of contact	Nomad Scout, smartRob EPFL
		Two independently driven wheels in the rear/front, 1 unpowered omnidirectional wheel in the front/rear	Many indoor robots, including the EPFL robots Pygmalion and Alice
		Two connected traction wheels (differential) in rear, 1 steered free wheel in front	Piaggio minitrucks
		Two free wheels in rear, 1 steered traction wheel in front	Neptune (Carnegie Mellon University), Hero-1
		Three motorized Swedish or spherical wheels arranged in a triangle; omnidirectional movement is possible	Stanford wheel Tribolo EPFL, Palm Pilot Robot Kit (CMU)
		Three synchronously motorized and steered wheels; the orientation is not controllable	"Synchro drive" Denning MRV-2, Georgia Institute of Technology, I-Robot B24, Nomad 200

Icons for the each wheel type are as follows:	
	unpowered omnidirectional wheel (spherical, castor, Swedish);
	motorized Swedish wheel (Stanford wheel);
	unpowered standard wheel;
	motorized standard wheel;
	motorized and steered castor wheel;
	steered standard wheel;
	connected wheels.

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

Wheel Configuration

# of wheels	Arrangement	Description	Typical examples
2		One steering wheel in the front, one traction wheel in the rear	Bicycle, motorcycle
		Two-wheel differential drive with the center of mass (COM) below the axle	Cye personal robot
3		Two-wheel centered differential drive with a third point of contact	Nomad Scout, smartRob EPFL
		Two independently driven wheels in the rear/front, 1 unpowered omnidirectional wheel in the front/rear	Many indoor robots, including the EPFL robots Pygmalion and Alice
		Two connected traction wheels (differential) in rear, 1 steered free wheel in front	Piaggio minitrucks
		Two free wheels in rear, 1 steered traction wheel in front	Neptune (Carnegie Mellon University), Hero-1
		Three motorized Swedish or spherical wheels arranged in a triangle; omnidirectional movement is possible	Stanford wheel Tribolo EPFL, Palm Pilot Robot Kit (CMU)
		Three synchronously motorized and steered wheels; the orientation is not controllable	"Synchro drive" Denning MRV-2, Georgia Institute of Technology, I-Robot B24, Nomad 200

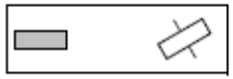
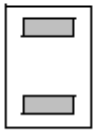
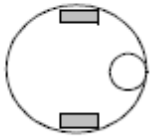
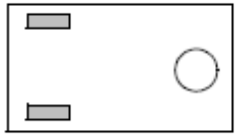
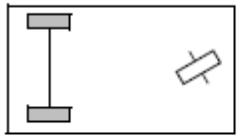
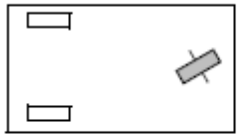
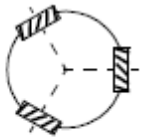
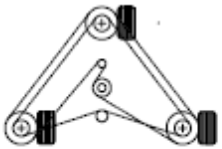
Icons for the each wheel type are as follows:	
	unpowered omnidirectional wheel (spherical, castor, Swedish);
	motorized Swedish wheel (Stanford wheel);
	unpowered standard wheel;
	motorized standard wheel;
	motorized and steered castor wheel;
	steered standard wheel;
	connected wheels.

Popular configurations:

- Limited number of wheels
- Limited motors
- Simple

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

Wheel Configuration

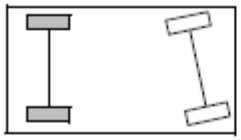
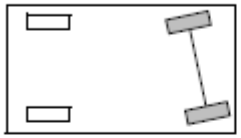
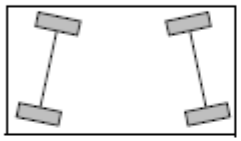
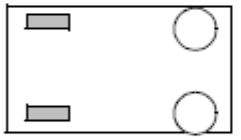
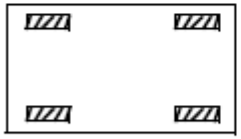
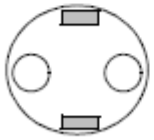
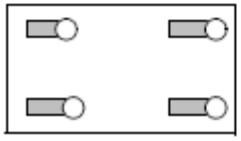
# of wheels	Arrangement	Description	Typical examples
2		One steering wheel in the front, one traction wheel in the rear	Bicycle, motorcycle
		Two-wheel differential drive with the center of mass (COM) below the axle	Cye personal robot
3		Two-wheel centered differential drive with a third point of contact	Nomad Scout, smartRob EPFL
		Two independently driven wheels in the rear/front, 1 unpowered omnidirectional wheel in the front/rear	Many indoor robots, including the EPFL robots Pygmalion and Alice
		Two connected traction wheels (differential) in rear, 1 steered free wheel in front	Piaggio minitrucks
		Two free wheels in rear, 1 steered traction wheel in front	Neptune (Carnegie Mellon University), Hero-1
		Three motorized Swedish or spherical wheels arranged in a triangle; omnidirectional movement is possible	Stanford wheel Tribolo EPFL, Palm Pilot Robot Kit (CMU)
		Three synchronously motorized and steered wheels; the orientation is not controllable	"Synchro drive" Denning MRV-2, Georgia Institute of Technology, I-Robot B24, Nomad 200

Icons for the each wheel type are as follows:	
	unpowered omnidirectional wheel (spherical, castor, Swedish);
	motorized Swedish wheel (Stanford wheel);
	unpowered standard wheel;
	motorized standard wheel;
	motorized and steered castor wheel;
	steered standard wheel;
	connected wheels.

Omnidirectional with 3 motors and a simple architecture

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

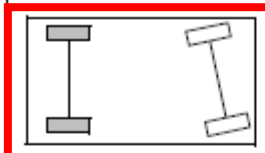
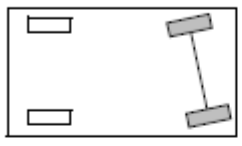
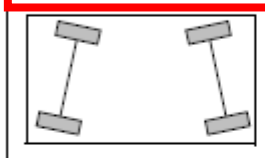
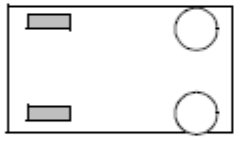
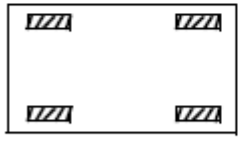
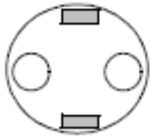
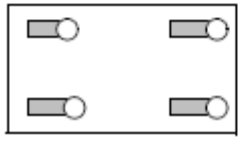
Wheel Configuration

# of wheels	Arrangement	Description	Typical examples
4		Two motorized wheels in the rear, 2 steered wheels in the front; steering has to be different for the 2 wheels to avoid slipping/skidding.	Car with rear-wheel drive
		Two motorized and steered wheels in the front, 2 free wheels in the rear; steering has to be different for the 2 wheels to avoid slipping/skidding.	Car with front-wheel drive
		Four steered and motorized wheels	Four-wheel drive, four-wheel steering Hyperion (CMU)
		Two traction wheels (differential) in rear/front, 2 omnidirectional wheels in the front/rear	Charlie (DMT-EPFL)
		Four omnidirectional wheels	Carnegie Mellon Uranus
		Two-wheel differential drive with 2 additional points of contact	EPFL Khepera, Hyperbot Chip
		Four motorized and steered castor wheels	Nomad XR4000

Icons for the each wheel type are as follows:	
	unpowered omnidirectional wheel (spherical, castor, Swedish);
	motorized Swedish wheel (Stanford wheel);
	unpowered standard wheel;
	motorized standard wheel;
	motorized and steered castor wheel;
	steered standard wheel;
	connected wheels.

From [Siegwart, Introduction to Autonomous Mobile Robots]

Wheel Configuration

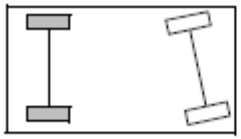
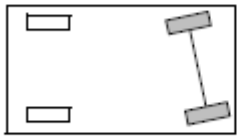
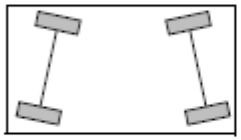
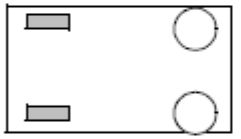
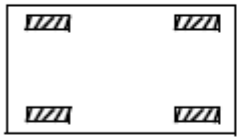
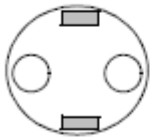
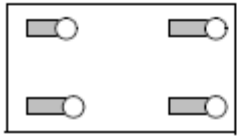
# of wheels	Arrangement	Description	Typical examples
4		Two motorized wheels in the rear, 2 steered wheels in the front; steering has to be different for the 2 wheels to avoid slipping/skidding.	Car with rear-wheel drive
		Two motorized and steered wheels in the front, 2 free wheels in the rear; steering has to be different for the 2 wheels to avoid slipping/skidding.	Car with front-wheel drive
		Four steered and motorized wheels	Four-wheel drive, four-wheel steering Hyperion (CMU)
		Two traction wheels (differential) in rear/front, 2 omnidirectional wheels in the front/rear	Charlie (DMT-EPFL)
		Four omnidirectional wheels	Carnegie Mellon Uranus
		Two-wheel differential drive with 2 additional points of contact	EPFL Khepera, Hyperbot Chip
		Four motorized and steered castor wheels	Nomad XR4000

From [Siegwart, Introduction to Autonomous Mobile Robots]

Icons for the each wheel type are as follows:	
	unpowered omnidirectional wheel (spherical, castor, Swedish);
	motorized Swedish wheel (Stanford wheel);
	unpowered standard wheel;
	motorized standard wheel;
	motorized and steered castor wheel;
	steered standard wheel;
	connected wheels.

Car configuration – parallel parking

Wheel Configuration

# of wheels	Arrangement	Description	Typical examples
4		Two motorized wheels in the rear, 2 steered wheels in the front; steering has to be different for the 2 wheels to avoid slipping/skidding.	Car with rear-wheel drive
		Two motorized and steered wheels in the front, 2 free wheels in the rear; steering has to be different for the 2 wheels to avoid slipping/skidding.	Car with front-wheel drive
		Four steered and motorized wheels	Four-wheel drive, four-wheel steering Hyperion (CMU)
		Two traction wheels (differential) in rear/front, 2 omnidirectional wheels in the front/rear	Charlie (DMT-EPFL)
		Four omnidirectional wheels	Carnegie Mellon Uranus
		Two-wheel differential drive with 2 additional points of contact	EPFL Khepera, Hyperbot Chip
		Four motorized and steered castor wheels	Nomad XR4000

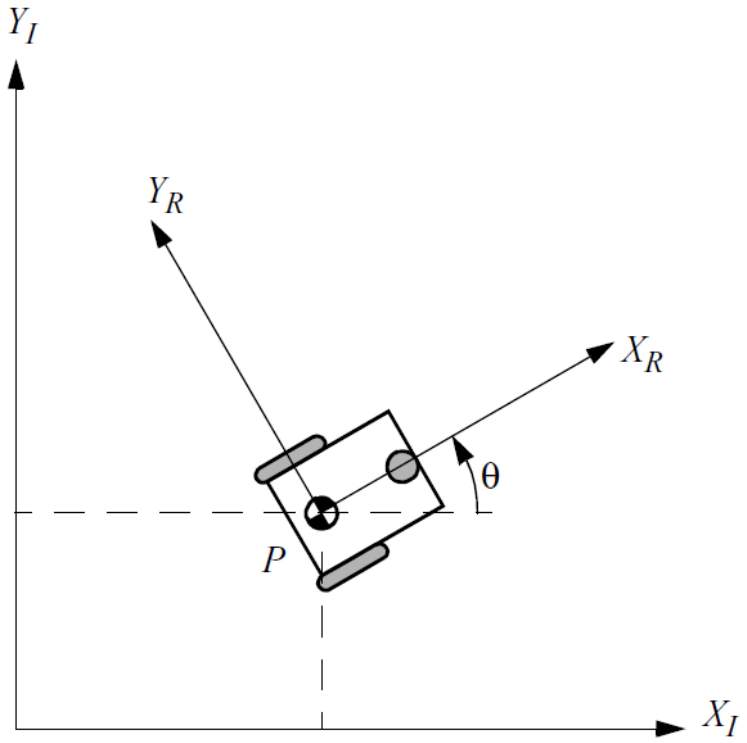
Icons for the each wheel type are as follows:	
	unpowered omnidirectional wheel (spherical, castor, Swedish);
	motorized Swedish wheel (Stanford wheel);
	unpowered standard wheel;
	motorized standard wheel;
	motorized and steered castor wheel;
	steered standard wheel;
	connected wheels.

Omnidirectional – 4 wheels

From [Siegwart, Introduction to Autonomous Mobile Robots]

Kinematics

- Describe how a mechanical system behaves, is needed to create control software for the robot
- Kinematic Model of the robot and Constraints
  - Representing the robot position and the robot movement in a global and local reference frame



The robot pose is expressed as

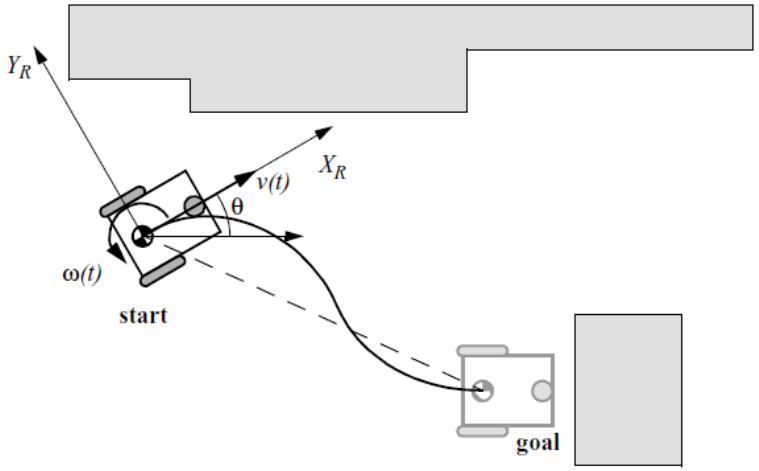
$$[x, y, \theta]$$

in the global reference frame

For a 3D robot, add axis z and roll pitch yaw

Kinematics

- Forward Kinematics computes the robot speed in the global reference frame given the spinning speed of each wheel
- Inverse Kinematics compute the robot actuators parameters to reach a given configuration
- Each wheel configuration results into a set of constraints



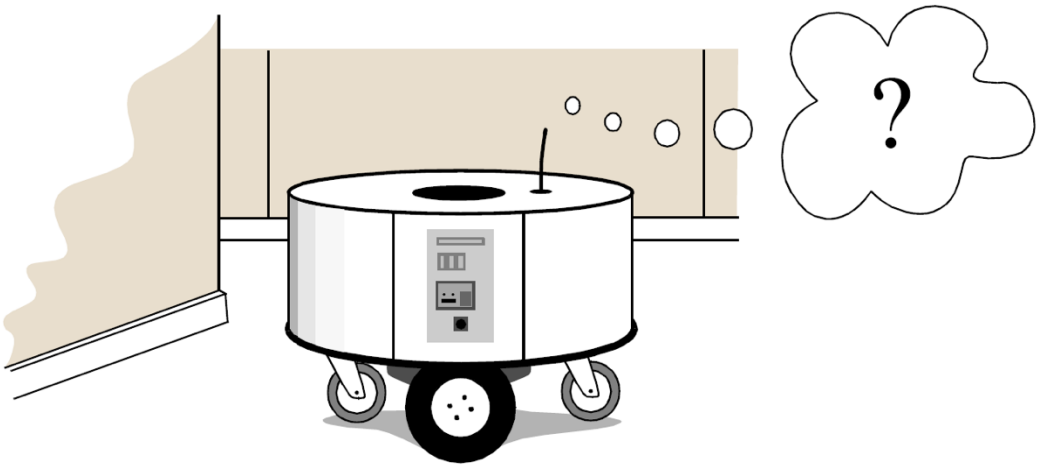
Usually, robot DDOF are considered: Differential Degrees of Freedom (that are equal to the degree of mobility of the robot)

$$DDOF \leq \delta_m \leq DOF$$

Classic navigation approaches

Overview of core concepts:

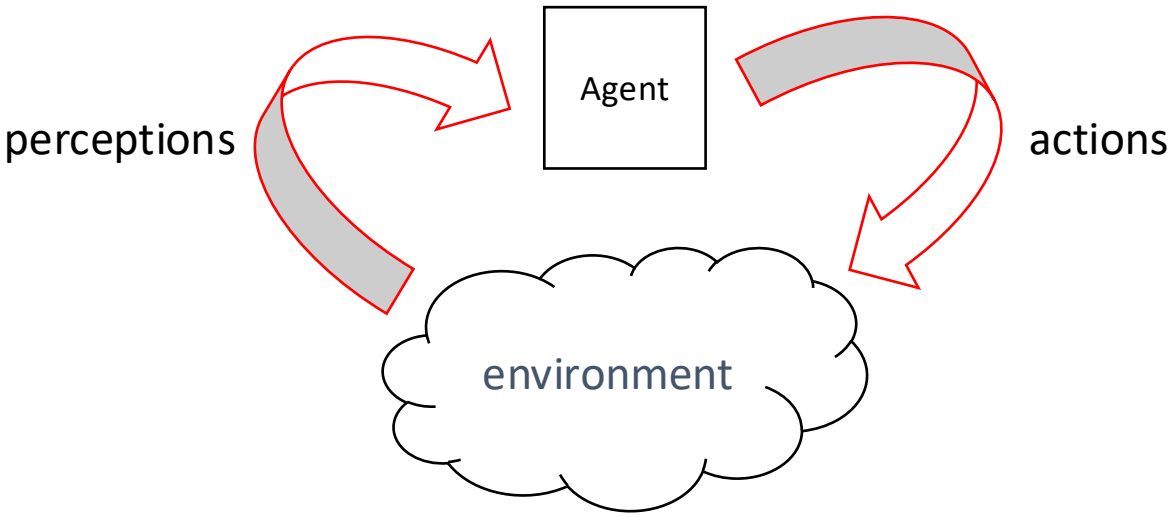
- Robot Motion
- Perception
- Localization and Mapping
- Navigation



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

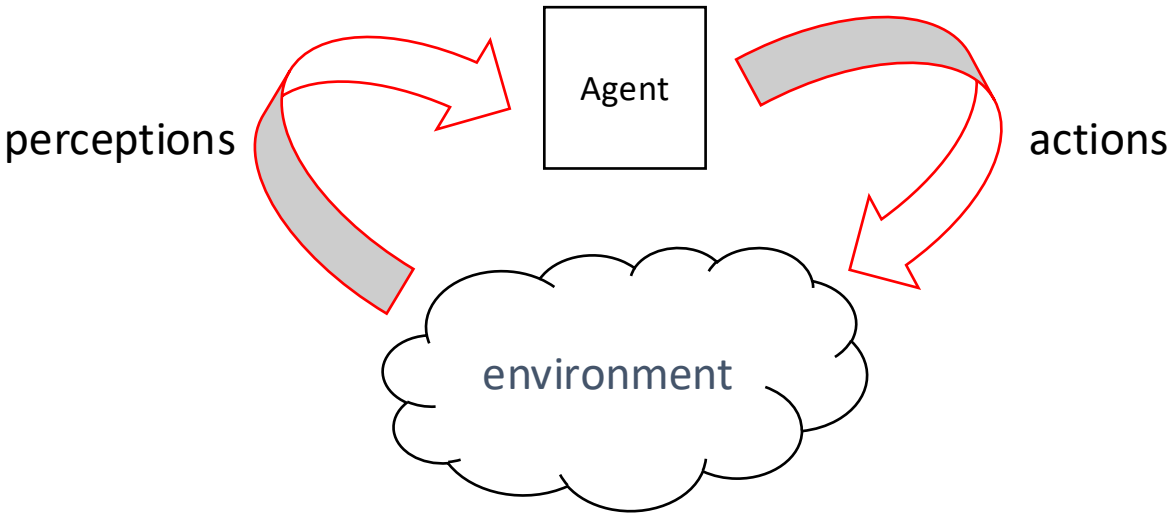
Perception: sensor types

- 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

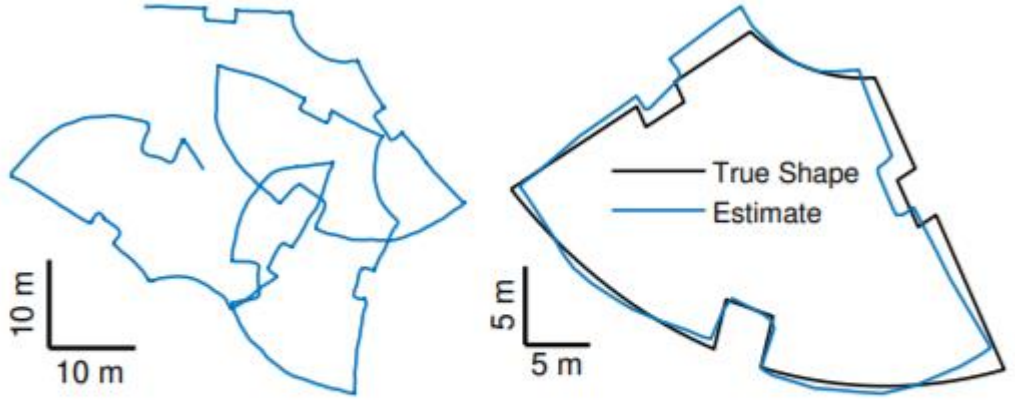
In Situ Sensors performance characterization

- 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 are noisy due to features of the environments or movements of the robot
  (e.g., glass surfaces, mirrors, robot rotation in place, …)
- Robot have different and concurrent sensors – different errors
- Combining different sensor is used to model error/noisy measurements 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.



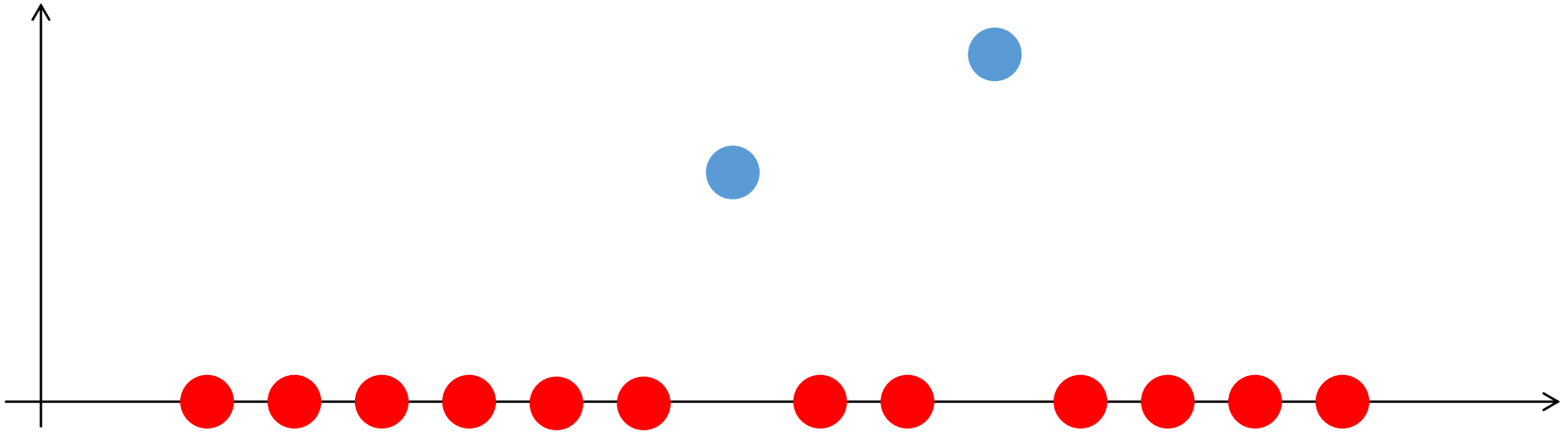
(b) The left panel shows the estimated path of the robot generated from its wheel odometry and the right panel the estimated map and the true shape of the test environment.

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)



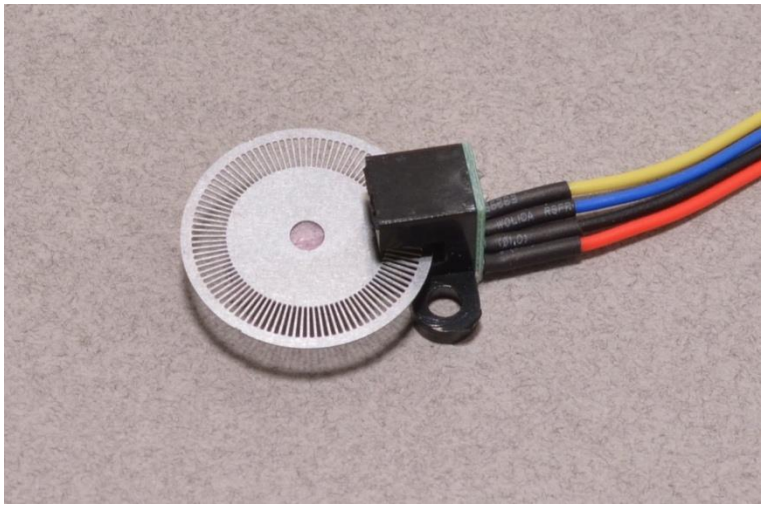
Sistemi Intelligenti Avanzati – 2025/2026
Università degli Studi di Milano

Introduction to Autonomous Mobile Robotics Parte 2

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

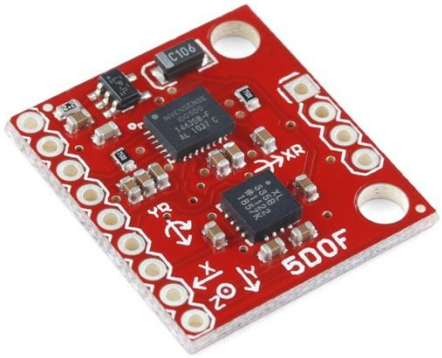
Wheel/Motor sensors

- 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



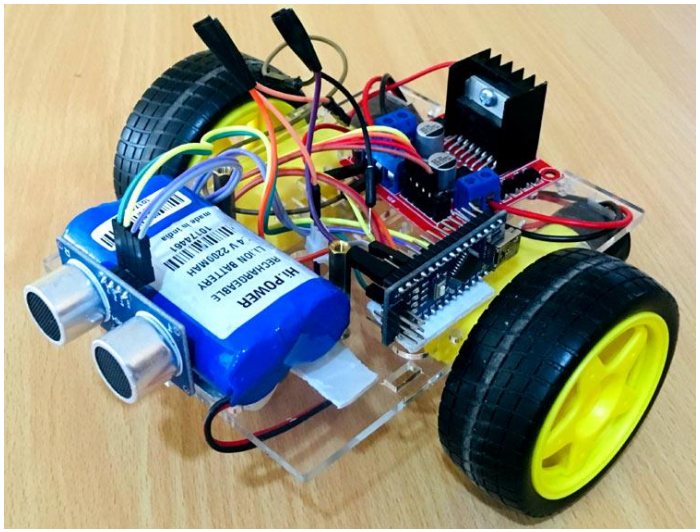
Active ranging

- 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

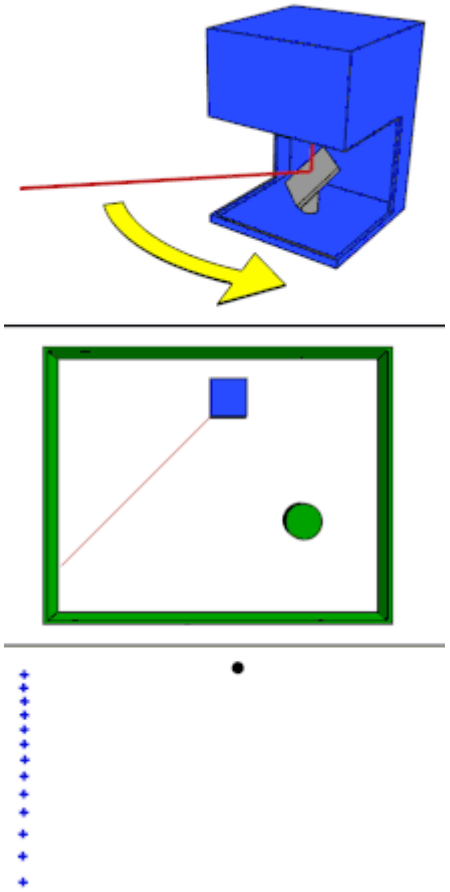
- Cheap
- Not particularly accurate
- Simple and interpretable measurements
- Good for proximity – obstacle avoidance
- Low range



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

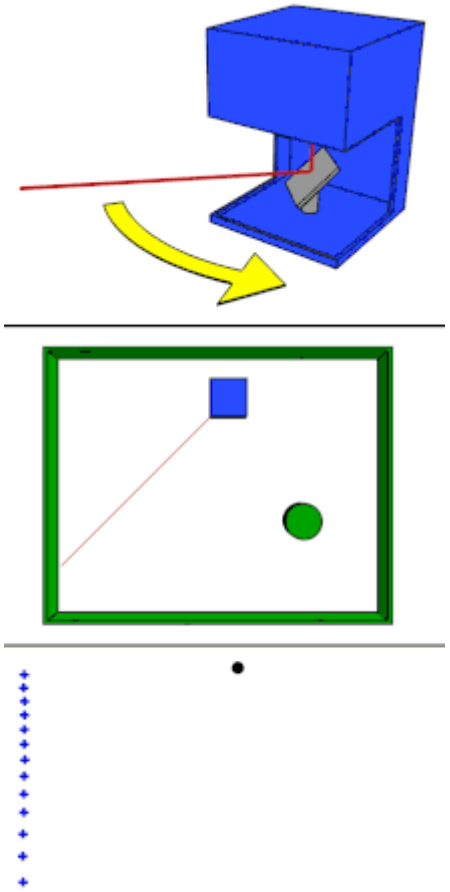


Laser range finders - Lidars

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 entry level lidars, 1000-5000€ for more robust models).

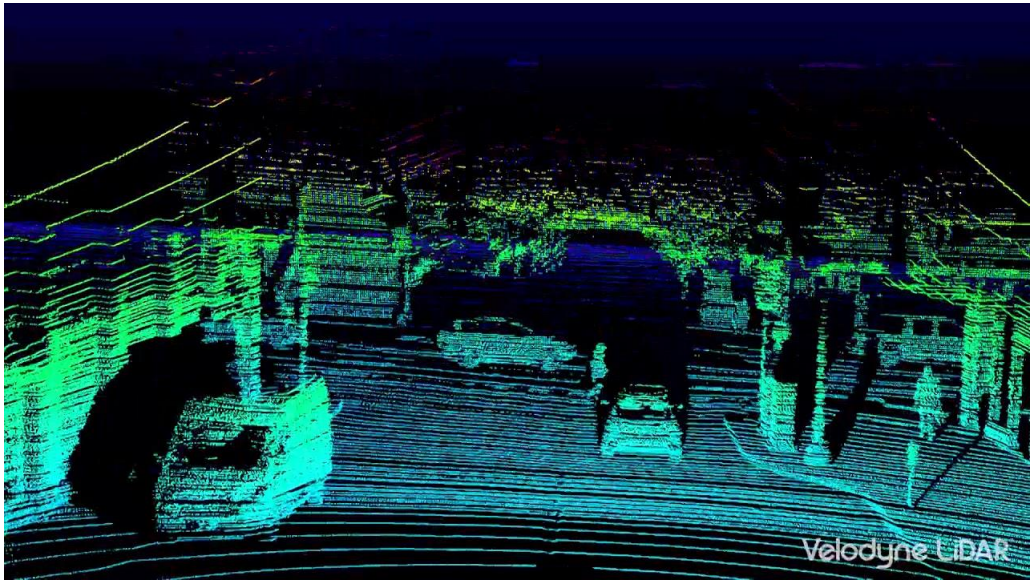
Outdoor lidars have a range from 10 to 30-50m, with a FOV of 180-360°.

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

- Cheap
- Long range
- Up to 360° FOV
- Usually 2D / 2D multi plane
- Simple output
- Subject to reflections
- Measure the spatial surrounding of the robot
- Good to infer spatial occupancy, difficult to infer semantics

Camera

- Cheap
- Close range
- Limited FOV
- 3D
- Complex output
- Subject to distortions, changing light conditions, …
- Measure the appearance of the surrounding of the robot
- 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€)

(b) Asus Xtion Sensor

Other sensors types

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

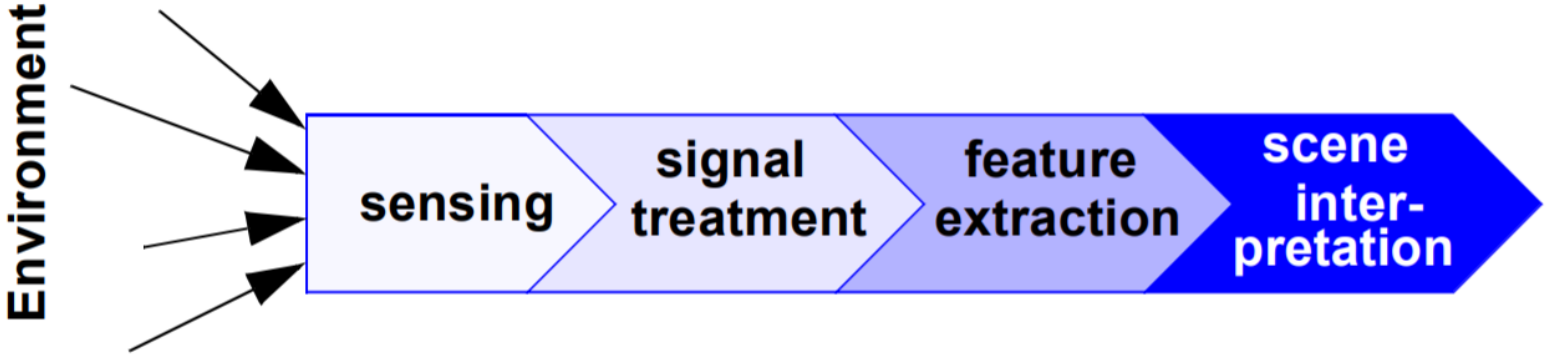
- Bumpers
- Olfactometry
- Chemicals
- Temperature sensors
- NFC readers
- RFID readers
- Radio – or other communication mechanisms…
- …

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 errors and uncertainty 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.

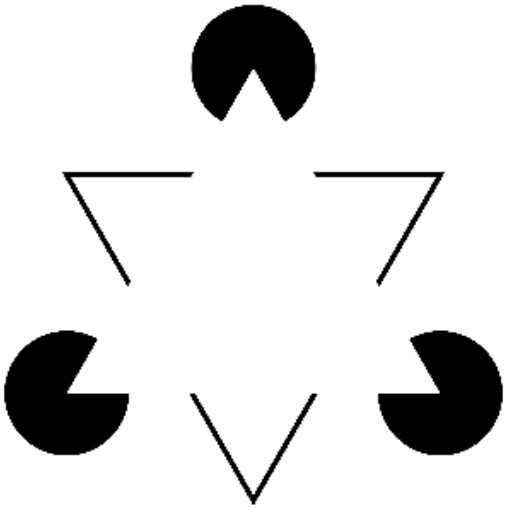
Features extraction and computation: two issues

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., use full image or only features (edges, lines, …)
- How much of that info can be processed in real time?
  E.g. ResNet18 processes images of size 224x224



Feature extraction for 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.

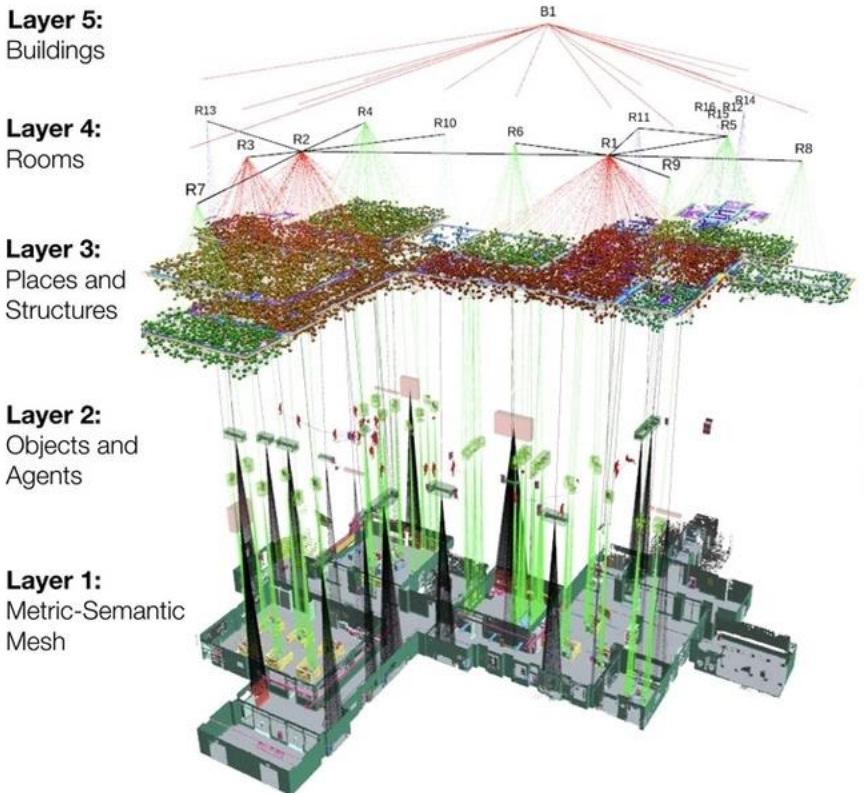
More properties of features

- 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) semantic perception.

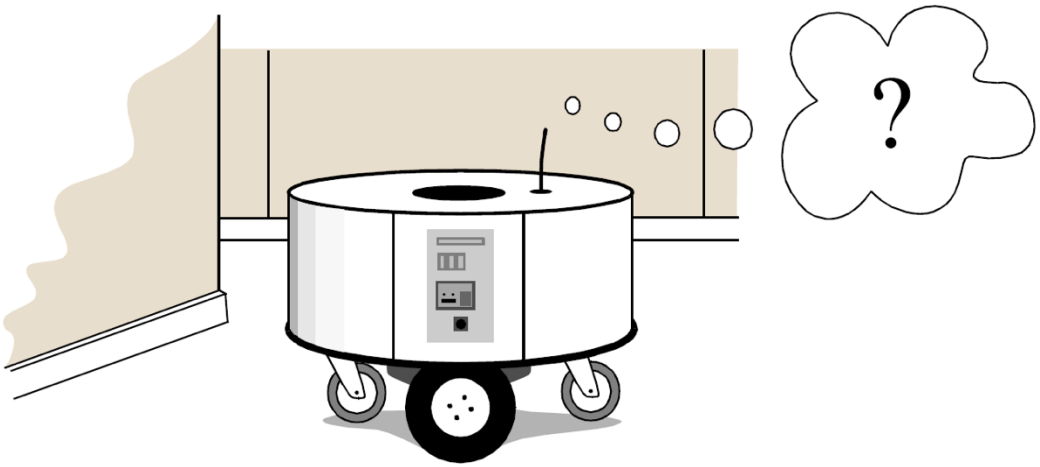
- Place recognition:
  Identification of rooms/locations (corridor, kitchen, office, …)
- Object detection
- Semantic segmentation:
- Loop closure:
  Identify that the robot has already observed the same location (perhaps from a different point of view).



Classic navigation approaches

Overview of core concepts involving robot mobility:

- Robot Motion
- Perception
- Localization and Mapping
- 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, Robot Mapping and Exploration

Other problems: effector noise

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

Example - wheeled robots effector noise:

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

(Current) Solution: Probabilistic Robotics

"A robot that carries a notion of its own uncertainty and that acts accordingly is superior to one that does not"

This is done by having a *belief* about the status of the world and of the robot.

This requires to have *models* and conditional probabilities distribution about what is relevant to the robot.

The belief is updated iteratively (with time) following two steps:

1. **Prediction**: given my current belief and the models, predict the next status (belief)
2. **Update**: given new observation, compute the error between what I was expecting and what has really happening. Use this to update the belief

Localization

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

It needs:

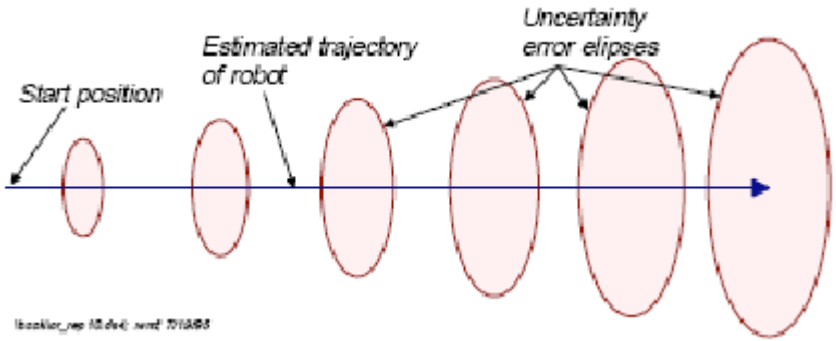
- A model of the robot
- A map of the environment,
- Perception (sensor readings)

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.

Odometry

Odometry is the use of data from motion sensors to estimate change in position over time, to estimate the robot position relative to a starting location.



It could be used open-loop to estimate the robot position, but needs integration.

Rotational movements are fare more difficult to measure than translational ones.

Odometry

Odometry is usually computed from proprioceptive sensors (as wheel encoders)

Other methods reconstruct odometry from exteroceptive sensors:

- Lidar-based odometry
- Visual Odometry

In addition, other sensors/input can be used as IMUs or GPS (multimodal).

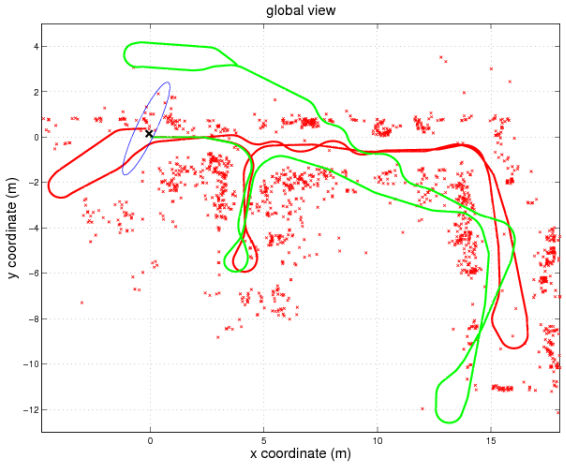
Examples: quadcopters usually have VIO (Visual-Intertial-Odometry)

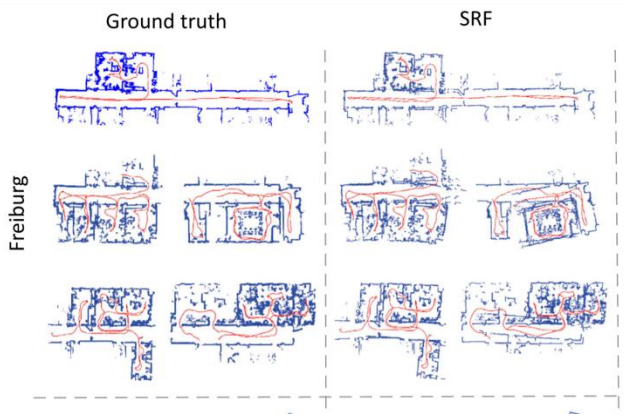
Odometry

Odometry is usually computed from proprioceptive sensors (as wheel encoders)

Other methods reconstruct odometry from exteroceptive sensors:

- Lidar-based odometry
- Visual Odometry





From https://doi.org/10.1109/TRO.2018.2861911

Localization vs Dead-reckoning

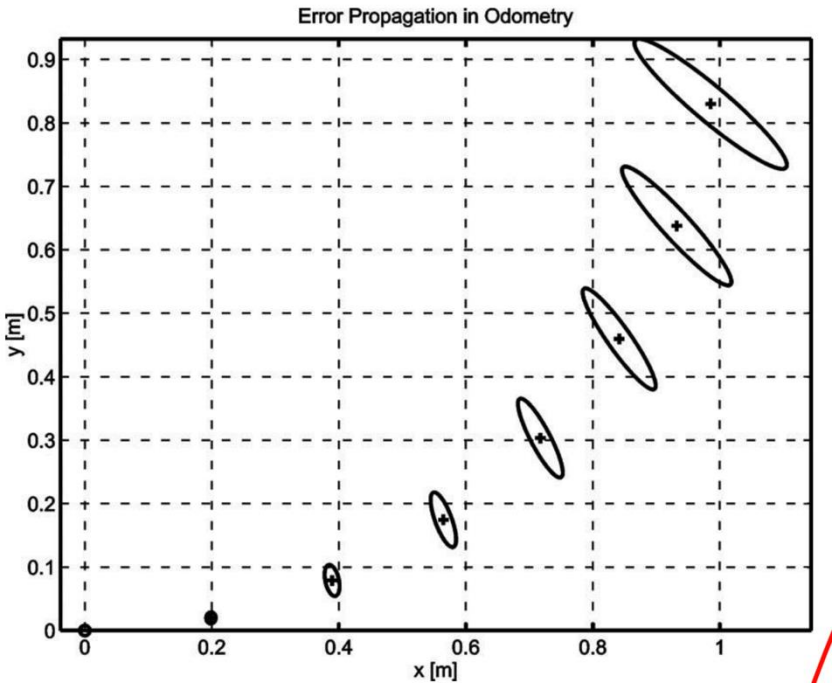
Odometry:

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.

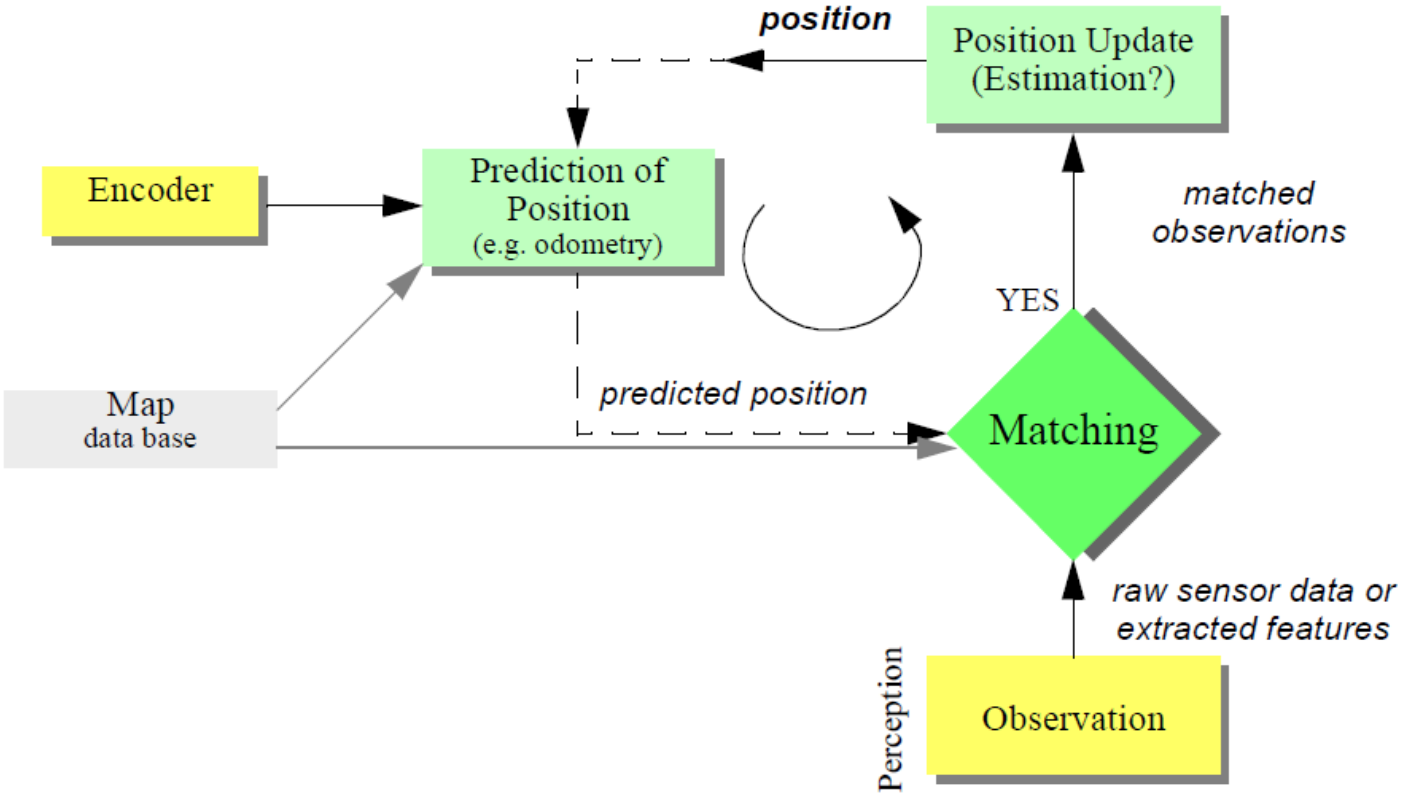
Localization:

Usually closed loop integration of the robot position



Localization

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



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

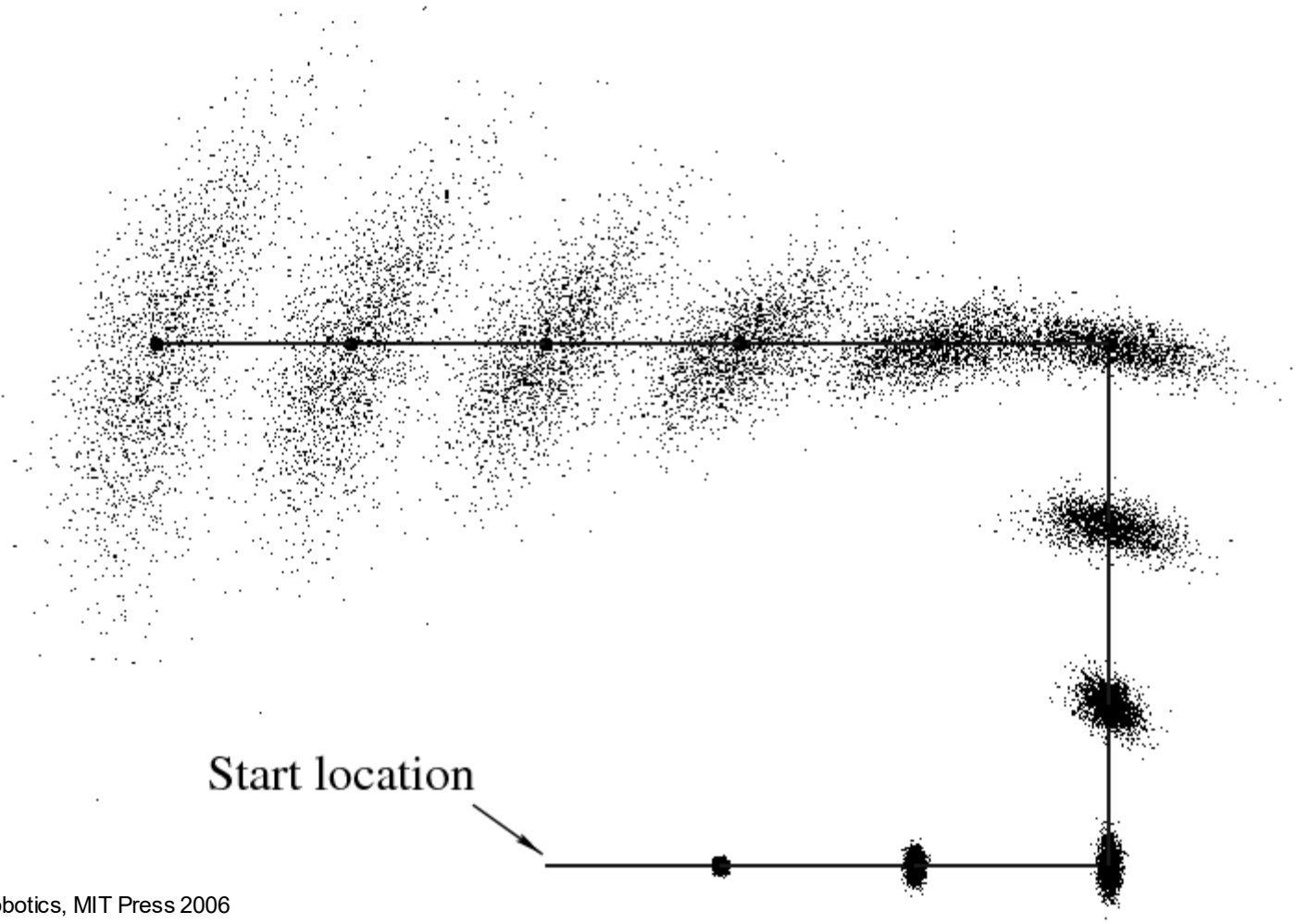
The robot motion model is the probability distribution of the robot pose at time t+1 given the robot pose and the expected robot movement, measured using motion or proprioceptive sensors:

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

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.

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.

When the map is available:

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

describes the probability of a robot measurement z_t given a robot pose $x_{r,t}$ given a map M

Example of Markov Localization
(here $x_r = x$)

$$\overline{bel}_0(x) = \frac{1}{L}$$

z_0: door detected

$$p(z_0|x)$$

$$bel_0(x) = \eta\, p(z_0|x_0)\overline{bel}_0(x) = \eta' p(z_0|x_0)$$

u_0: the robot moves

$$\overline{bel}_1(x) = \int_0^L p(x|x_0 = y, u_0) bel_0(y) dy$$

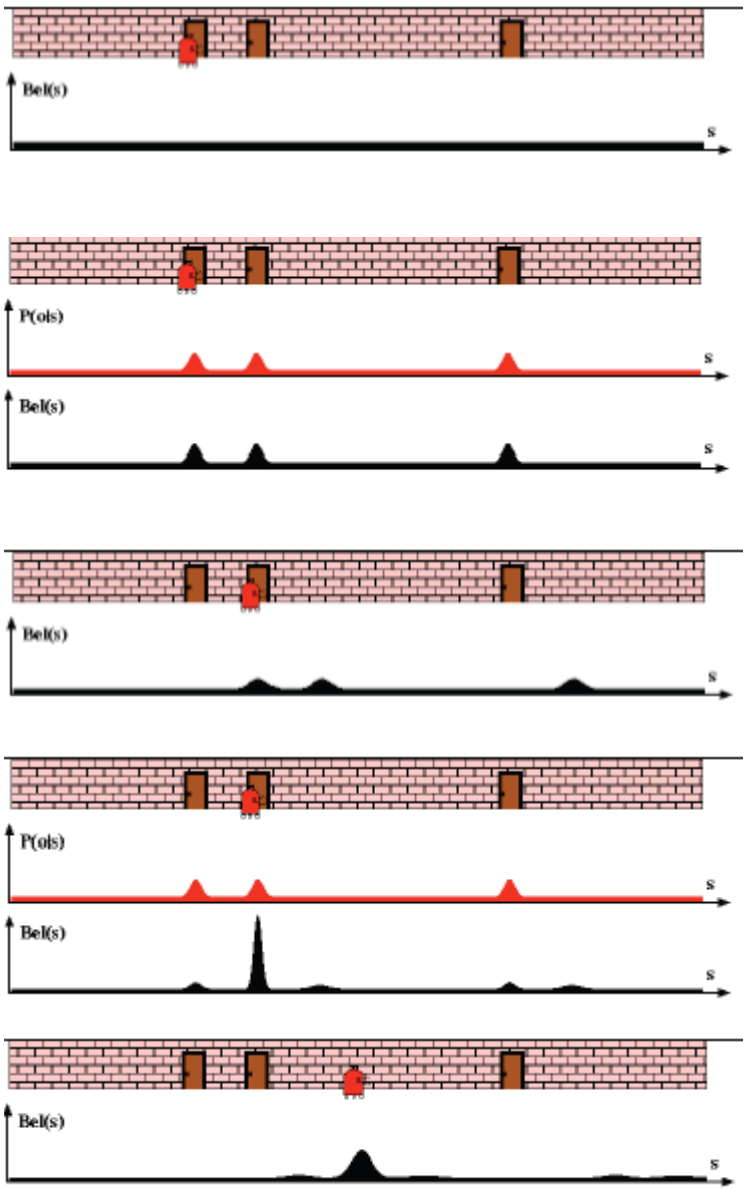
z_1: door detected again

$$p(z_1|x)$$

$$bel_1(x) = \eta\, p(z_1|x_1)\overline{bel}_1(x)$$

u_1: the robot moves again

$$\overline{bel}_2(x) = \int_0^L p(x|x_1 = y, u_1) bel_1(y) dy$$



From Thrun Burgard Fox, Probabilistic Robotics, MIT Press 2006

Robot localization algorithms

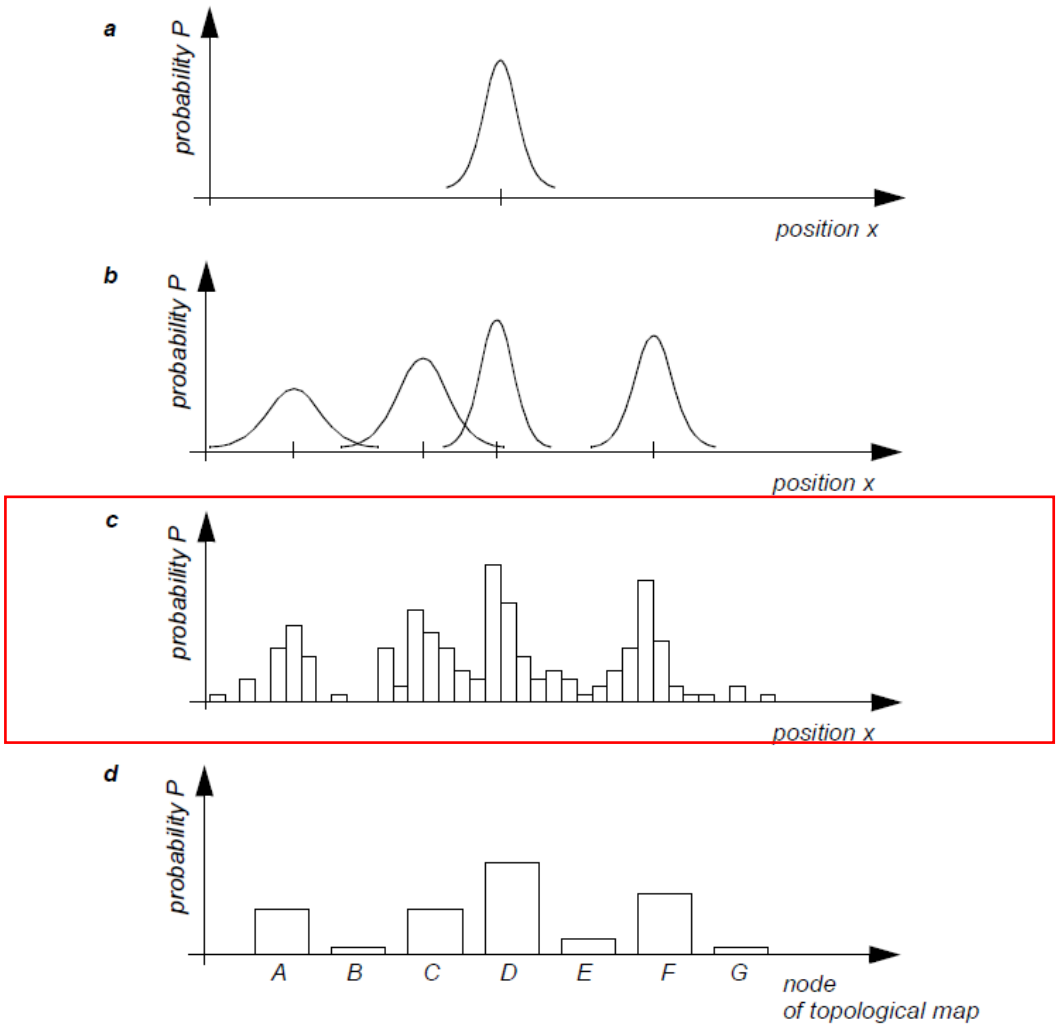
The localization model assumes so to predict the current position from movement, to observe if the measurements are coherent with the estimated position, and to close the loop by updating the robot belief.

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

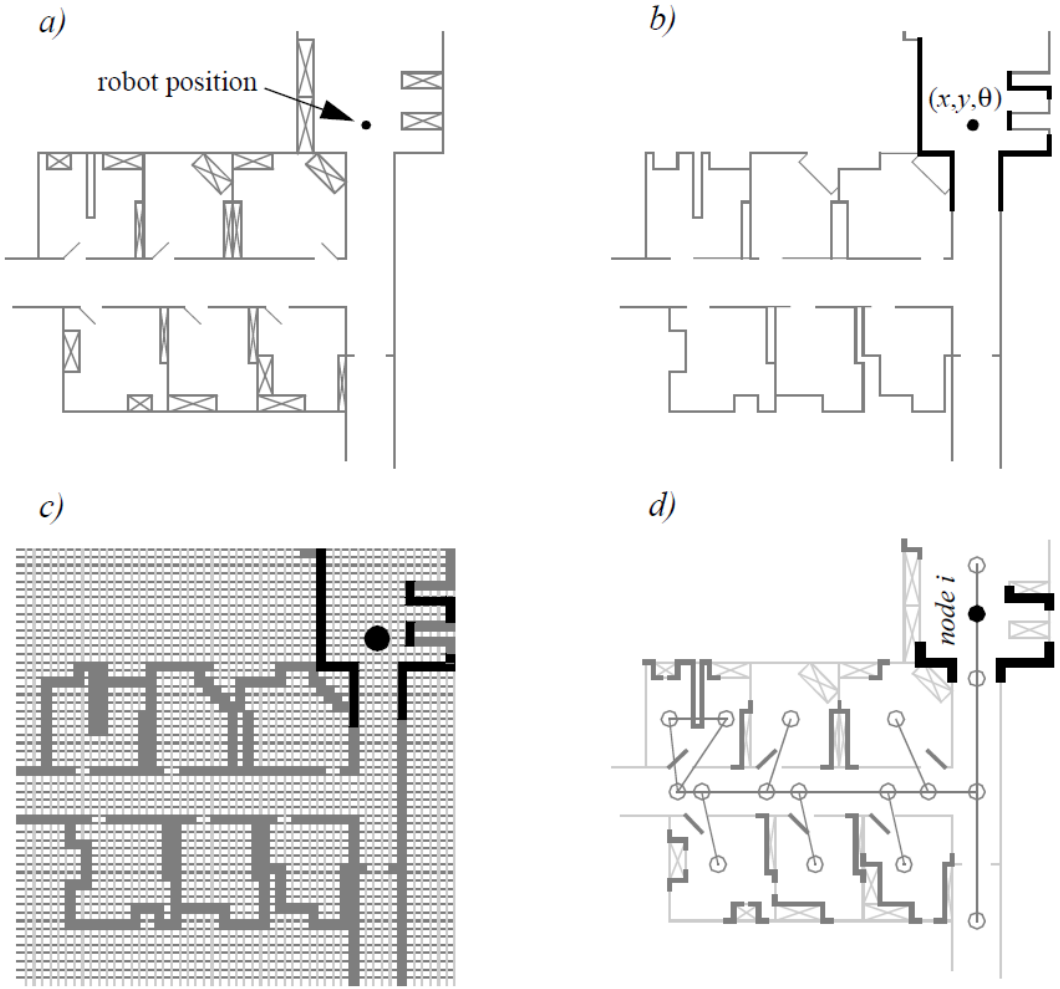
We can represent the belief as a single hypothesis of by have a distribution probability
over <u>multiple hypotesis</u>



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.

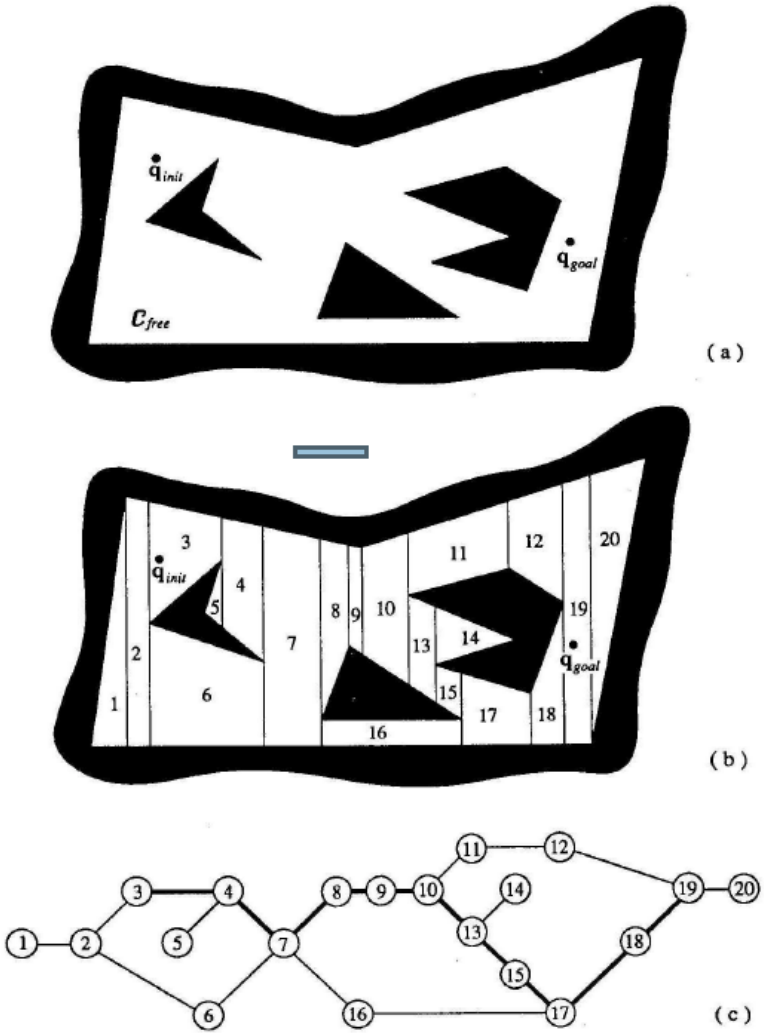
- Continuous vs discrete representation
- Occupancy vs topological maps
- Closed world assumption: only what there is in the map exists
- Static vs dynamic
- 2D or 3D

A map, overall, is just 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.



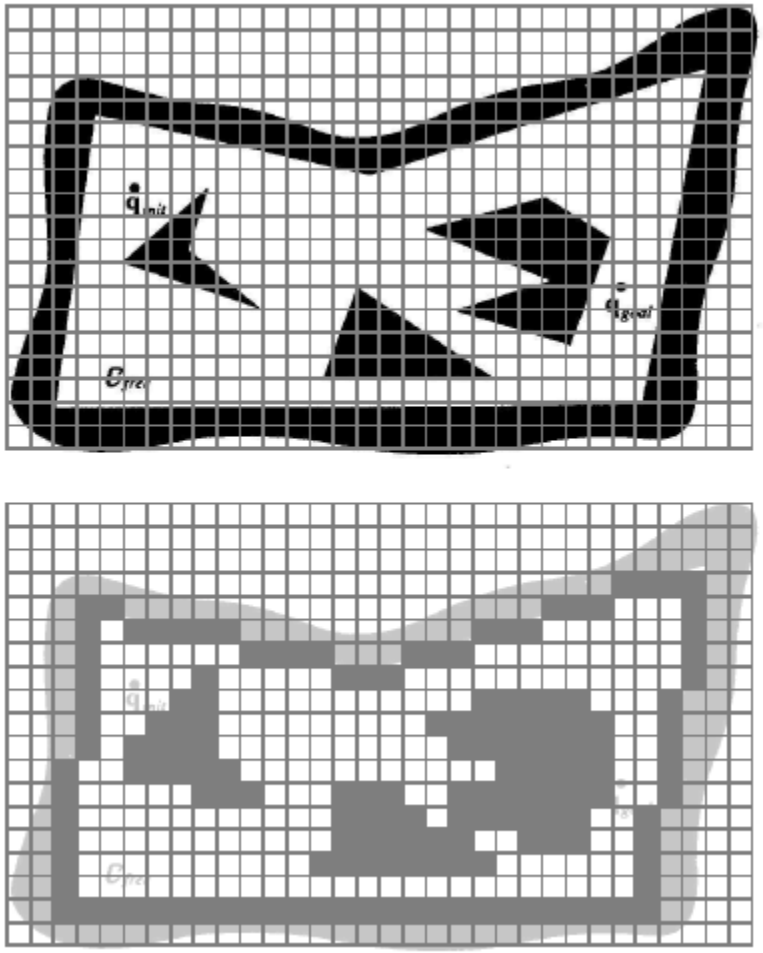
From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

Fixed decomposition maps

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

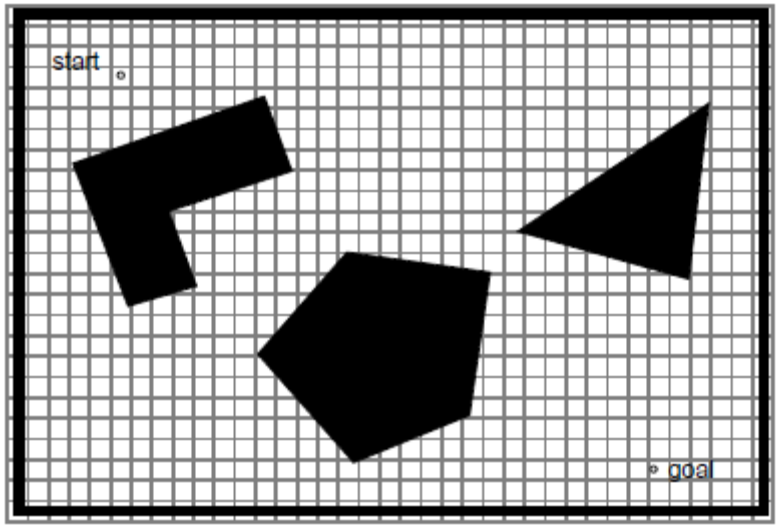
Narrow passages disappear, but each cell has the same representation.

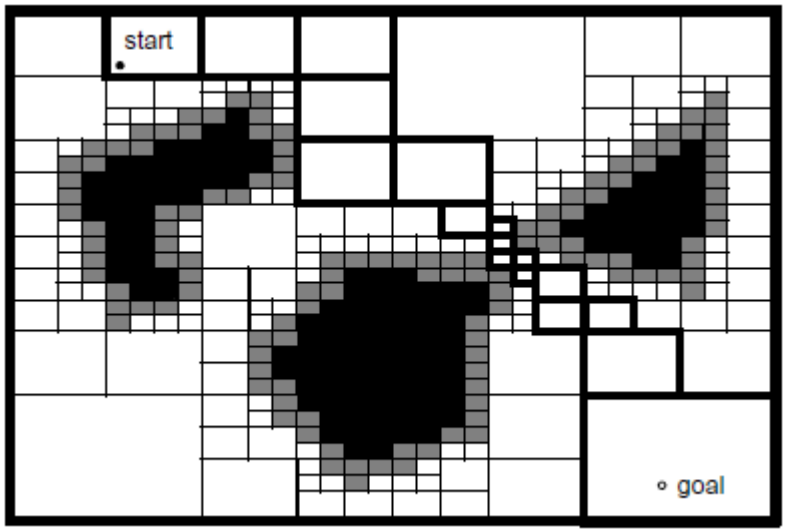
We obtain a grid map.
We can also assign different type of values to each cell (instead of 1-0, e.g. occupancy probability)

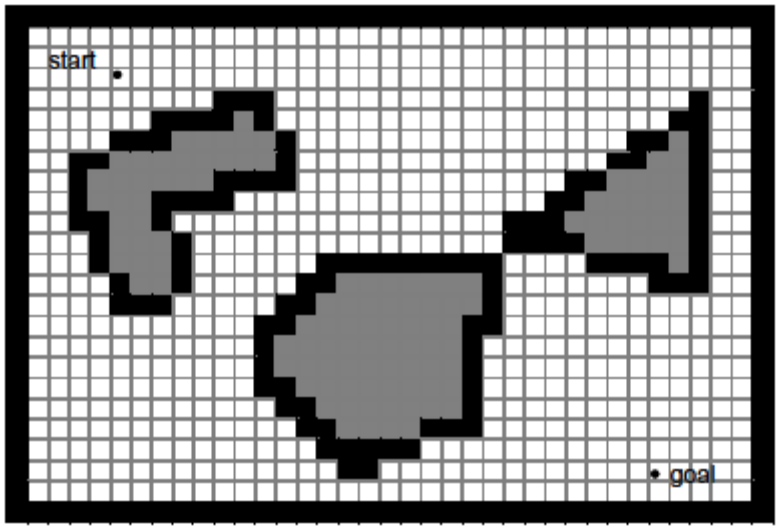


From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

Grid maps





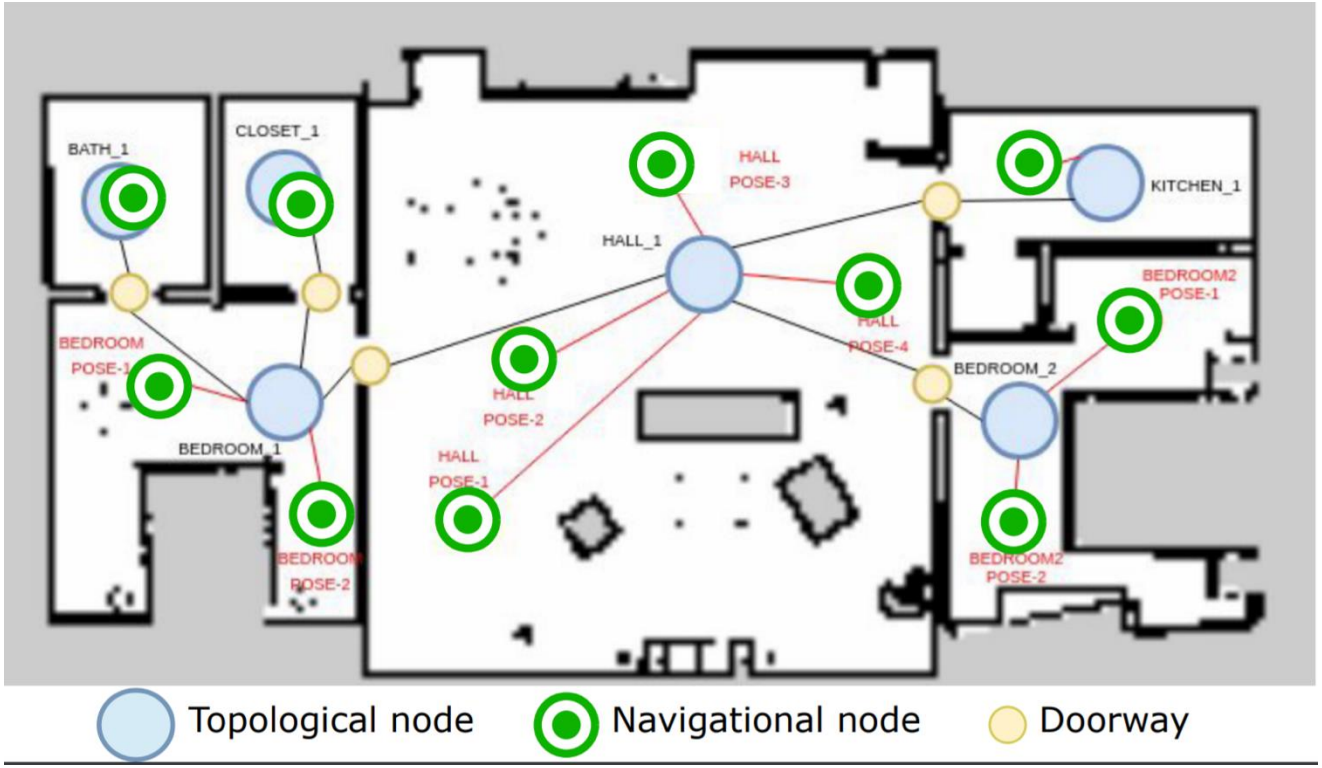


Grid map are a popular approach widely adopted.
As free space is not interesting, we can use cells of different size to optimize the grid map representation.

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.



Localization Example: AMCL

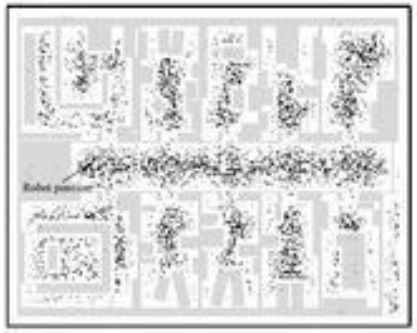


Fig. 2: Global localization: Initialization.

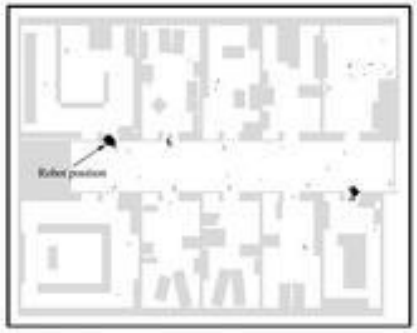


Fig. 3: Ambiguity due to symmetry.

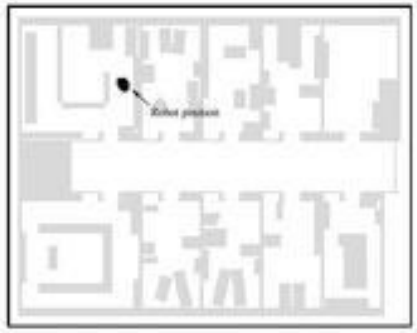
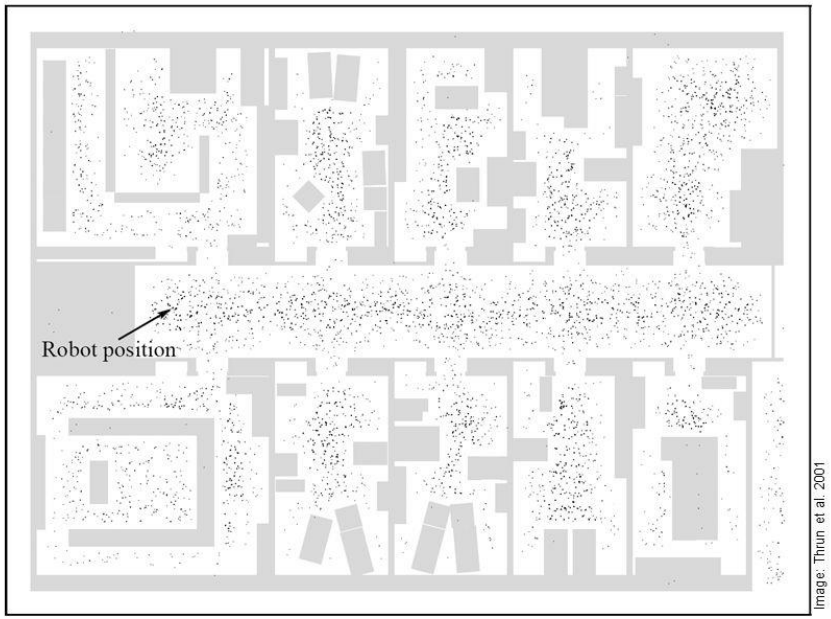


Fig. 4: Successful localization.

Monte Carlo Localization (MCL)

- Particle-filter
- several estimated poses (beliefs) are maintained and updated
- the more the “cloud” of particles is thick, the more localization is accurate



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?

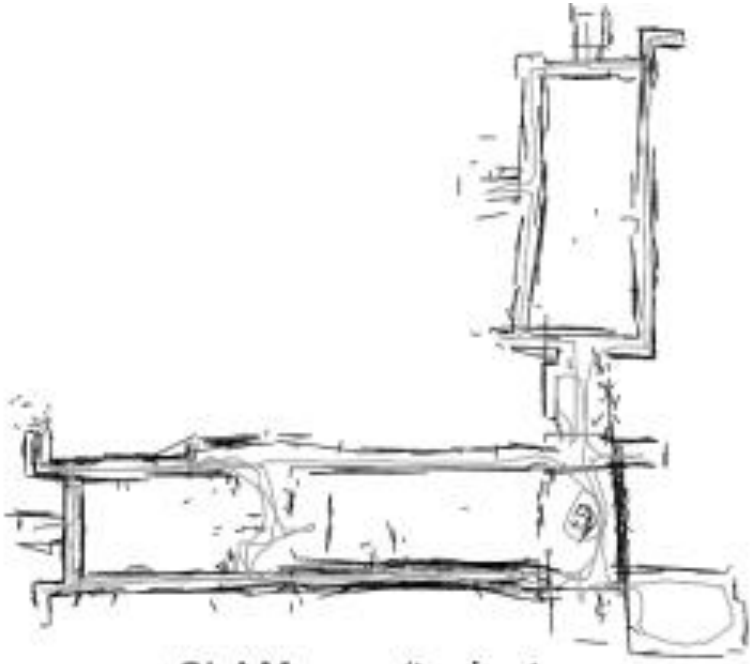
The robot needs at the same time to:

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

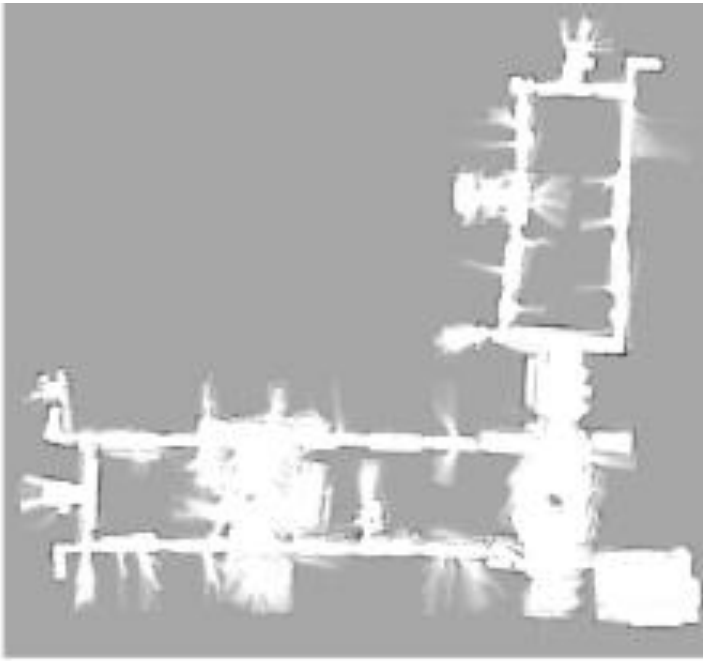
This is called Simultaneous Localization and Mapping (SLAM), a joint estimate of both the environment map and the robot pose.

SLAM

raw



SLAM map/trajectory



occupancy map

How to build a map?

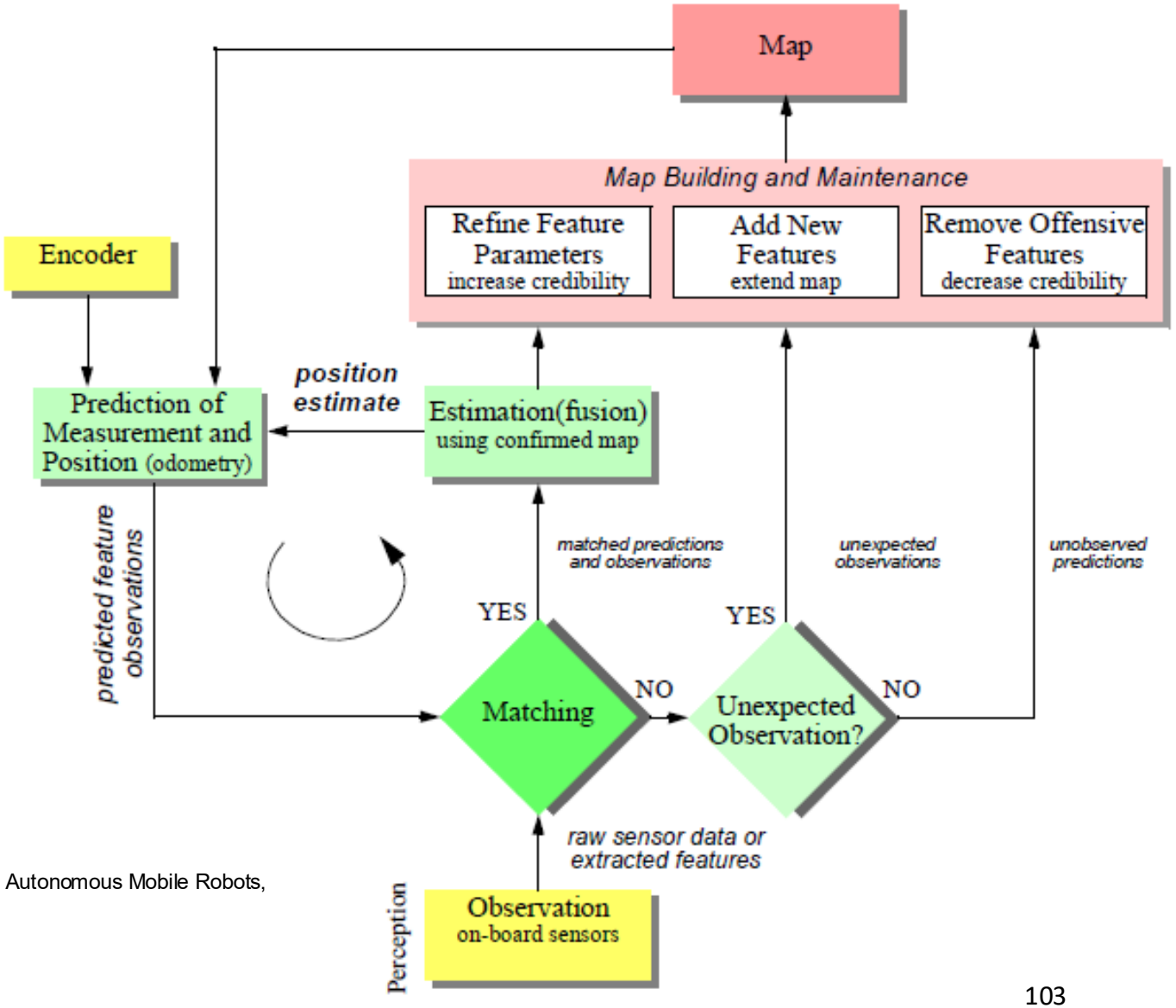
Incrementally, by fusing together sensor readings while correcting sensor error.

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

SLAM

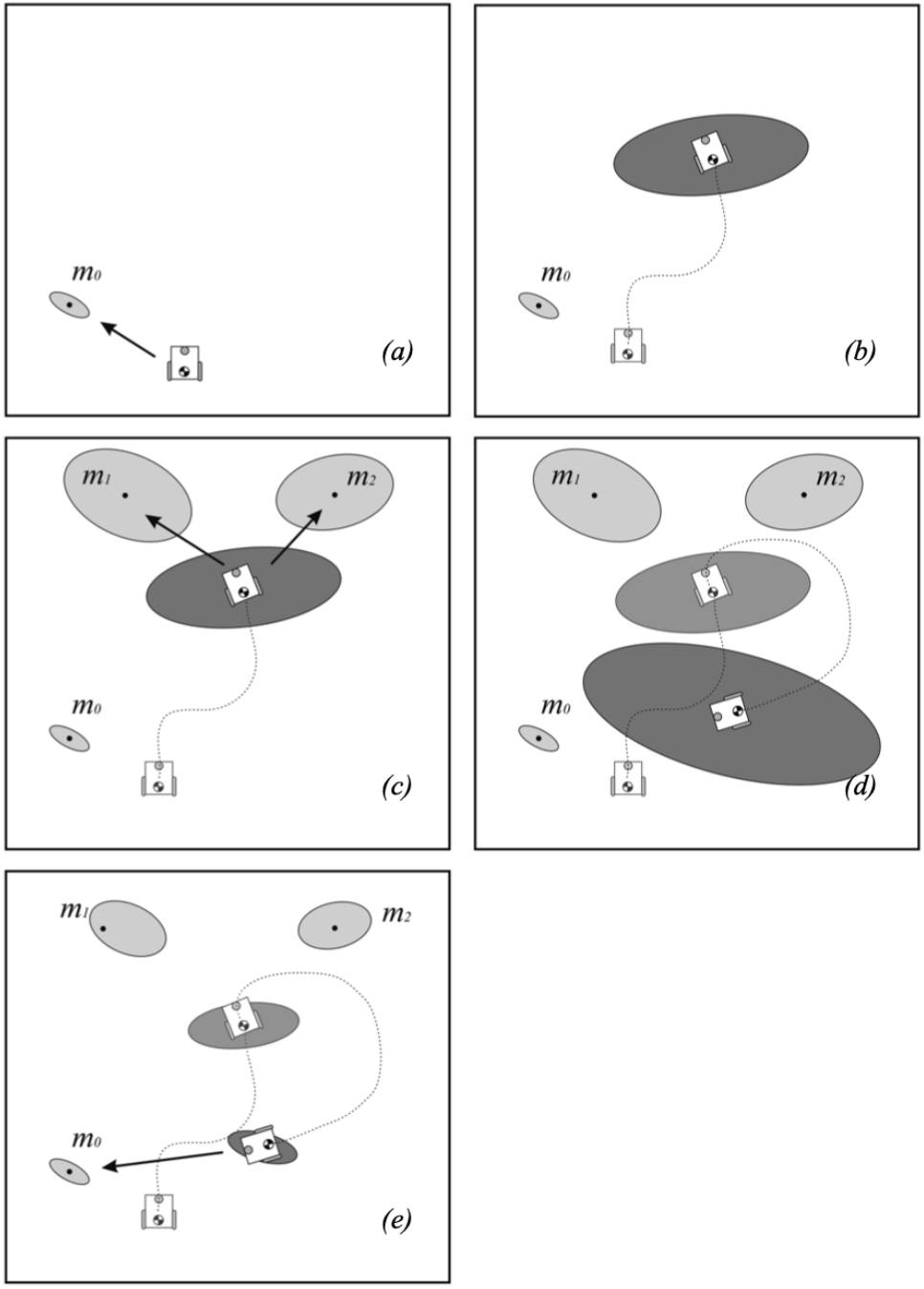
During SLAM the robot integrates sensorial input by correcting odometry and sensing error to provide an estimate of the environment.

At the same time it estimates its pose in it.



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

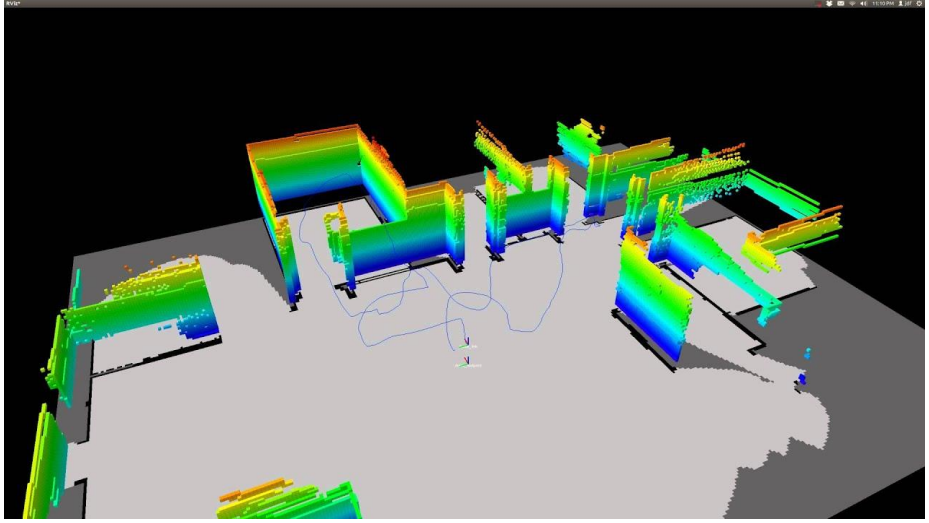
SLAM Example

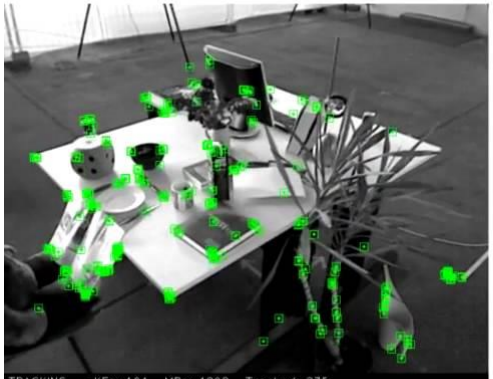


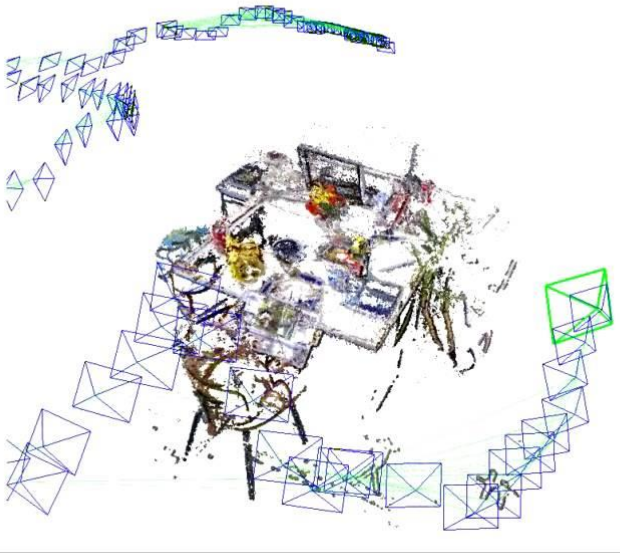
From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2011

Loop closure detection: the robot observes
again the same features,
reducing pose uncertainty

SLAM

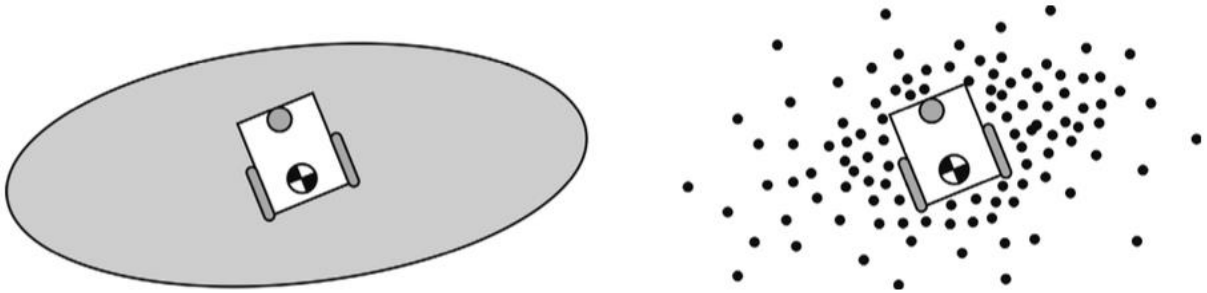






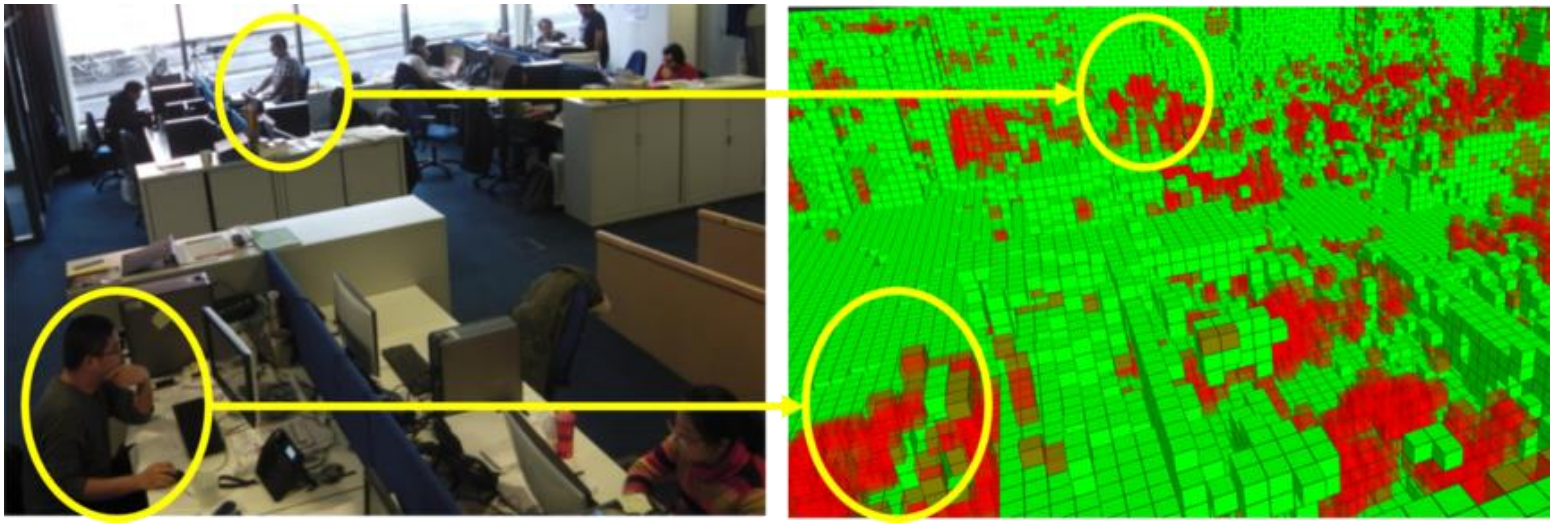
SLAM belief state

- 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) samples.
- 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 discarded 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 represents 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

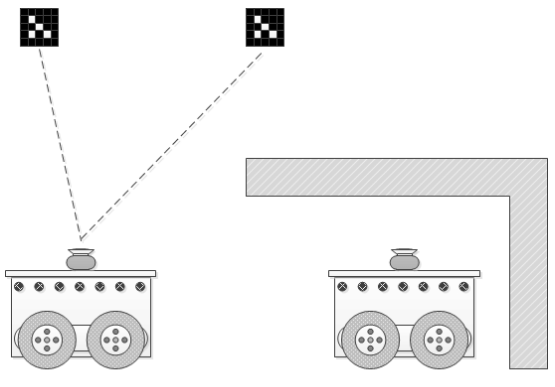
An alternative is to put a set of easily recognizable landmarks in the environment:

- 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

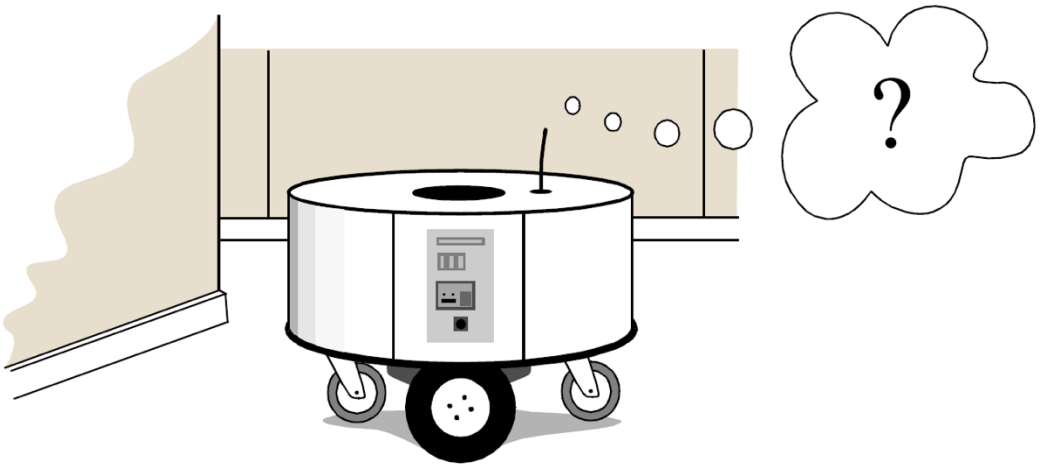
Landmarks: beacons, QR codes or visual markers, usually in placed without obstruction (ceiling)



Classic navigation approaches

Overview of core concepts:

- Robot Motion
- Perception
- Localization and Mapping
- Navigation

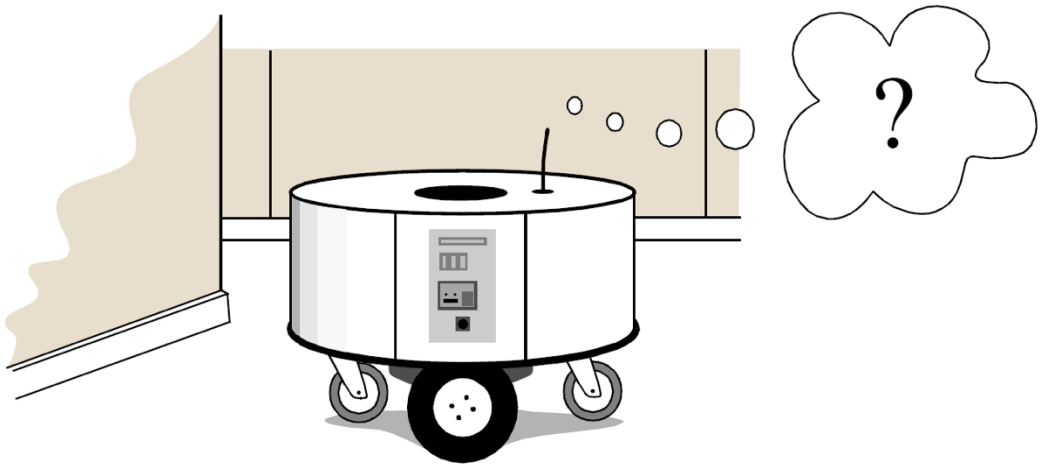


From Siegwart,Introduction to Autonomous Mobile Robots, MIT Press 2004

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.

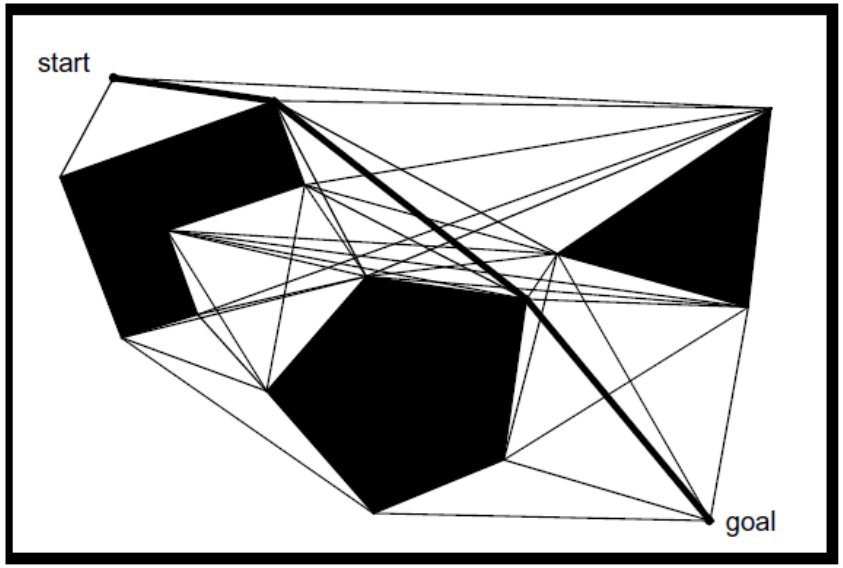
However, as we've seen, we have to provide a proper formulation for this problem:

- 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

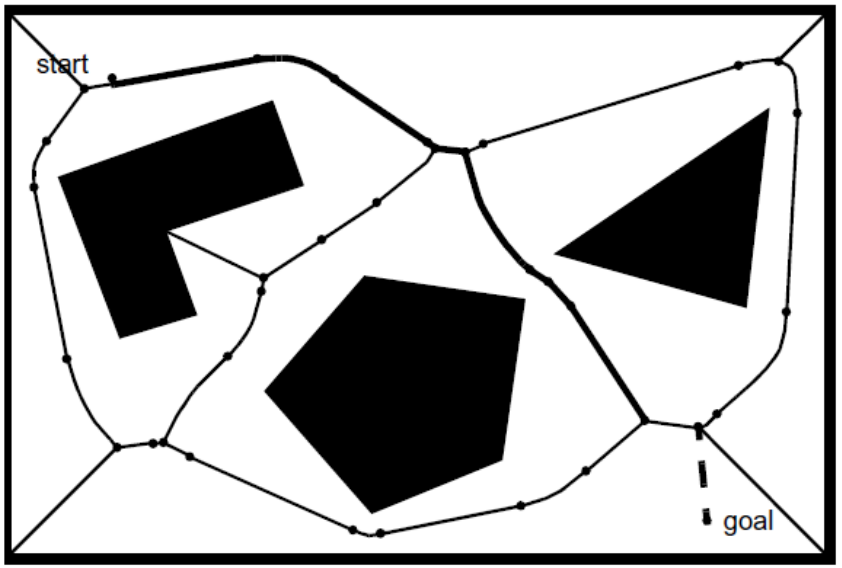
Road map path planning

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

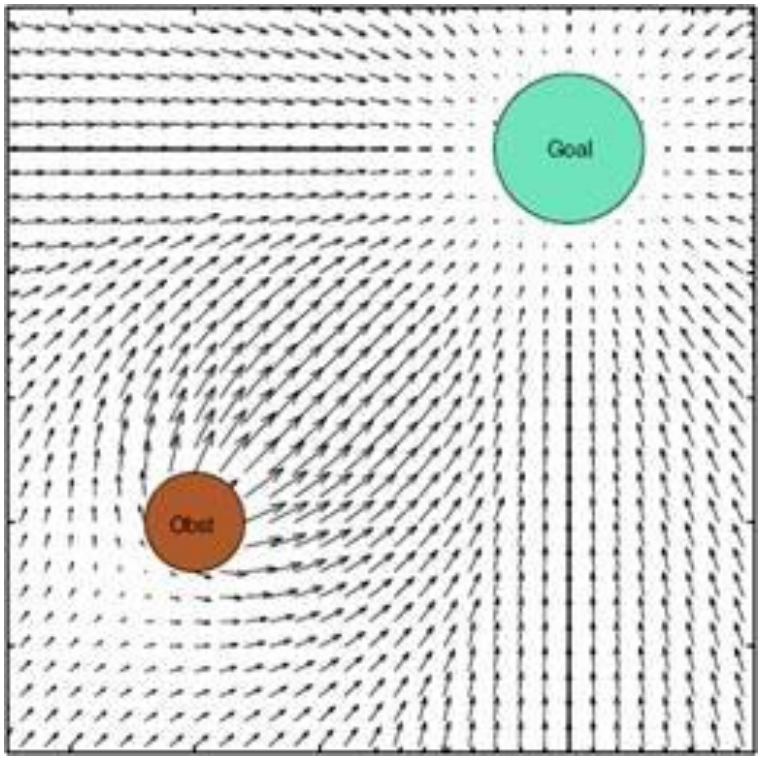


Voronoi Decomposition

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

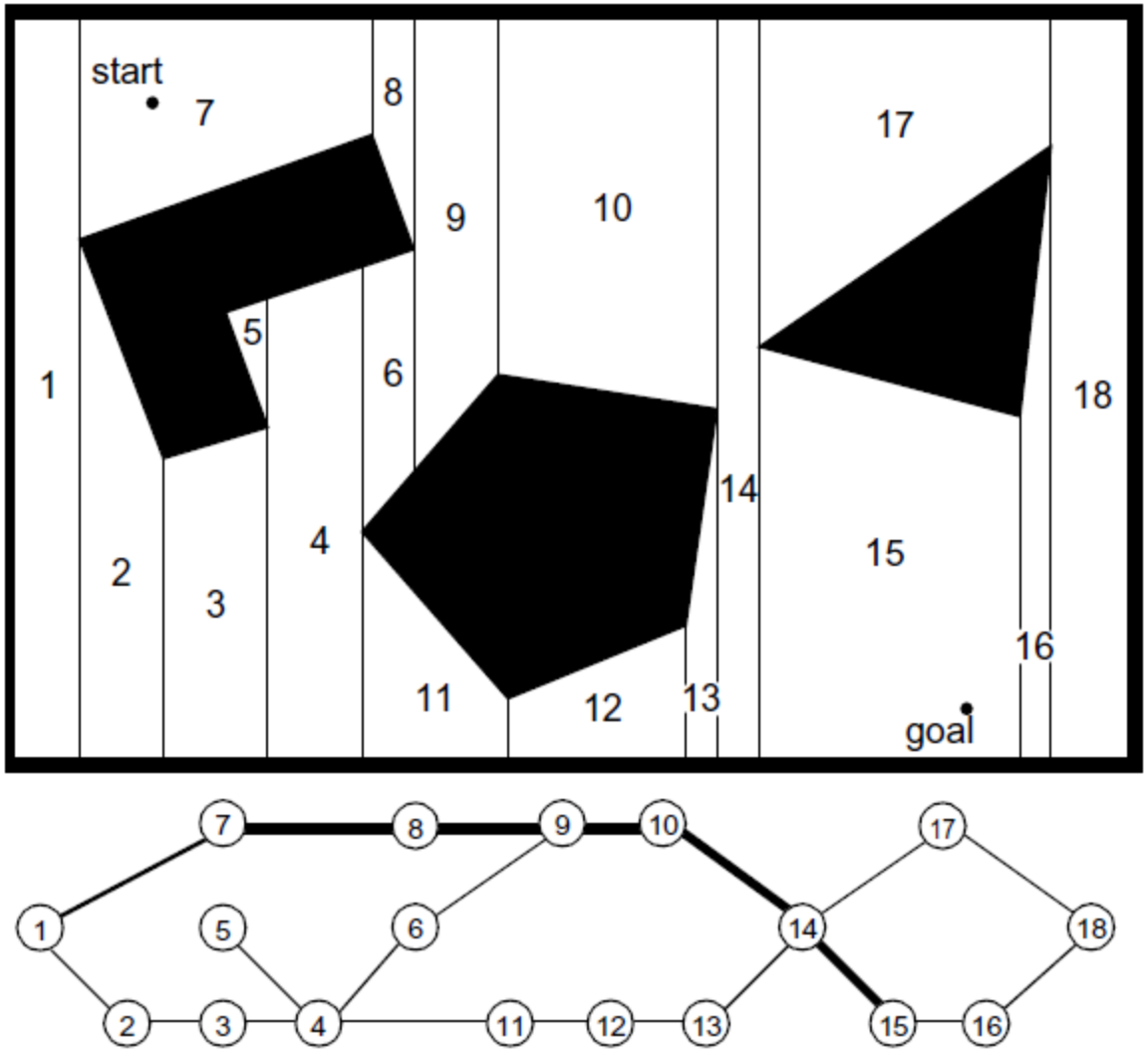
Potential field path planning

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/

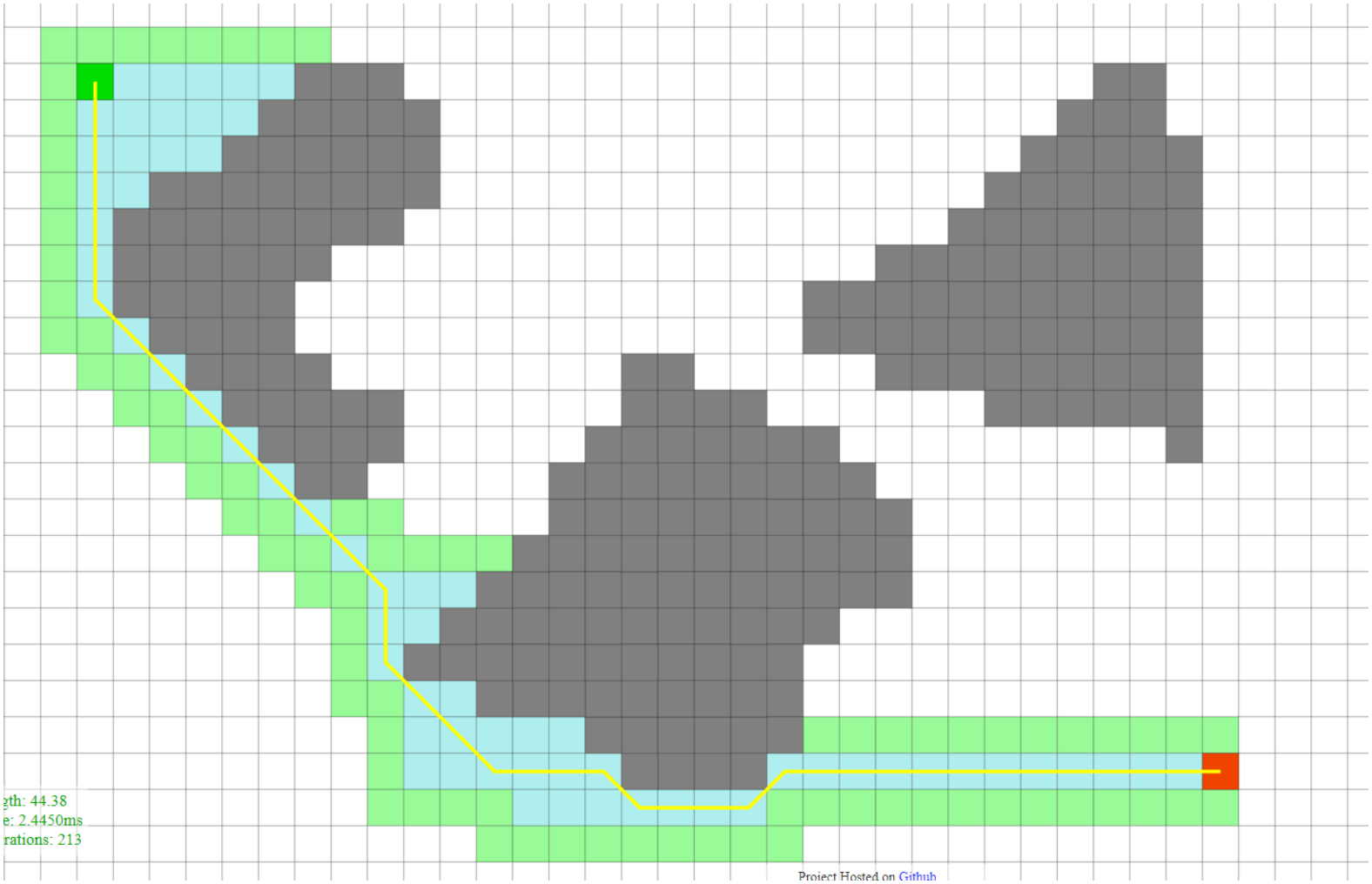
Cell decomposition path planning



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



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

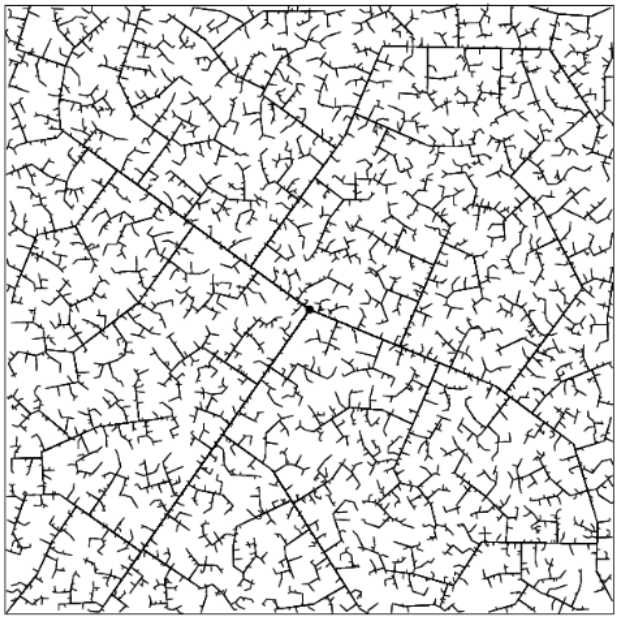
Path Planning Algorithms

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

Examples of algorithms used :

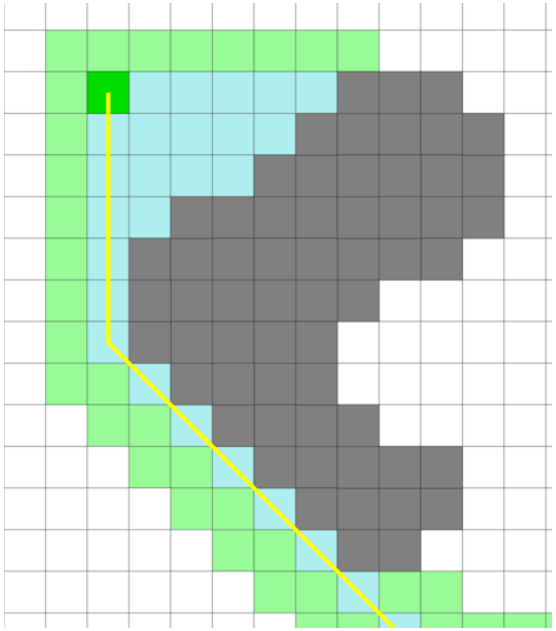
- A*
- RRT (Rapidly Exploring Random Trees)

45 iterations



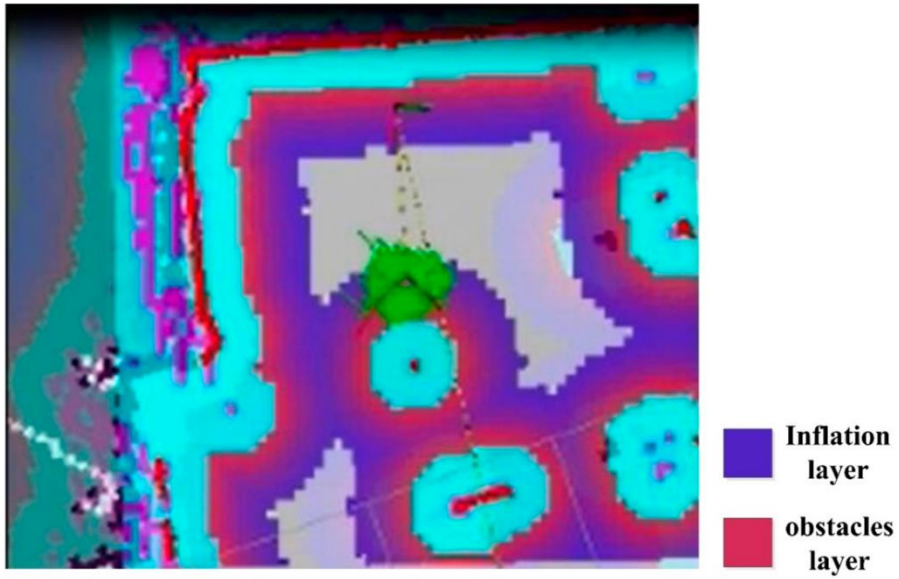
2345 iterations

Obstacle avoidance

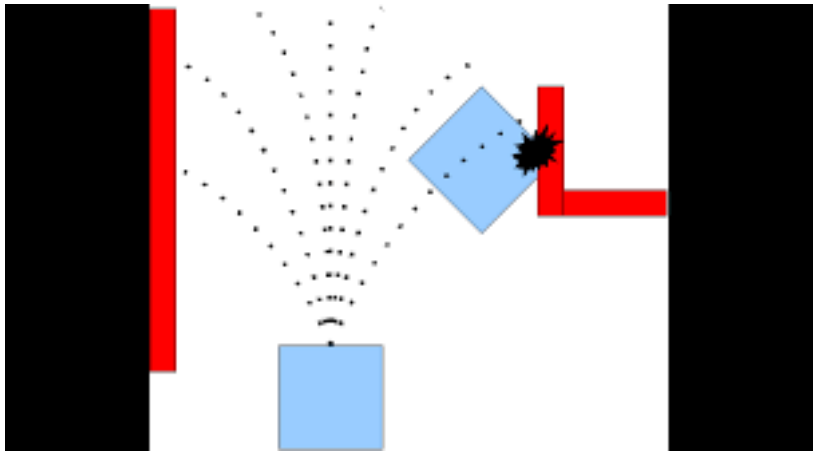


What happens if the robot is bigger than 1 cell (e.g., a 2x2 cell)? Shall we allow trajectories that are that close to the obstacles?

Several techniques are used for obstacle avoidance.
Ex: inflating either the obstacle (considering the robot as a point) or the robot (the robot acts as it is bigger than its actual size)



Obstacle avoidance

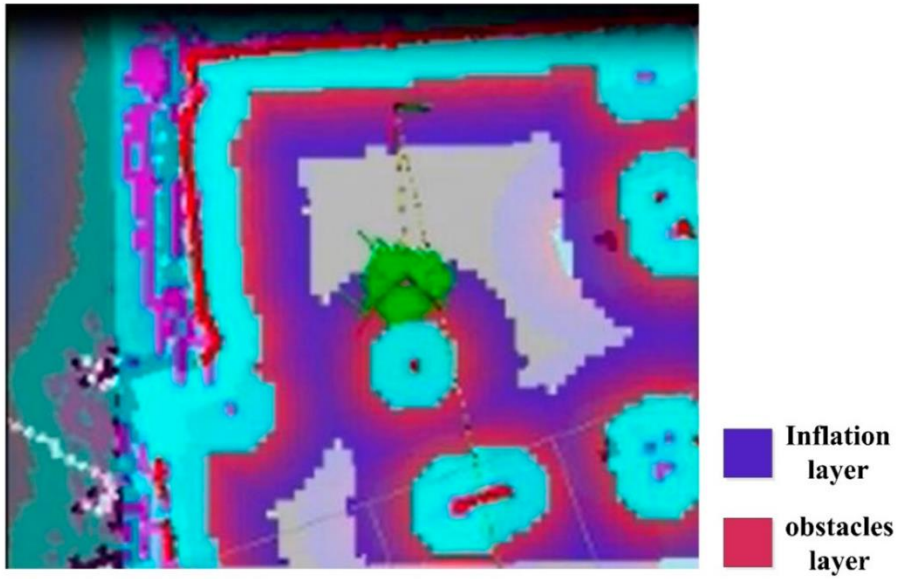


Two-layer architecture

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

Local path planning uses:

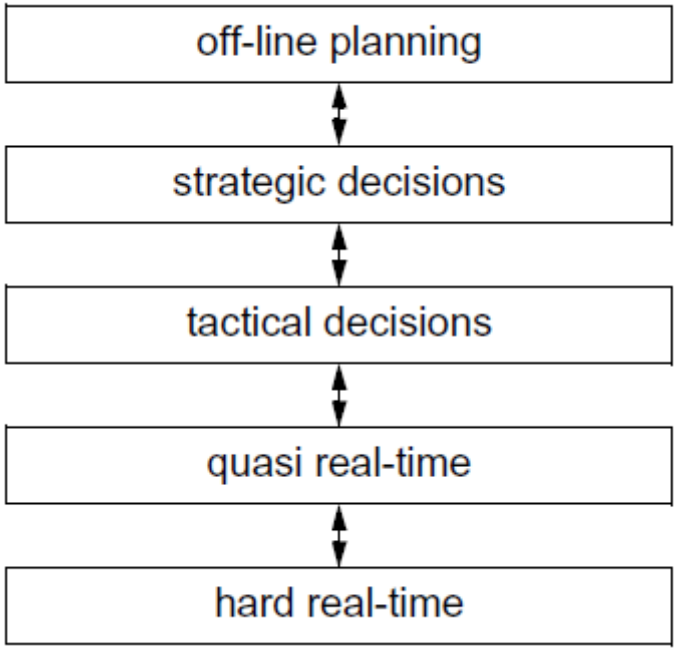
- Potentials fields
- Dynamic windows approaches
- Time-Elastic Bends (TEB) planners

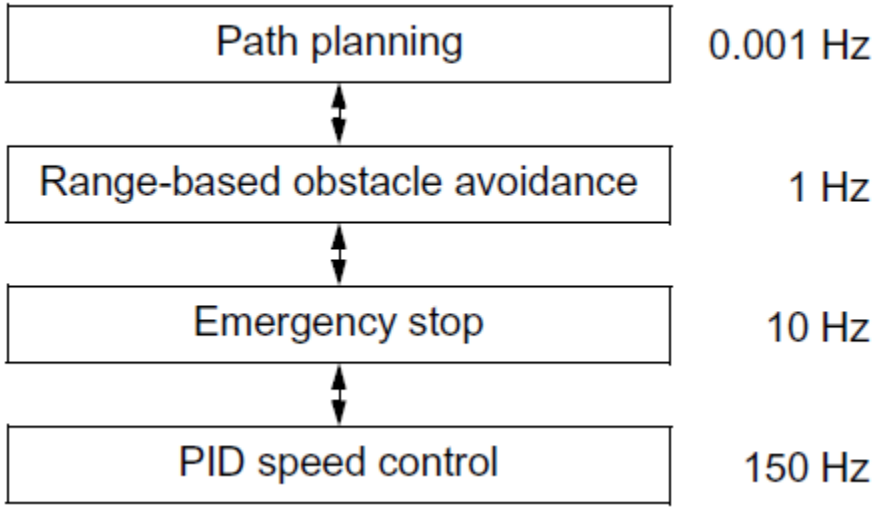


Navigation Architecture

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

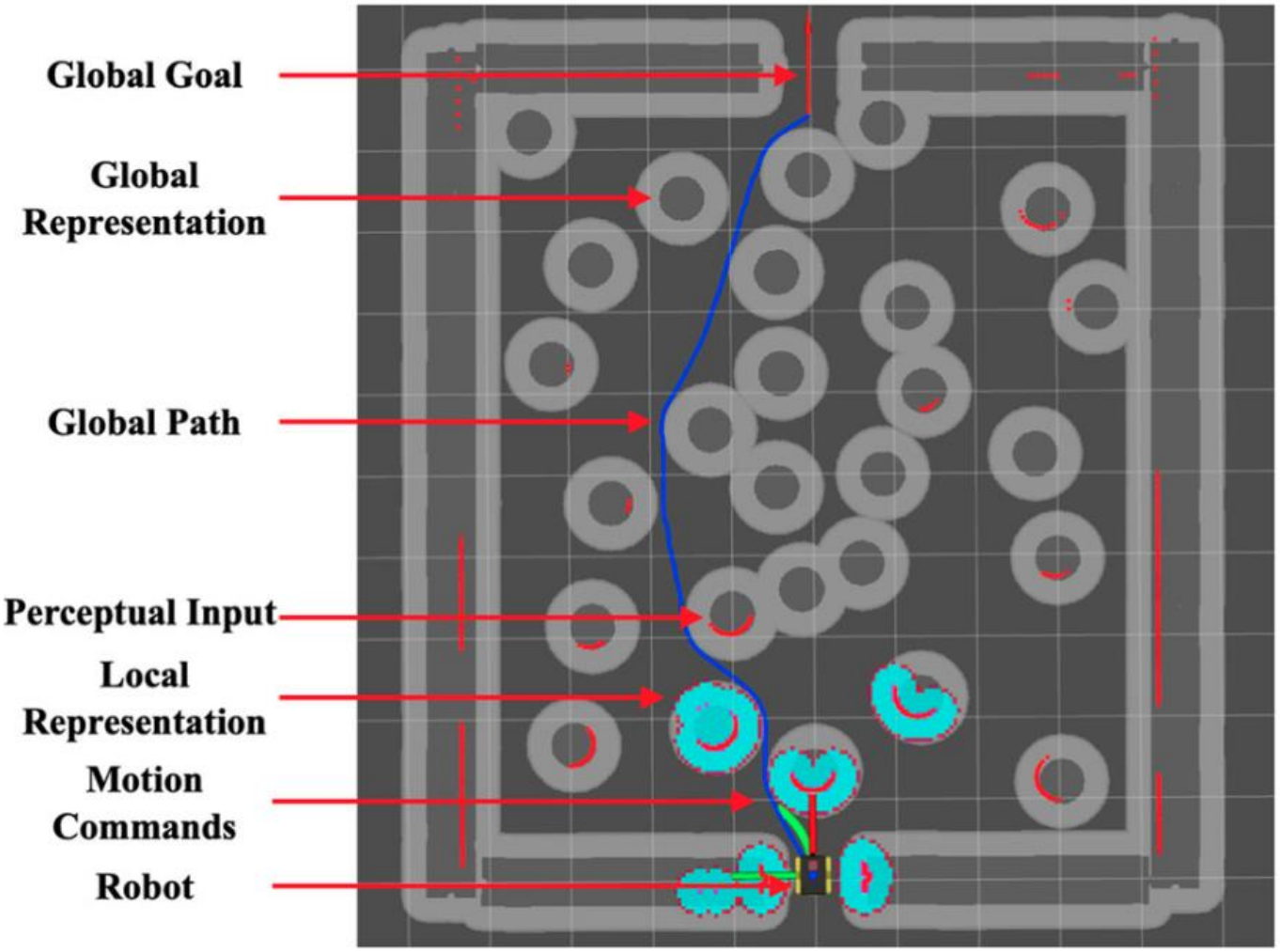
We can organize modules of the robot according to different hierarchies, as performing a temporal decomposition:





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

Classic navigation approach



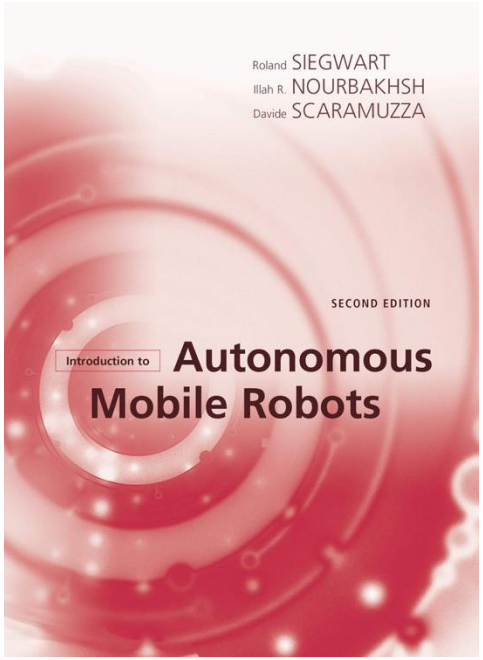
From Xiao, Stone, AURO 2023

What is missing in classic navigation approaches?

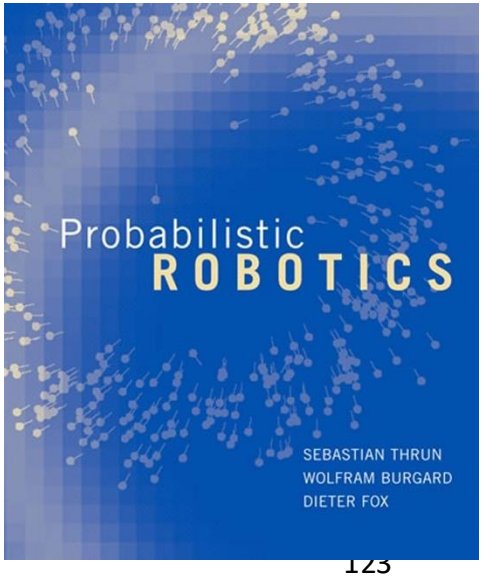
- Dynamics and time varying environment
  Classic navigation approaches have been developed with rigid and static word assumption
- Semantic understanding / Semantic mapping
  Objects and persons in navigation are usually just “obstacles”.
- Safety
  No failsafe methods to recover from errors.
  No guarantees of safety constraints wrt the robot itself or the environment (we - humans – have them)
- Learning from experience – and forgetting and remembering
  Classic navigation stack performances do not involve learning from experience
- Active Learning – robot can be active interacting with the environment
- Resource-constrained platforms
  Robot should be cheap; sensors and GPU/CPU are not (price/battery/computational costs)

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
2025-26
Università degli Studi di Milano

Introduction to Autonomous Mobile Robotics

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