Module 4

Module - 4 (Localization and Mapping)

Position and Orientation - Representing robot position. Basics of reactive navigation; Robot Localization, Challenges in localization - An error model for odometric position estimation

Map Representation - Continuous representations - Decomposition strategies - Current challenges in map representation. Probabilistic map-based localization (only Kalman method), Autonomous map building, Simultaneous localization and mapping (SLAM) - Mathematical definition of SLAM - Visual SLAM with a single camera - Graph-based SLAM - Particle filter SLAM - Open challenges in SLAM

ROBOT LOCALIZATION

- Robot localization is the process by which a robot determines its own position and orientation within an environment. This is a fundamental capability for autonomous robots to navigate and operate effectively in their surroundings. Localization allows robots to understand where they are relative to their surroundings, enabling them to plan paths, avoid obstacles, and interact with the environment in various ways.

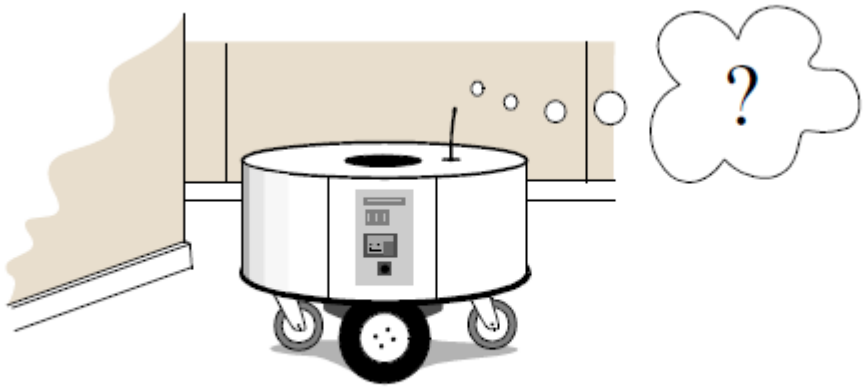


Figure 5.1
Where am I?

- Effective localization is crucial for tasks such as autonomous navigation, exploration, and manipulation in robotics applications ranging from self-driving cars to warehouse robots to drones.

- There are several approaches to robot localization, including:

- **Odometry**
- **Beacon-based Localization**
- **Simultaneous Localization and Mapping (SLAM)**
- **Particle Filters**
- **Kalman Filters**

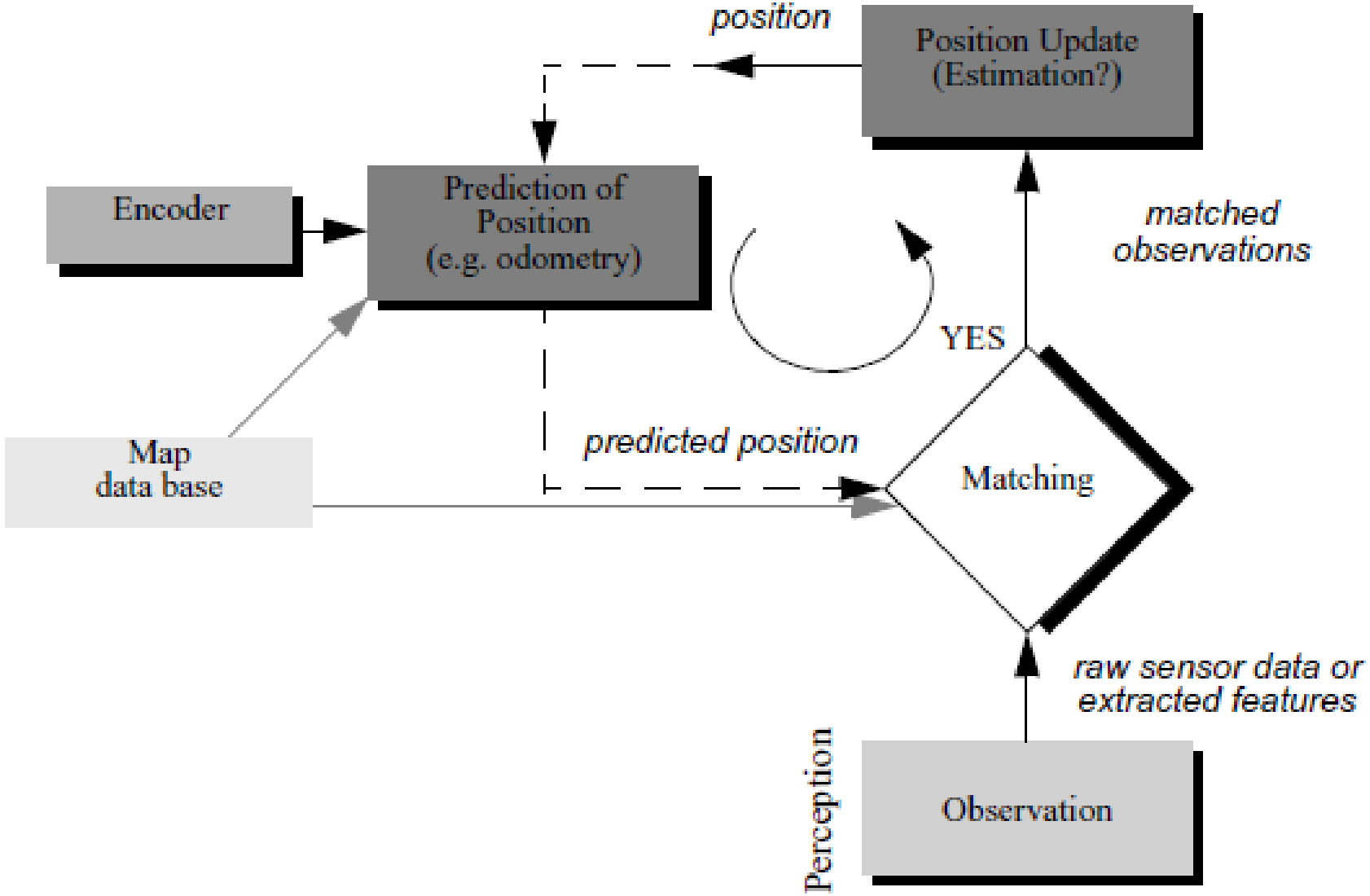


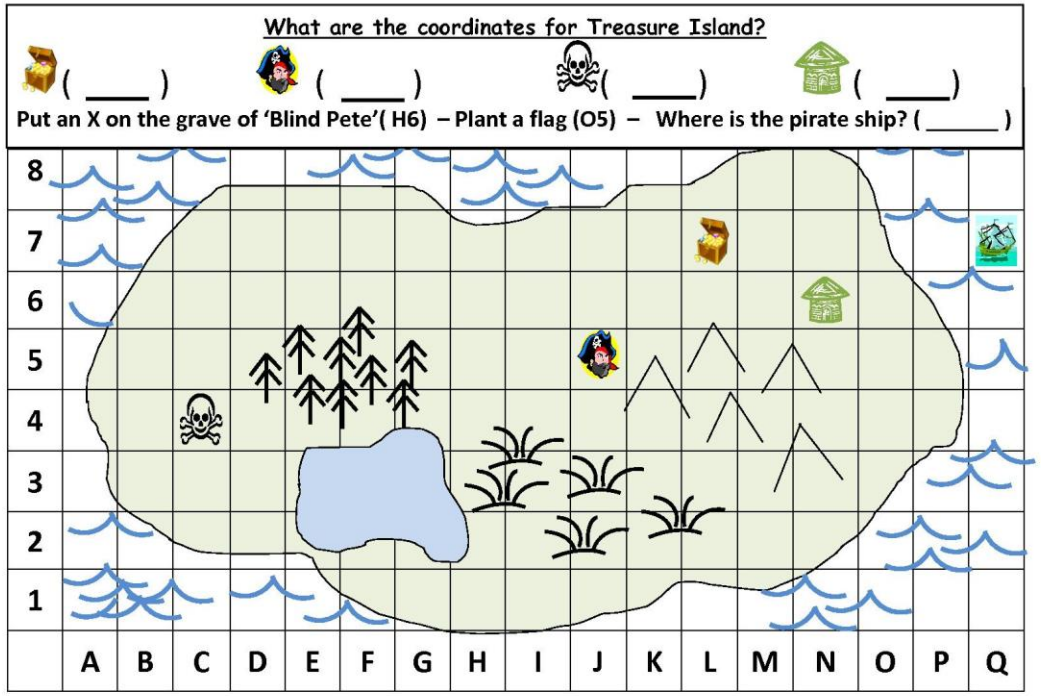
Figure 5.2
General schematic for mobile robot localization.

- **Sensors**: The robot is equipped with various sensors to perceive its surroundings. These sensors may include:
  - **Wheel Encoders**: These sensors measure the rotation of the robot's wheels, allowing for estimation of distance traveled.
  - **Inertial Measurement Unit (IMU)**: An IMU consists of sensors like accelerometers and gyroscopes to measure the robot's acceleration and angular velocity
  - **Range Sensors**: Sensors like ultrasonic sensors, LIDAR (Light Detection and Ranging), or depth cameras provide information about the distances to nearby obstacles.
  - **Vision Sensors**: Cameras capture visual information from the environment.
- In robotics**, odometry** refers to the method of estimating a robot's position and orientation by tracking its movement using sensors, typically wheel encoders. Wheel encoders are sensors that measure the rotation of a robot's wheels, providing information about the distance traveled and the direction of movement.
- **Map** representation refers to how the environment is encoded and stored in a format that the robot can understand and utilize for localization and navigation tasks.

- Ex: **grid map** representation for a mobile robot navigating indoors
- The map is typically created and maintained by the robot as it navigates through its environment.
- There are situations in which humans provide maps to robots.

The process of **matching** in robot localization refers to the comparison of sensor measurements obtained by the robot with the information stored in its map or model of the environment. This comparison aims to determine the robot's current position and orientation (pose) within the map.

- **Pose Update**: This update involves adjusting the robot's position and orientation within the environment to align with the matched features.



The Challenges of Localization

- "The Challenge of Localization" refers to the difficulty of accurately determining the position and orientation of a robot within its environment.
- Clearly, the robot's sensors play an integral role in the localization.
- Because of the inaccuracy and incompleteness of these sensors, localization poses difficult challenges.

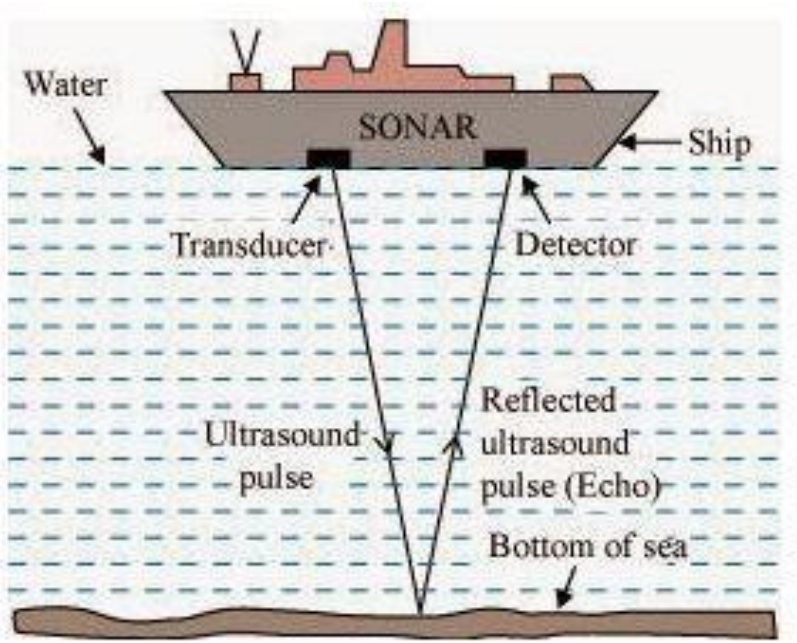
- **Sensor noise**
- **Sensor aliasing**
- **Effector noise**

Sensor noise

- Sensors are the fundamental robot input for the process of *perception*
- *Sensor noise* induces a limitation on the consistency of sensor readings in the same environmental state
- Illumination dependence is one example of the apparent noise in a vision-based sensor system.
- [Imagine a robot using a camera to navigate inside a building. The camera looks at colors to help the robot figure out where it is. But when the sun is covered by clouds, the light in the building changes because of the windows. This makes the camera's color readings change randomly, so the robot might get wrong or confusing information. To fix this, the robot needs to know if the sun is out or if clouds are blocking it, so it can adjust and make sense of the color data..]
- Picture jitter, signal gain, blooming, and blurring are all additional sources of noise, potentially reducing the useful content of a color video or image.

- **"Picture jitter"** refers to a type of distortion or instability in a video or image where the visual content appears to shake or oscillate rapidly.
- **Blooming** ,spreading of light beyond its intended boundaries.
- Sonar stands for Sound Navigation and Ranging.
- Sonar works on the principle of sending out sound waves into the water, which then bounce off objects and return as echoes. By measuring the time taken for the echo to return and the characteristics of the returned signal, sonar systems can determine the distance, size, and shape of underwater objects.

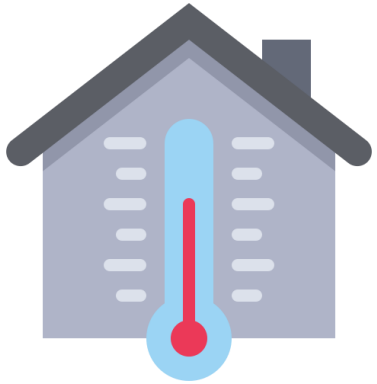
- When a sonar transducer emits sound toward a relatively smooth and angled surface, much of the signal will coherently reflect away, failing to generate a return echo, then the sonar will fail to detect the object. The result can be dramatically large errors.



- In conclusion, sensor noise reduces the useful information content of sensor readings. Clearly, the solution is to take multiple readings into account.

Sensor aliasing

- Imagine you have a digital thermometer that takes a measurement every second. This thermometer can measure temperatures up to 100 degrees Celsius. Now, suppose you place this thermometer in a room where the temperature changes very rapidly. The thermometer might not be able to keep up with these rapid changes because it only takes a measurement once every second.



- Sensor aliasing happens when a sensor misses or mixes up information because it's not fast enough to keep up with the changes in what it's sensing. It's like trying to take a picture of a fast-moving object with a slow camera. you might end up with a blurry or distorted image because the camera couldn't capture all the details quickly enough.

- The problem posed to navigation because of sensor aliasing is that, even with noise-free sensors, the amount of information is generally insufficient to identify the robot's position from a single-percept reading. Thus techniques must be employed by the robot programmer that base the robot's localization on a series of readings and, thus, sufficient information to recover the robot's position over time.

Effector noise

- robot effectors are what enable robots to interact with and manipulate their surroundings, allowing them to carry out a wide range of tasks across different industries and applications.
- End effector noise in robotics refers to unwanted variations or disturbances that affect the performance or accuracy of the end effector during its operation.

1. Mechanical wear and tear: Over time, mechanical components of the end effector, such as gears, bearings, or joints, can degrade, leading to increased friction which can introduce noise into the system.
2. Vibrations: External vibrations from the environment or other machinery can cause oscillations or disturbances in the end effector, affecting its stability and accuracy.

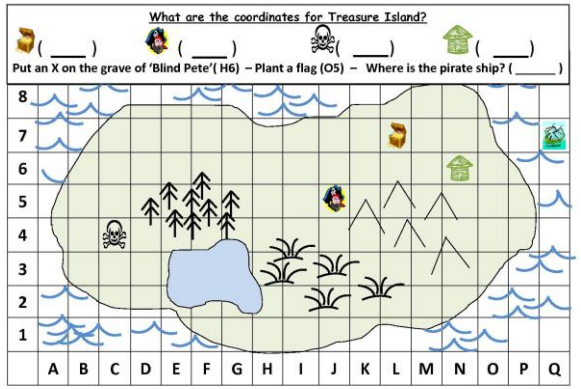
There are many sources of odometric error

- Misalignment of the wheels (deterministic);
- Uncertainty in the wheel diameter and in particular unequal wheel diameter (deterministic);
- Variation in the contact point of the wheel;
- Unequal floor contact (slipping, nonplanar surface, etc.).

Map Representation

- a "map" refers to a representation of the environment in which the robot operates.

- **Grid Map:** This is a common representation where the environment is divided into a grid of cells, and each cell is marked as occupied, free based on sensor data. grid maps are often used for navigation and obstacle avoidance.



- The problem of representing the environment in which the robot moves is a dual of the problem of representing the robot's possible position or positions.

- as the robot moves around and explores its surroundings, it not only creates or updates a map of the environment based on what it senses (like obstacles or landmarks) but also continuously improves its understanding of where it is located in that environment.

Three fundamental things must be important when choosing a particular map representation:

1. The map should be precise.
2. The map needs to be detailed enough for the robot to reach its goals effectively.
3. The map's details should align with what the robot's sensors can detect.
   - If the robot has a LIDAR sensor, the map it creates should reflect the positions of walls, obstacles, or other surfaces that the LIDAR can measure.
   - If the robot uses cameras, the map might include visual details that the camera picks up, like colors, shapes, or landmarks.

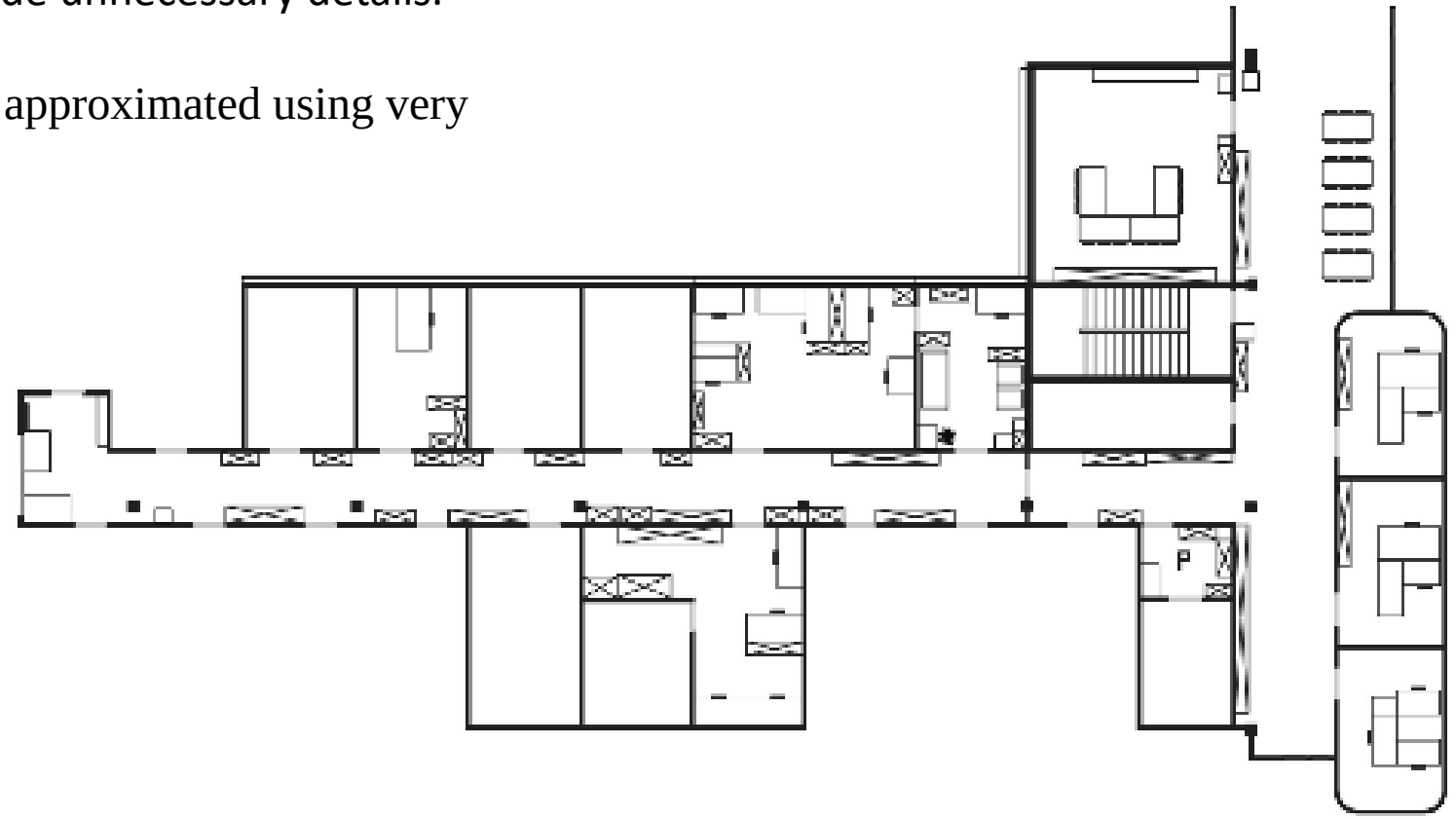
4. The more detailed the map, the more complex it is to use for tasks like mapping, finding its location, and navigating.

Continuous representations

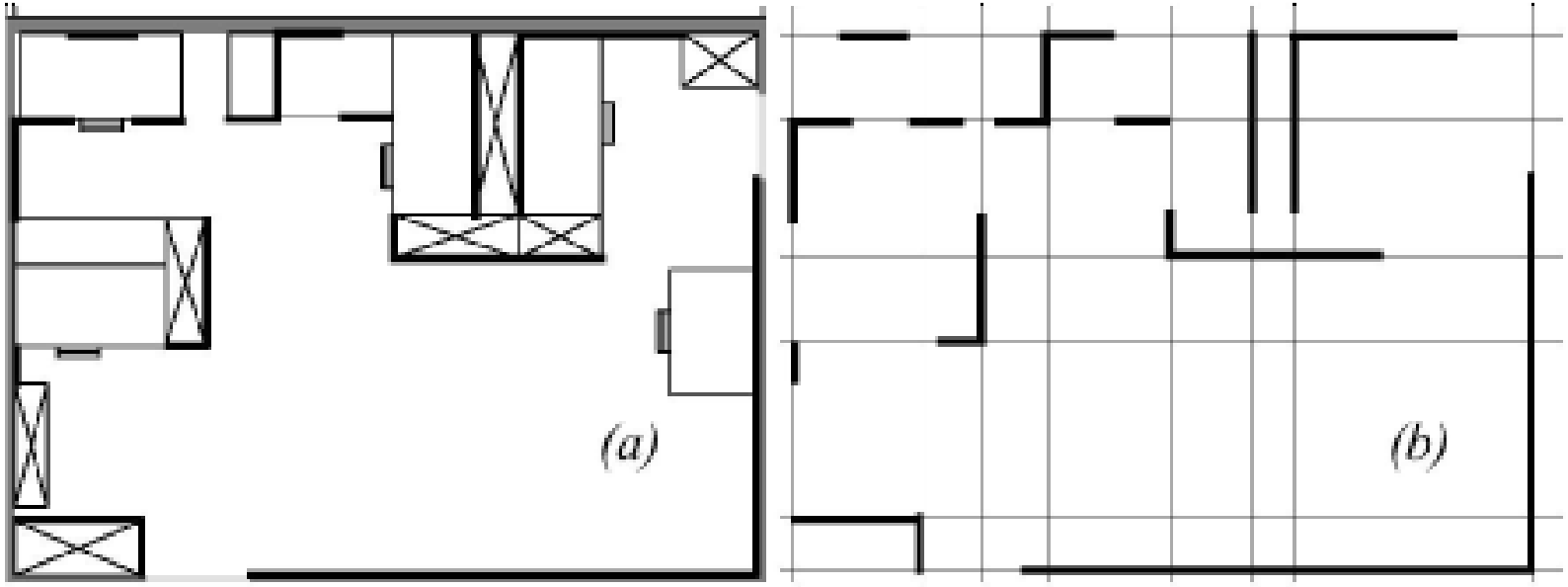
- A continuous-valued map is like a detailed drawing of the environment, where everything is represented with exact measurements and positions in a smooth, continuous way.
- it's like having a map that shows every little detail of the surroundings without any grid-like structure.
- So, in this method, the features of the environment are precisely labeled and located in a smooth, continuous space rather than being divided into separate grid cells.
- When people make maps for robots to use, they often only include important things that the robot's sensors can detect.
- For example, if a robot can't detect the color of objects, there's no point in including colors on the map.
- They also simplify the details of what they include.
- For example, instead of drawing every little detail of an object, they might just draw a simple shape that represents it well enough for the robot to understand where it is.

- a mobile robot map can further reduce memory usage because you don't need to include unnecessary details.

- For example, all objects may be approximated using very simple convex polygons



A continuous representation using polygons as environmental obstacles.



- One excellent example involves line extraction.

- when the robot makes a map, it doesn't draw every single point it sees with its laser sensor. Instead, it draws lines to represent where the walls and corners are.
- This way, the map is simpler and easier to understand, matching the kind of information the robot gets from its sensor.

- Imagine a robot exploring inside a building. It has a special sensor that shoots out laser beams to measure distances to objects around it. This sensor gives the robot a lot of data about the surfaces it sees, like walls and corners.

- In summary, the key advantage of a continuous map representation is the potential for high accuracy and expressiveness with respect to the environmental configuration as well as the robot position within that environment.

Decomposition strategies(in maps)

- Imagine you're making a map of a city. You could include every single detail, like every tree, building, and road. But that would make the map huge and complicated.
- Instead, you might decide to simplify things. You could group together similar things, like all the buildings in one area, all the parks in another, and so on. This is called abstraction.
- It means you're breaking down the real world into simpler, more general features.
- But there's a trade-off. When you simplify the map like this, you lose some details. The map won't look exactly like the real world. It might not show every little street or building.
- So, why would you do this? Well, because it makes the map easier to understand and use. It's like drawing a cartoon version of the city instead of a photograph. It might not be as detailed, but it gets the main idea across more clearly and quickly.

Exact cell decomposition.

- Imagine you have a big area with a bunch of obstacles like walls or furniture, and you want to make a map for a robot to navigate through it. Instead of drawing every single detail of where each obstacle is, you can simplify things by dividing the area into smaller sections where there's no obstacle, which we call "free space".

- In the map, each of these sections is represented as a single point or node. So, instead of needing to remember the exact shape and position of every obstacle, you only need to remember where the free space is and how it's connected.
- The idea behind this is that as long as the robot can move from one free space area to another, it doesn't really matter exactly where it is within each area.
- However, this approach only works well if the robot doesn't need to know its precise position within each free space area. If it does need that level of detail, then this kind of simplified map might not be the best choice.

Fixed decomposition

- Imagine you have a map of an area with obstacles, like walls or furniture. Instead of trying to represent every little detail of the obstacles, you decide to divide the area into smaller squares, kind of like a grid.

- Each square is either filled with an obstacle or empty space. This makes it easier to understand and work with the map, especially for a robot trying to navigate through it.
- However, there's a downside to this approach. Sometimes, narrow passages or small spaces might get overlooked or merged with nearby squares during this grid-like division. So, while this method is helpful, it's not perfect. Some details might get lost in the process, making it harder for the robot to navigate accurately.
- So, in simple terms, fixed decomposition turns the continuous real world into a simplified grid-like map. It's helpful, but it's not perfect

Adaptive cell decomposition

- This approach adjusts the division of the environment based on the complexity of the surroundings or the needs of the task.
- Instead of using a fixed grid or predefined divisions, it dynamically divides the space into smaller areas as needed.
- For example, in a simple environment with few obstacles, the division might be coarse, with larger areas. But in a complex environment with many obstacles or intricate paths, the division might be finer, with smaller areas to capture more details.
- Adaptive decomposition helps balance the level of detail in the map with the computational resources needed to process it.
- Think of it like zooming in or out on a map depending on how much detail you need in different areas.

Topological Decomposition:

- This method focuses on capturing the connectivity and relationships between different areas of the environment rather than their exact shapes or sizes.
- Instead of representing the environment with precise geometric shapes, it represents it in terms of how different areas are connected or related to each other.
- For example, in a building, topological decomposition might represent rooms as nodes and doors or hallways as connections between them, without worrying too much about the exact shapes of the rooms or corridors.
- Topological decomposition is often used when the exact geometry of the environment is less important than understanding how to move between different areas.
- It simplifies the map while preserving essential information about how to navigate through the environment.
- Think of it like drawing a simplified network diagram of a building's layout, focusing on the connections between rooms rather than their exact shapes.

Current challenges in map representation

1. **Dynamic Environments**: Maps need to adapt to changes in the environment, such as moving objects, newly constructed obstacles, or changes in lighting conditions. Dynamic environments require real-time updating of maps to ensure accuracy for tasks like navigation or object recognition.
2. **Semantic Understanding**: Beyond geometric representation, maps need to incorporate semantic information about objects and their relationships. This means understanding not just where objects are located but also what they are and how they interact with each other. Achieving robust semantic understanding remains a challenge due to the complexity and variability of real-world scenes.
3. **Scale and Complexity**: As environments become larger and more complex, mapping algorithms must scale accordingly to handle the increased amount of data. This includes efficient data storage, processing, and retrieval to maintain real-time performance.
4. **Multi-Modal Fusion**: Integrating data from various sensors, such as cameras, LiDAR, and radar, into a coherent map representation remains challenging. Each sensor provides different types of information with varying levels of accuracy and reliability, requiring sophisticated fusion techniques to create a comprehensive map.

5. **Long-Term Autonomy**: Enabling robots to operate autonomously over extended periods without human intervention requires robust and reliable map representations. This includes handling long-term localization, map maintenance, and adaptation to changing environmental conditions over time.
6. **Uncertainty and Robustness**: Dealing with uncertainty in sensor measurements and environmental models is crucial for building robust map representations. This involves probabilistic methods to model uncertainty and incorporate it into decision-making processes for navigation and planning.
7. **Privacy and Ethics**: With the increasing use of mapping technologies in public spaces and private domains, concerns about privacy and ethical considerations arise. Developing map representations that balance the need for detailed information with privacy concerns is an ongoing challenge.

Odometry

- In robotics, "odometry" refers to the process of estimating a robot's change in position over time by utilizing data from its motion sensors, like wheel encoders, to calculate how far it has traveled based on its wheel rotations

An error model for odometric position estimation

- "An error model for odometric position estimation" refers to the concept of describing or quantifying errors that can occur when using **odometry** to estimate the position of a robot, vehicle, or other mobile system.

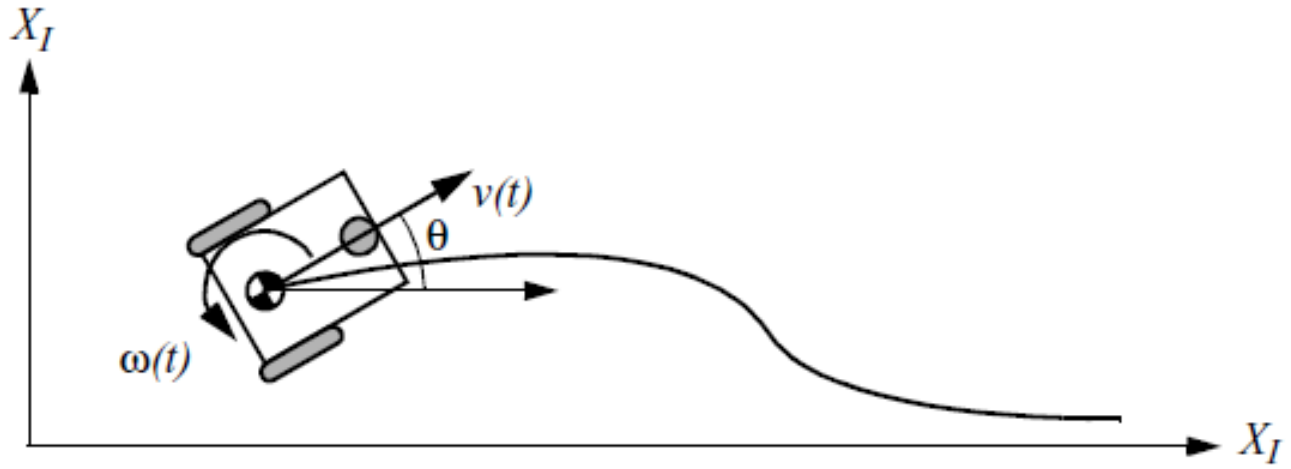


Figure 5.3
Movement of a differential-drive robot.

5.2.4 An error model for odometric position estimation

Generally the pose (position) of a robot is represented by the vector

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{5.1}$$

For a differential-drive robot the position can be estimated starting from a known position by integrating the movement (summing the incremental travel distances). For a discrete system with a fixed sampling interval Δt the incremental travel distances $(\Delta x;\Delta y;\Delta\theta)$ are

$$\Delta x = \Delta s \cos(\theta + \Delta\theta/2) \tag{5.2}$$

$$\Delta y = \Delta s \sin(\theta + \Delta\theta / 2) \tag{5.3}$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \tag{5.4}$$

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \tag{5.5}$$

where

$(\Delta x;\Delta y;\Delta\theta)$ = path traveled in the last sampling interval;

$\Delta s_r;\Delta s_l$ = traveled distances for the right and left wheel respectively;

b = distance between the two wheels of differential-drive robot.

Thus we get the updated position p' :

$$p' = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = p + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta/2) \\ \Delta s \sin(\theta + \Delta\theta/2) \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta/2) \\ \Delta s \sin(\theta + \Delta\theta/2) \\ \Delta\theta \end{bmatrix} \tag{5.6}$$

By using the relation for $(\Delta s;\Delta\theta)$ of equations (5.4) and (5.5) we further obtain the basic equation for odometric position update (for differential drive robots):

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \tag{5.4}$$

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \tag{5.5}$$

$$p' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix} \tag{5.7}$$

- When a robot or vehicle tries to estimate its position over time (by integrating or combining sensor data), there can be some mistakes or uncertainties in this process. These "integration errors" come from small inaccuracies in how the data is combined.

- As the robot moves step by step (incrementally), there are also errors in how well it can track its motion. For example, the robot might not move exactly as expected because of slippage, uneven surfaces, or sensor limitations.

In the next step we will establish an error model for the integrated position p' to obtain the covariance matrix $\Sigma_{p'}$ of the odometric position estimate. To do so, we assume that at the starting point the initial covariance matrix Σ_p is known. For the motion increment $(\Delta s_r;\Delta s_l)$ we assume the following covariance matrix Σ_Δ:

$$\Sigma_\Delta = covar(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix} \tag{5.8}$$

where Δs_r and Δs_l are the distances traveled by each wheel, and k_r, k_l are error constants representing the nondeterministic parameters of the motor drive and the wheel-floor interaction. As you can see, in equation (5.8) we made the following assumptions:

- The two errors of the individually driven wheels are independent[5];
- The variance of the errors (left and right wheels) are proportional to the absolute value of the traveled distances $(\Delta s_r;\Delta s_l)$.

If we assume that p and $\Delta_{rl} = (\Delta s_r; \Delta s_l)$ are uncorrelated and the derivation of f [equation (5.7)] is reasonably approximated by the first-order Taylor expansion (linearization),

$$\Sigma_{p'} = \nabla_p f \cdot \Sigma_p \cdot \nabla_p f^T + \nabla_{\Delta_{rl}} f \cdot \Sigma_\Delta \cdot \nabla_{\Delta_{rl}} f^T$$

$F_p = \nabla_p f$ and $F_{\Delta_{rl}} = \nabla_{\Delta_{rl}} f$:

$$F_p = \nabla_p f = \nabla_p (f^T) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s \sin(\theta + \Delta\theta/2) \\ 0 & 1 & \Delta s \cos(\theta + \Delta\theta/2) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

$$F_{\Delta_{rl}} = \begin{bmatrix} \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \tag{5.11}$$

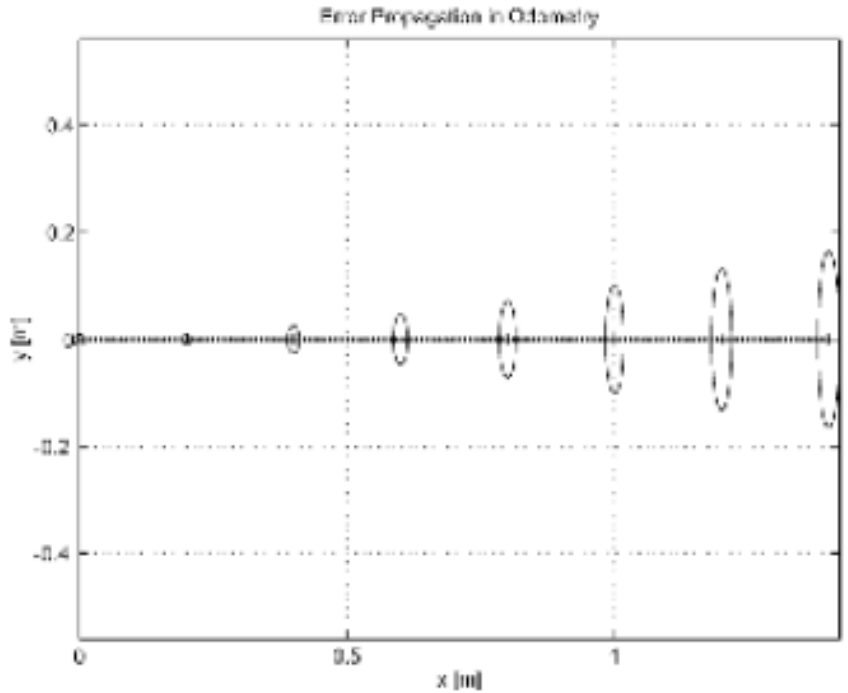


Figure 5.4
Growth of the pose uncertainty for straight-line movement: Note that the uncertainty in y grows much faster than in the direction of movement. This results from the integration of the uncertainty about the robot's orientation. The ellipses drawn around the robot positions represent the uncertainties in the *x,y* direction (e.g. 3σ). The uncertainty of the orientation θ is not represented in the picture although its effect can be indirectly observed.

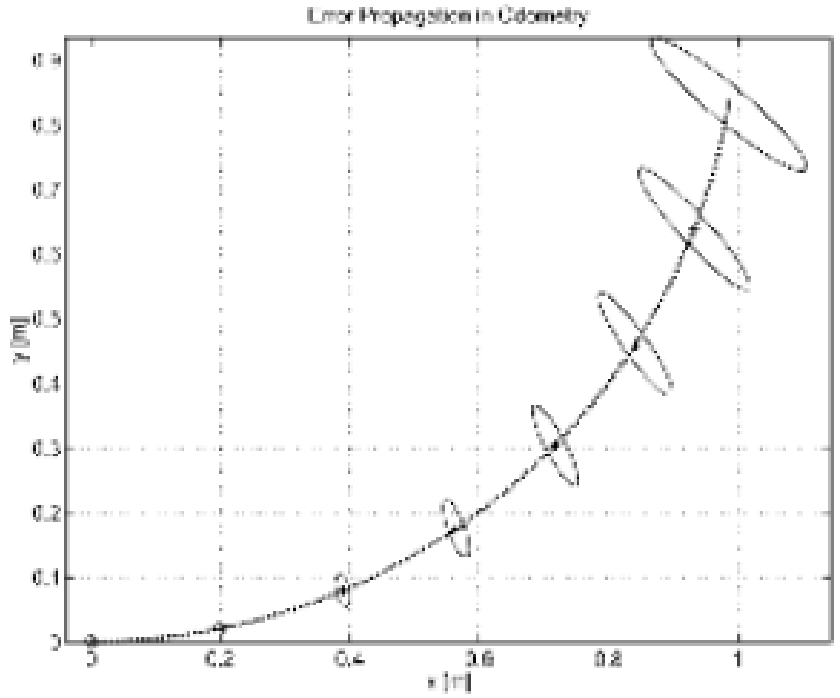


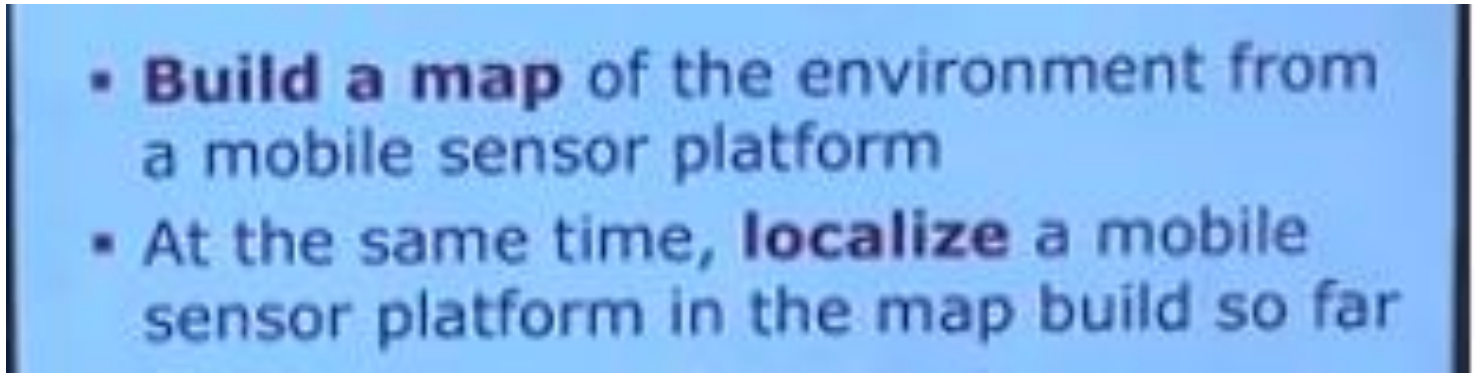
Figure 5.5
Growth of the pose uncertainty for circular movement (r = const): Again, the uncertainty perpendicular to the movement grows much faster than that in the direction of movement. Note that the main axis of the uncertainty ellipse does not remain perpendicular to the direction of movement.

SLAM

Simultaneous Localization and Mapping

Introduction

'SLAM' refers to the method of simultaneous localization (i.e. position/orientation) of some sensor with respect to its surroundings while at the same time mapping the environment. This is an active area of research in the fields of robotics and autonomous systems.



Example….

- Imagine you're playing a treasure hunt game in a big, dark maze. You have a special map, but it's blank at first. As you walk around with your flashlight, you draw the walls and paths on the map and mark where you are. That's like what SLAM does for robots or phones. It helps them create a map of where they are and what's around them as they move, just like you drawing the map in the maze. So, they can find their way and not get lost, just like you finding your way to the treasure.

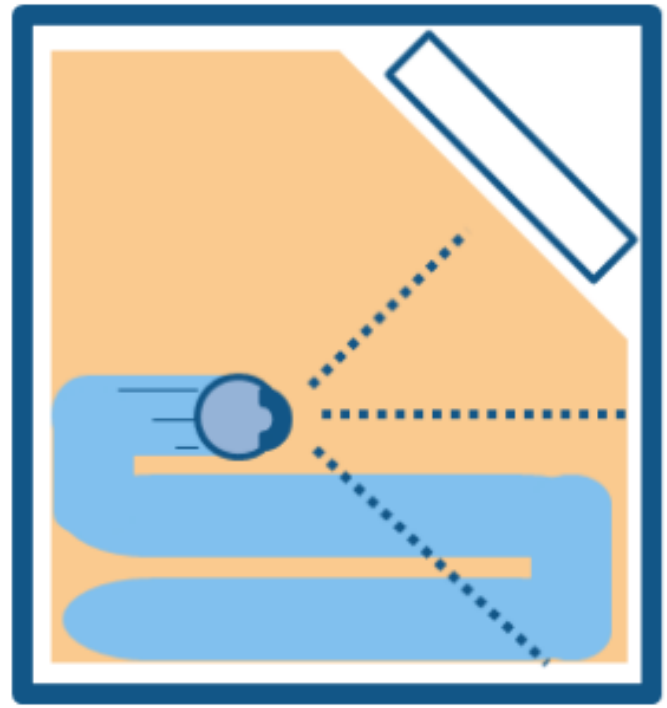
- Imagine a blind person navigating through a new city without a cane or a guide dog. They use a special device equipped with sensors that can detect obstacles and map out their surroundings in real-time. As they walk, the device builds a map of the environment and tracks their location within it, much like SLAM. This helps them safely navigate unfamiliar places, avoiding obstacles and finding their way independently. So, just like SLAM helps robots "see" and move around, it can also assist blind individuals in understanding and traversing their surroundings.

SLAM is Relevant

- It is considered a fundamental problem for truly autonomous robots
- SLAM is the basis for most navigation systems

autonomous navigation

Without SLAM:
Cleaning a room randomly.



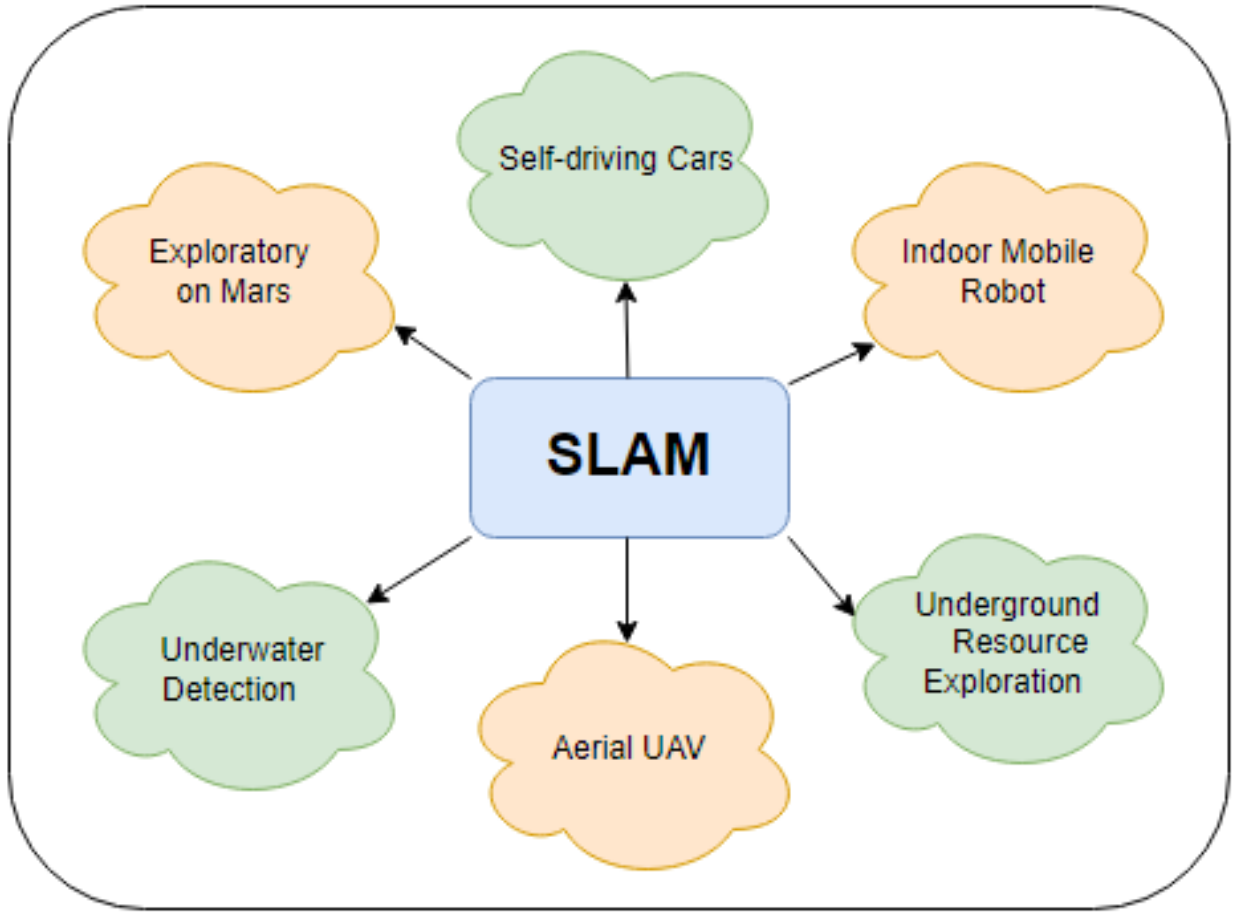
With SLAM:
Cleaning while understanding the room's layout.

SLAM Applications

- SLAM is central to a range of indoor, outdoor, air and underwater applications for both manned and autonomous vehicles.

Examples:

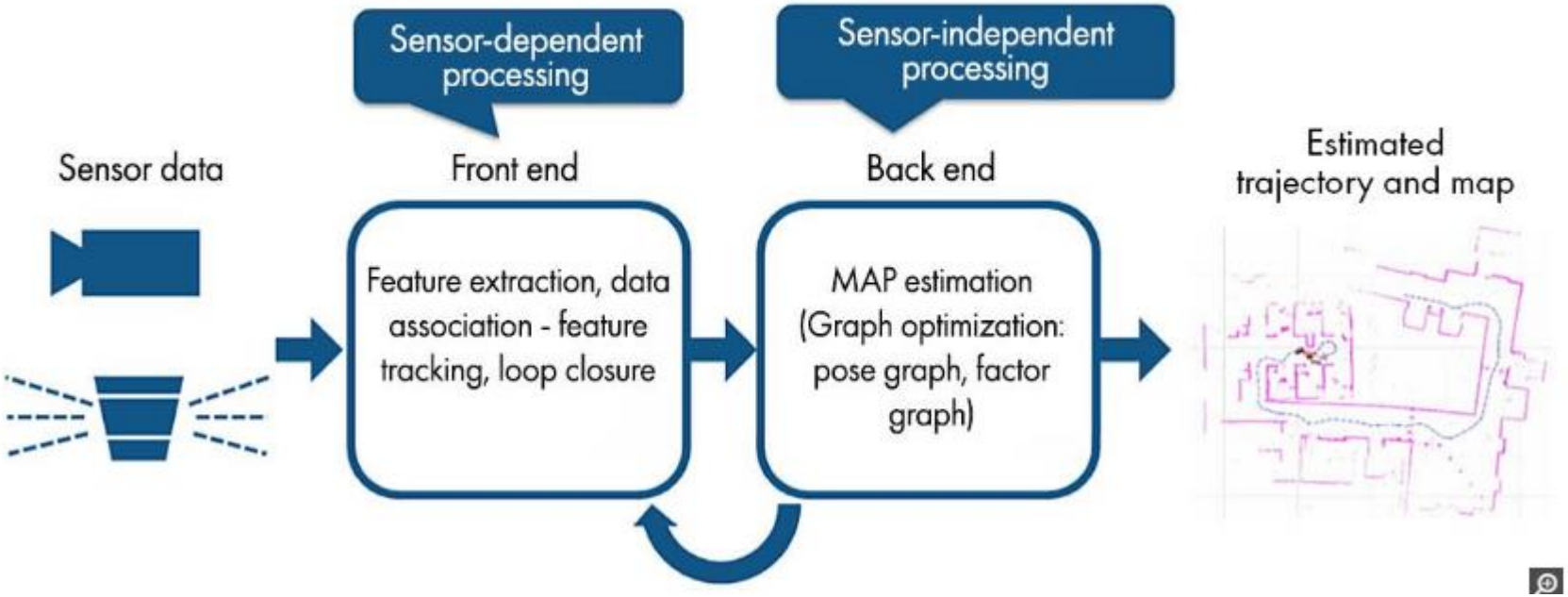
- At home: vacuum cleaner, lawn mower
- Air: surveillance with unmanned air vehicles
- Underwater: reef monitoring
- Underground: exploration of mines
- Space: terrain mapping for localization



Applications of SLAM

How SLAM Works

Broadly speaking, there are two types of technology components used to achieve SLAM. The first type is sensor signal processing, including the front-end processing, which is largely dependent on the sensors used. The second type is pose-graph optimization, including the back-end processing, which is sensor-agnostic.



Visual SLAM

Introduction

- Visual Simultaneous Localization and Mapping (SLAM) is a cutting-edge technology that combines Computer Vision, artificial intelligence, and robotics to enable machines to perceive and navigate unknown environments. It allows robots or autonomous systems to understand their surroundings, build a map of the environment, and simultaneously determine their own position within that map.

- There are several different types of SLAM technology, some of which don’t involve a camera at all. Visual SLAM is a specific type of SLAM system that leverages 3D vision to perform location and mapping functions when neither the environment nor the location of the sensor is known.

- Traditional SLAM methods typically rely on range sensors like Lidar or sonar, which provide accurate distance measurements but lack detailed visual information. Visual SLAM, on the other hand, utilizes cameras as the primary sensor, allowing for richer and more detailed perception of the environment.
- Visual SLAM algorithms extract visual features from camera images, track these features over time, and use them to estimate the camera's motion and generate a map of the environment. Key challenges in Visual SLAM include robust feature detection and matching, handling occlusions, dealing with dynamic objects, and managing scale ambiguity.

Visual SLAM

- As the name suggests, visual SLAM (or vSLAM) uses images acquired from cameras
- Visual SLAM can use simple cameras, compound eye cameras, and RGB-D cameras.
- Visual SLAM can be implemented at low cost with relatively inexpensive cameras.
- **since cameras provide a large volume of information, they can be used to detect landmarks**
- **Monocular SLAM** is a type of SLAM algorithm when vSLAM uses a single camera as the only sensor

How Visual SLAM Works

- Visual SLAM typically follows a pipeline consisting of several key steps:

1. Feature Extraction

- The first step in a visual SLAM pipeline involves extracting distinctive visual features, such as corners, edges, or blobs, from the camera images. These features serve as landmarks that can be tracked over time.

2. Feature Tracking

- **Feature Tracking** is about following specific points (called features) in a video or a sequence of images to see how they move over time. This helps us understand how the camera is moving.
- You can track features using different methods like:

1. **Optical Flow**: This looks at how pixels move between two frames and estimates the motion of the camera.
2. **Feature Matching**: This matches the same feature across different frames to track it.
3. **Direct Image Alignment**: This compares entire images to align them and find how they change over time.

- Basically, by tracking features from one frame to the next, you can figure out the camera's movement.

3. Mapping

Mapping is the process of using the tracked features to build a map of the environment and figure out where the camera is in that space.

As the camera moves and tracks features, it can:

1. Estimate its **pose** (position and orientation) in the environment.
2. Create a **map** of the surroundings.

This map can look different depending on how detailed it is:

- A **sparse point cloud** shows just a few key points in 3D space.
- A **dense occupancy grid** is like a grid where each cell is filled with information about whether it's occupied or empty.

4. Loop Closure

- To improve the accuracy and consistency of the generated map, visual SLAM algorithms often employ loop closure techniques. Loop closure aims to detect revisited locations in the environment and correct any accumulated errors in the estimated trajectory and map.

• 5. Optimization

Optimization is the final step where the system makes everything as accurate as possible.

After detecting loop closures (when the camera revisits a place), the algorithm works to fix any remaining errors in the map and camera movement. It does this by comparing where the features should be (based on the camera's estimated path) with where they actually are in the images. Then, it adjusts everything to minimize the differences.

6. Localization

With the optimized map and trajectory, visual SLAM algorithms can provide real-time localization estimates, enabling the camera or robot to understand its position within the environment.

Graph-Based SLAM

- Graph-based SLAM (also known as Graph SLAM) uses a graph to represent the environment and the robot's pose estimates. It is widely used in many robotics applications like autonomous vehicles and mobile robots.

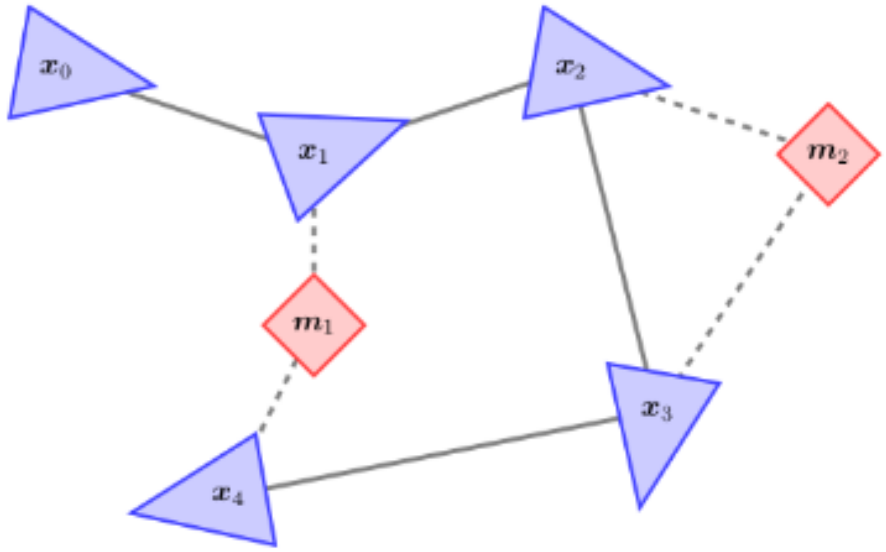


Fig 1: Graph SLAM illustration in 2D. The blue triangles represent robot poses, and the red diamonds the landmarks positions. The solid lines represent the robot motion and the dashed lines the robot measurements [source]

Key Components of Graph SLAM

- **Graph representation:** The environment and robot's pose estimates are represented as a graph, where each node represents a pose estimate, and each edge represents a measurement or a motion estimate between two poses. The graph is a mathematical model that allows us to represent and manipulate the information from the robot's sensors and control inputs. The nodes in the graph represent the robot's position and orientation in the environment, and the edges represent the constraints between these poses.

- **Motion model:** The motion model is a mathematical model that predicts the pose of the robot based on previous pose estimate and control input obtained from sensors like wheel encoders or accelerometers. The motion model takes into account robot's kinematics and dynamics to estimate the robot's position and orientation in the environment.

- **Sensor model:** The sensor model is a mathematical model that predicts the expected sensor measurement given the current pose estimate and the environment's map. The sensor model is used to evaluate the consistency between the sensor measurements and the predicted measurements. If there is a mismatch between the predicted and actual measurements, the optimization algorithm adjusts the pose estimate to reduce the error.

- **Optimization algorithm**: The optimization algorithm estimates the robot's pose and the environment's map by minimizing the error between the predicted and actual measurements. The error is represented as a cost function, which is minimized using optimization techniques such as gradient descent or Gauss-Newton. The cost function is a sum of the errors between the predicted and actual measurements, and it includes terms that account for the uncertainty in the sensor measurements and the motion estimates.

Algorithm Break-down

Graph SLAM algorithm involves the following steps:

1. Map initialization: The map is initialized with some prior knowledge or assumptions about the environment. The robot's initial pose estimate is also obtained through some means, such as GPS, odometry, or motion sensors. Map initialization is important to provide a starting point for the optimization algorithm and to reduce the uncertainty in the robot's pose estimate.
2. Graph construction: The robot moves in the environment and takes sensor measurements, which are used to construct the graph. Each node in the graph represents a pose estimate, and each edge represents a measurement or a motion estimate between two poses.

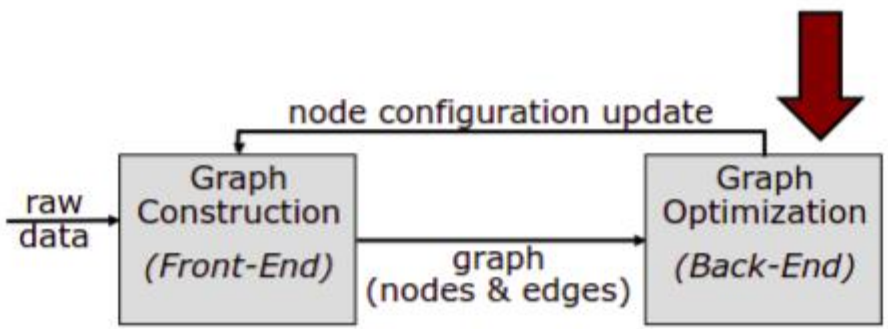


Fig 3: Front end and Back end of Graph SLAM [source]

3. Loop closure detection: As the robot moves, it may revisit a previously visited location. This is known as a loop. Loop closure detection detects loops in the robot's trajectory and adds new edges to the graph, connecting the previously visited pose with the current pose estimate. Loop closure detection is important to reduce the accumulation of errors in the robot's pose estimate, which can cause the robot to lose track of its position in the environment.
4. Optimization: The optimization algorithm estimates the robot's pose and the environment's map by minimizing the error between the predicted and actual measurements. This is achieved by iteratively adjusting the pose estimates and map parameters until the error is minimized.
5. Map update: The updated map is used to improve the robot's pose estimate and sensor measurements. This completes a single iteration of the Graph SLAM algorithm.
6. Repeat steps 2-5 until convergence: Steps 2-5 are repeated until the error is minimized, or a certain convergence criterion is met.

In a SLAM process, a pose-graph representation [shown in figure 4] is used to represent the environment and the robot's pose estimates. Each node corresponds to a pose estimate, and adjacent poses are connected by edges that represent spatial constraints between robot poses based on measurements. The edges connecting consecutive poses, denoted as $e_{t-1\,t}$, model odometry measurements, while the other edges represent spatial constraints resulting from multiple observations of the same part of the environment.

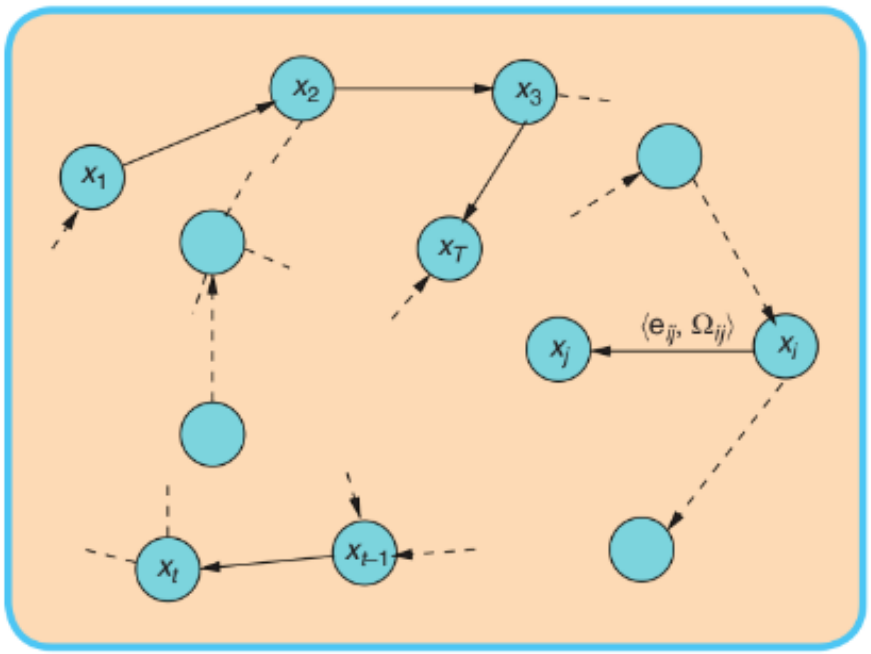


Fig 4: In a SLAM process, a pose-graph representation [source]

Open challenges in SLAM

- **Uncertainty and Noise:** Sensor measurements are inherently noisy, leading to uncertainty in both localization and mapping. Dealing with this uncertainty and effectively integrating noisy sensor data into the estimation process is a significant challenge.
- **Non-Linearity:** Many real-world environments and robot sensor models exhibit non-linear behavior. Traditional estimation techniques like Kalman filters may struggle with such non-linearities, requiring more advanced methods like particle filters or extended Kalman filters.
- **Data Association:** Associating sensor measurements with features in the environment is crucial for both localization and mapping. However, in complex environments with many features, data association can be challenging, leading to ambiguities and errors in the estimation process.
- **Computational Complexity:** SLAM algorithms often involve complex computations, especially as the size of the environment or the number of features increases. Real-time performance is essential for many robotic applications, requiring efficient algorithms and optimization techniques.
- **Sensor Fusion:** Modern robotic systems often use multiple sensors, such as cameras, LiDARs, and IMUs, to gather information about the environment. Integrating data from these heterogeneous sensors into a coherent estimation framework is challenging due to differences in sensor characteristics, coordinate frames, and noise properties.

- **Dynamic Environments**: SLAM algorithms are typically designed for static environments, but real-world environments are often dynamic, with moving objects and changing features. Handling dynamic elements in the environment while maintaining accurate localization and mapping is a significant challenge.
- **Scale and Robustness**: SLAM algorithms should be scalable to large environments while remaining robust to various environmental conditions, lighting changes, occlusions, and other challenges encountered in real-world scenarios.
- **Loop Closure Detection**: Detecting loop closures, where the robot revisits a previously visited location, is essential for correcting drift and improving map consistency. However, loop closure detection can be challenging, especially in large-scale environments with similar-looking regions.