Localization and Mapping

Contents are borrowed

from NVIDIA and Standford teaching materials

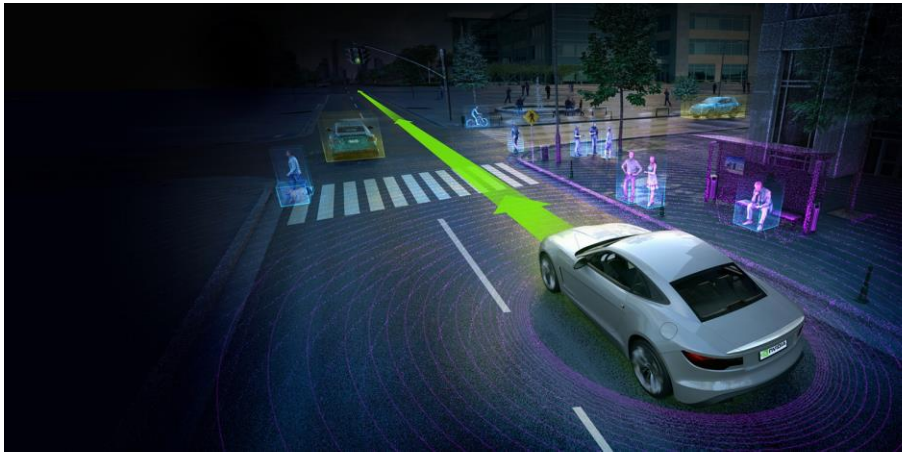
Localization

Robot Localization

- Localization is knowing a robot’s position and orientation in the world around it.

Applications of Localization

- Precision driving
- Room to room navigation

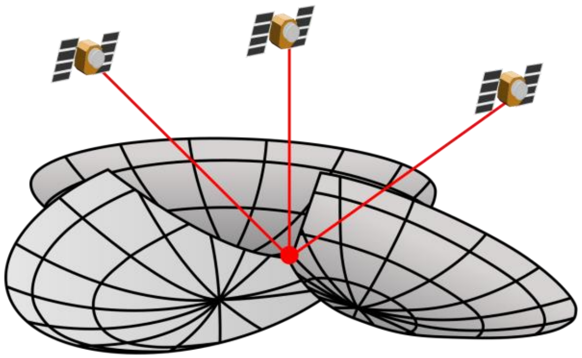


Localization and Mapping

- Localization is the task of determining a robot position
- Mapping is the task of building an accurate map of the environment
- Simultaneous Localization and Mapping (SLAM)
  - A process of dynamically building a map of the world around a robot while estimating the position of the robot within this map

Localization and Sensors

- Sensors on the robot
  - Encoders
  - Gyroscope
  - Accelerometer
  - Compass



- Active Sensors
  - Sonar
  - Laser
  - Radar
  - Camera
  - Radio
  - GPS, Wi-fi

Uncertainty in Localization

- The problem with using only odometry or motion sensors is that as the robot moves, there is noise in the motion
  - Wheels can slip on the ground
  - Wheel encoders have a finite amount of precision
  - Accumulated gyroscope readings can drift over time
- Sensors that provide readings of the surround environment also contribute to localization uncertainty
- Because of these reasons, localization often cannot be exact

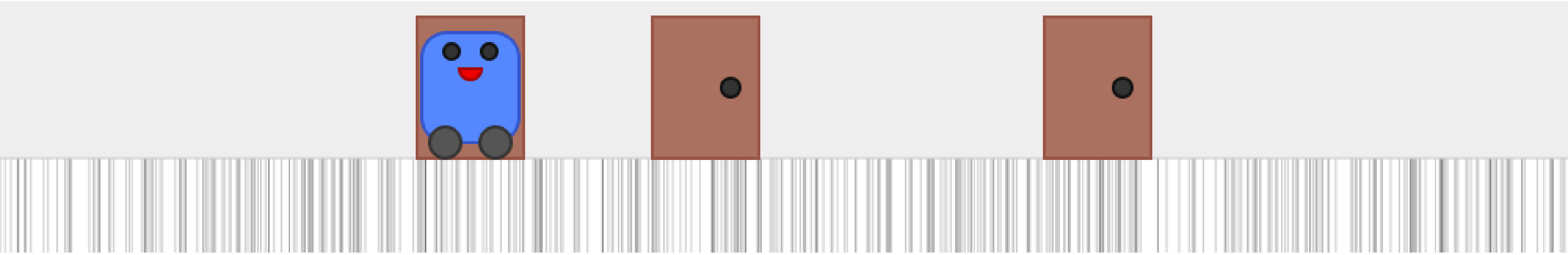
Monte Carlo Localization

- Proposed by Fox et al. as a means to estimate the position of a robot in an environment
- The belief of the robot position is modeled as set of particles over the possible state space
  - In location where there are more particles, the higher the probability that the robot is located there

Monte Carlo Localization Overview

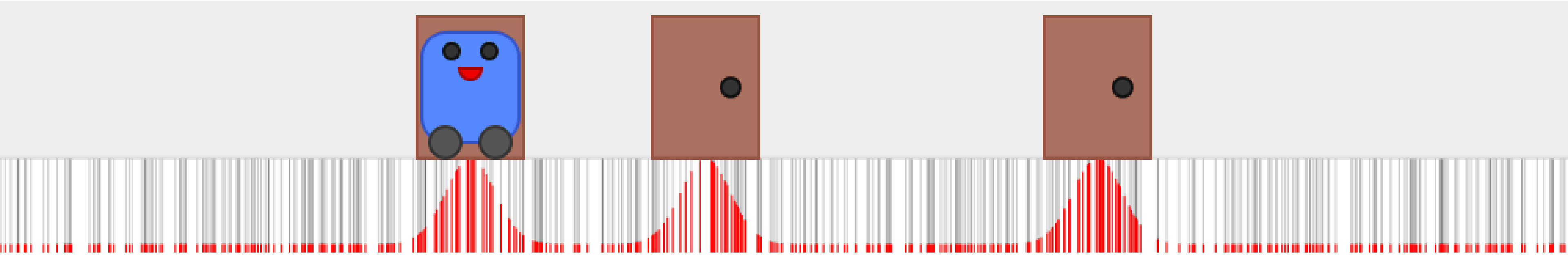
- The algorithm itself can be divided into 2 steps:
  - Prediction
    - The prediction steps involves updating the positions of all the particles given information about the motion of the robot (motion update)
  - Update
    - The update step involves computing the posterior probability of each of the particles once a new data reading has arrived (sensor update)

Monte Carlo Localization - Step #1



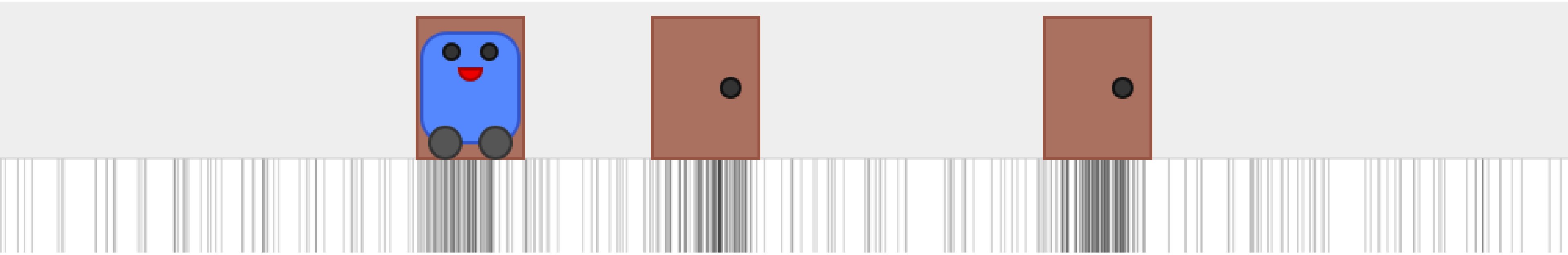
- Robot is trying to determine its position
  - There are 3 similar looking doors in the space
- All particles are initialized to random locations
- Robot has not yet received a sensor reading so all location are equally probable

Monte Carlo Localization - Step #2



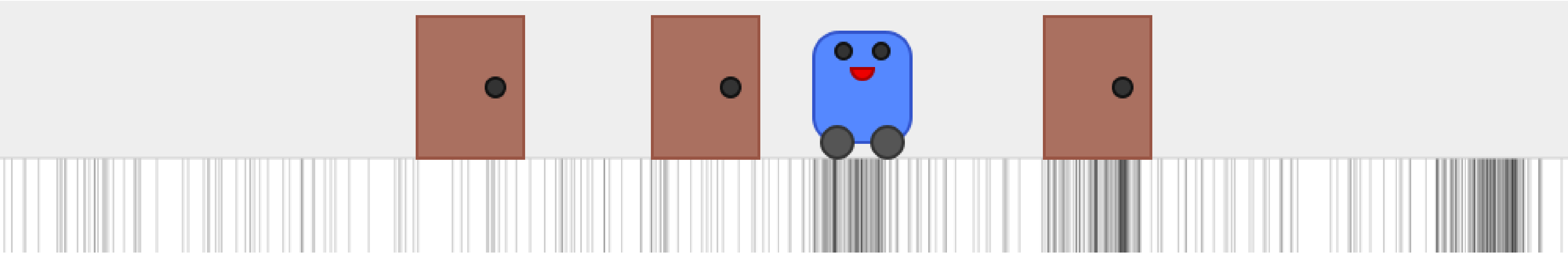
- Robot has received a sensor reading
- The red bars indicate the probability that the sensor would give that reading if the robot was at that particle
  - the red bar heights are used as weights for the particle
- The particles that are near the door have a higher probability

Monte Carlo Localization - Step #3



- The particles are resampled according to weights
- Particles with higher weights are selected more often and duplicated
- Particles with lower weights are less likely to be selected
- The new set of particles cluster around the more likely locations

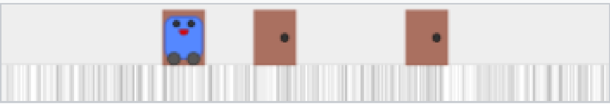
Monte Carlo Localization - Step #4



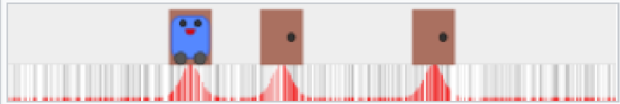
- Robot moves a certain amount
- The particles are moved the same amount with some motion noise introduced
- Repeat step #2 to update particle weights

Monte Carlo Localization

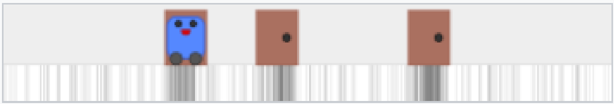
$t = 0$



The algorithm initializes with a uniform distribution of particles. The robot considers itself equally likely to be at any point in space along the corridor, even though it is physically at the first door.

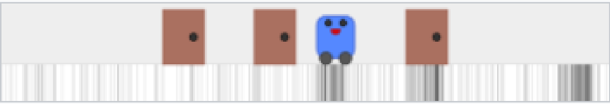


Sensor update: the robot detects **a door**. It assigns a weight to each of the particles. The particles which are likely to give this sensor reading receive a higher weight.

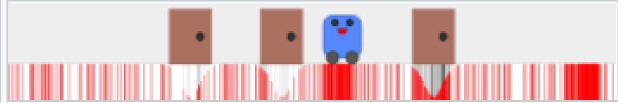


Resampling: the robot generates a set of new particles, with most of them generated around the previous particles with more weight. It now believes it is at one of the three doors.

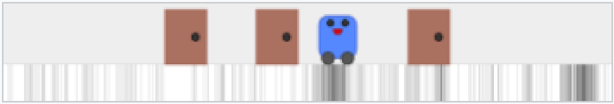
$t = 1$



Motion update: the robot moves some distance to the right. All particles also move right, and some noise is applied. The robot is physically between the second and third doors.

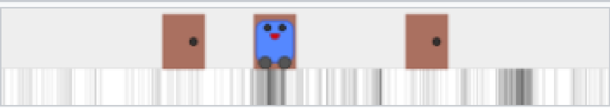


Sensor update: the robot detects **no door**. It assigns a weight to each of the particles. The particles likely to give this sensor reading receive a higher weight.

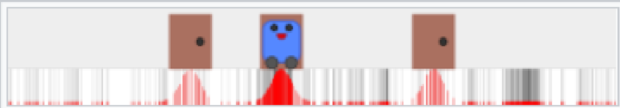


Resampling: the robot generates a set of new particles, with most of them generated around the previous particles with more weight. It now believes it is at one of two locations.

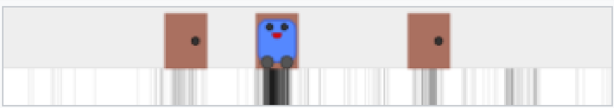
$t = 2$



Motion update: the robot moves some distance to the left. All particles also move left, and some noise is applied. The robot is physically at the second door.



Sensor update: the robot detects **a door**. It assigns a weight to each of the particles. The particles likely to give this sensor reading receive a higher weight.



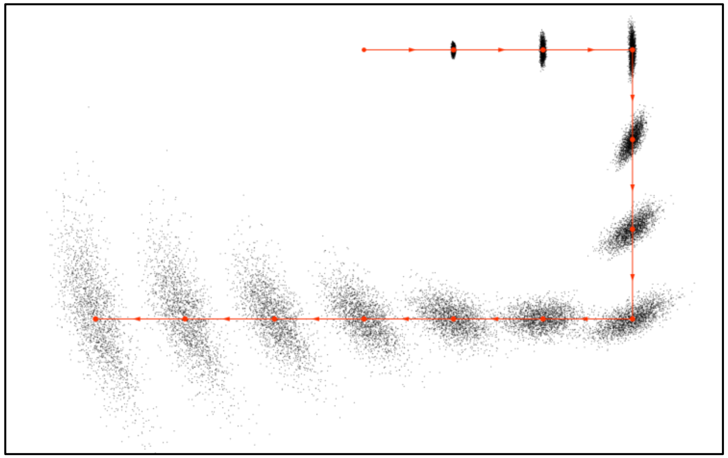
Resampling: the robot generates a set of new particles, with most of them generated around the previous particles with more weight. The robot has successfully localized itself.

Motion Model

- As the robot moves, there is noise in modelling its position
  - wheel slippage
  - uneven flooring
- This is why dead reckoning is not completely reliable
- It is common to model the motion noise as Gaussian

Motion Update

- The motion update step moves the position of all the particles according to the applied motion
  - The applied motion is the commands sent to the robot motors
- As the robot moves, there is noise in its position
  - wheel slippage
  - uneven flooring
- It is because of this noise that we can only estimate the position of the robot in the first place
- It is common to model the motion noise as Gaussian
  - The robot may command the motors to move 1m, but the robot may move .9m or 1.1m



Belief after moving several steps for a 2D robot using a typical motion model without sensing.

Sensor Update

- The sensor model gives the likelihood that the sensor will give a particular reading given the robot is in a particular location
  - p (z | x) = probability of getting a reading z given the robot is a location x

- For example, if the particle position is facing a wall, then the likelihood of getting a reading “free space in front of robot” is very small
  - the likelihood is still non-zero because the model incorporates the fact the sensor is noisy

- This model is built by characterizing the sensor in actual measurement scenarios

Particle Deprivation

- The particles that exist in low probability states will have weights so small that they will eventually be never selected
- This can lead to regions of the state space that do not have any particles
  - This can lead to incorrect localization if the robot is moved by an externally entity
  - This loss of particles is the particle deprivation problem
- During each iteration of the algorithm, randomly particles can be introduced throughout the state space
  - These randomly placed particles provide coverage should the robot localize incorrect

Determining the Position Estimate

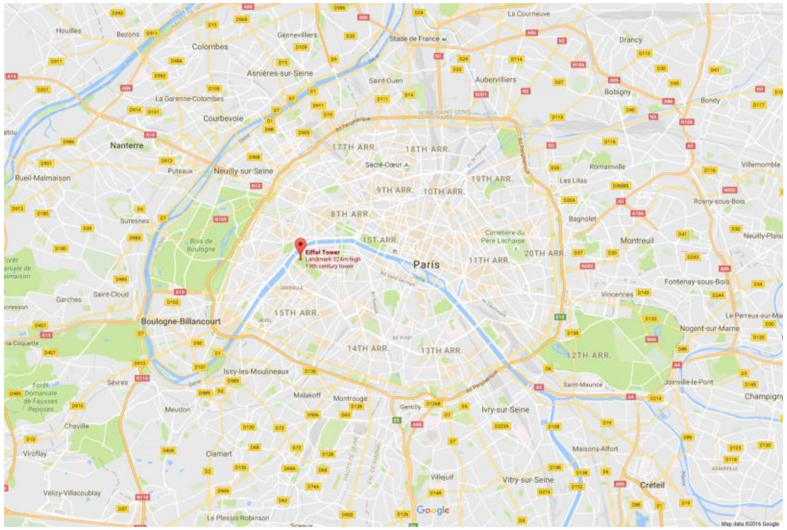
- After each iteration, the particles will begin to converge on the most likely position for the robot

- The position of the robot can be estimated using:
  - the position of the highest weight particle
  - the mean of the positions of all the particles
  - the mean of a small number of particles around the highest weight particle (robust mean)

Mapping

Mapping in Robotics

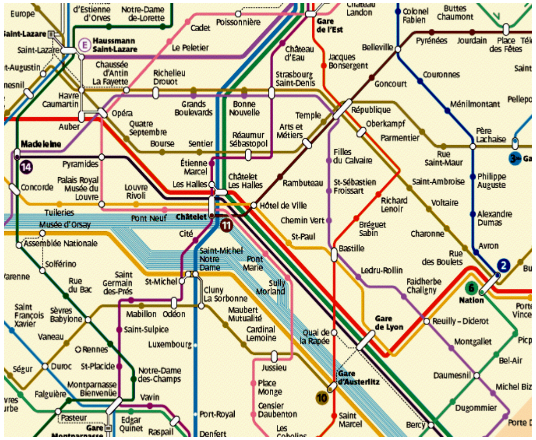
- Maps are essential for robots to devise plans for navigating
- Robots use maps to avoid obstacles and find shortest paths
- Often robots update their own maps as they explore and learn

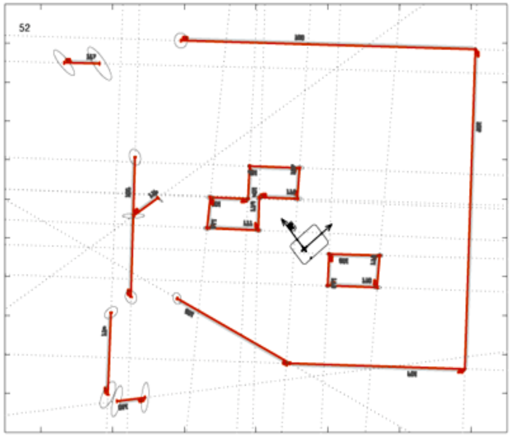


Map Representations

Examples:

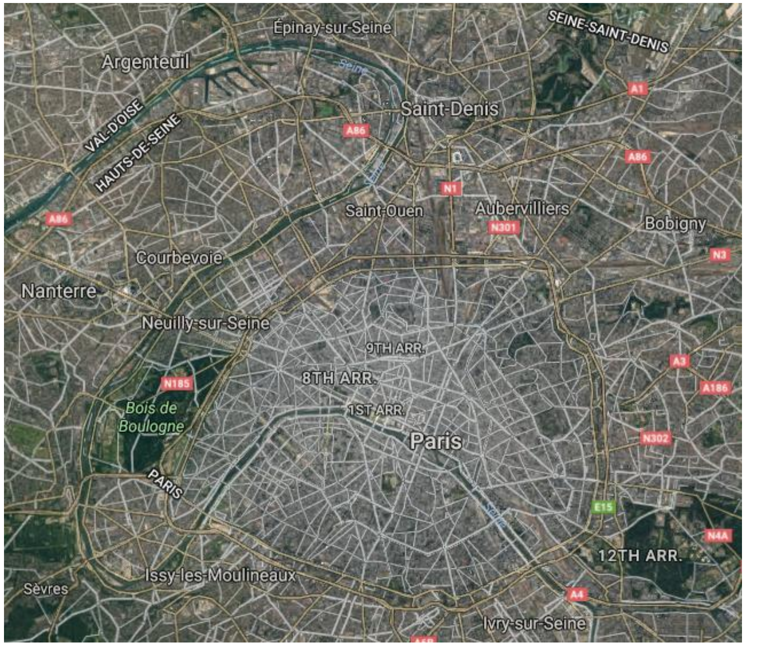
City map, subway map, landmark-based map





Aerial Maps

- A top-down view of an environment provides useful information that may be impossible to gather by a ground robot
- Sources of aerial maps
  - satellites
  - drones
  - planes

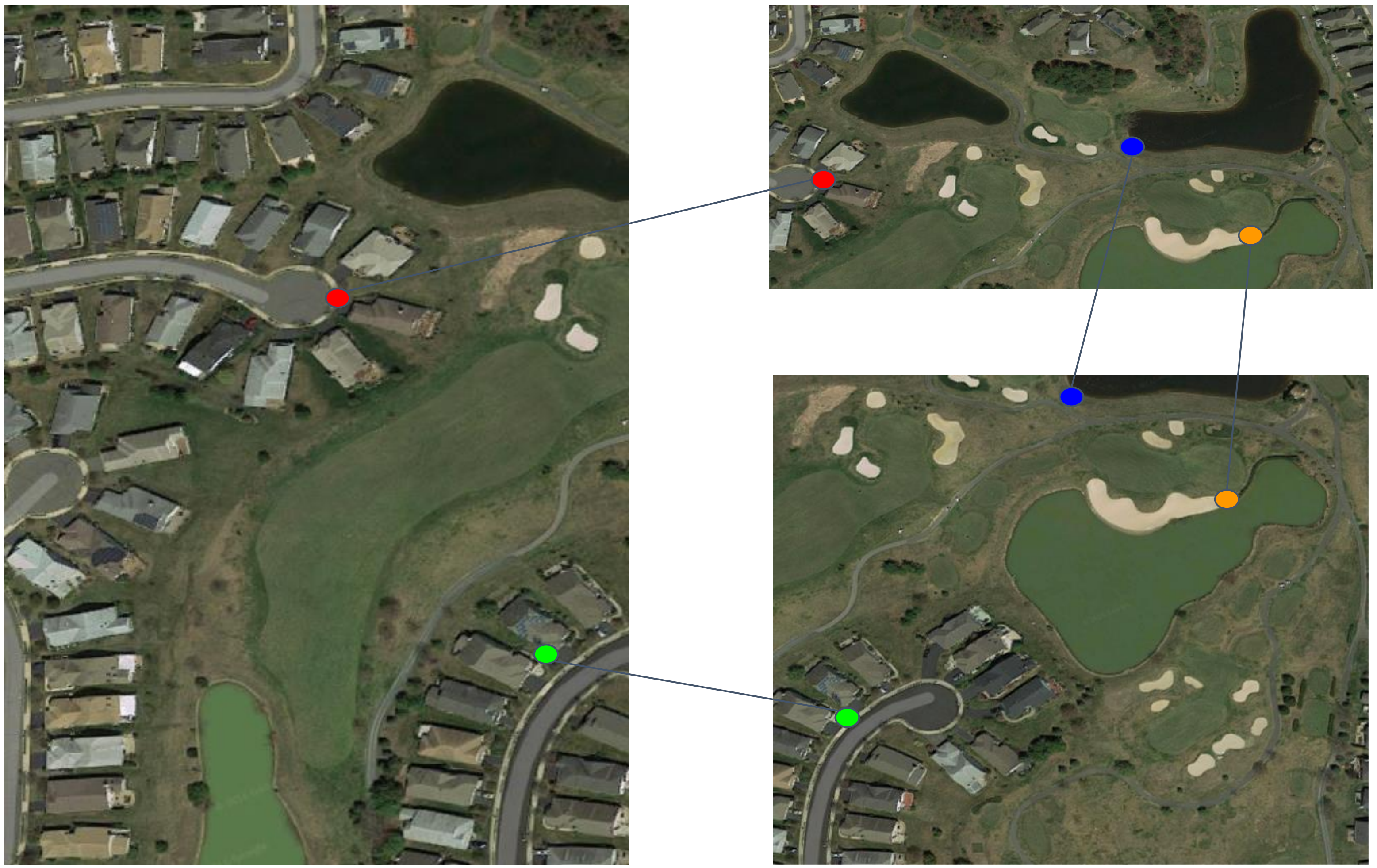


Map Stitching

- Discrete images must be combined to create a map
- Image stitching is the process of joining overlapping images
- Image Stitching pipeline:
  - define relationship between coordinate systems in images
  - determine alignment (and orientation) of images with each other
  - detect features (keypoints) in images
  - match features in each image with other images to show how images overlap
  - output the stitched image by projecting it onto a surface (plane, cylinder, sphere, etc)

Map Stitching: Example

Map Stitching: Example



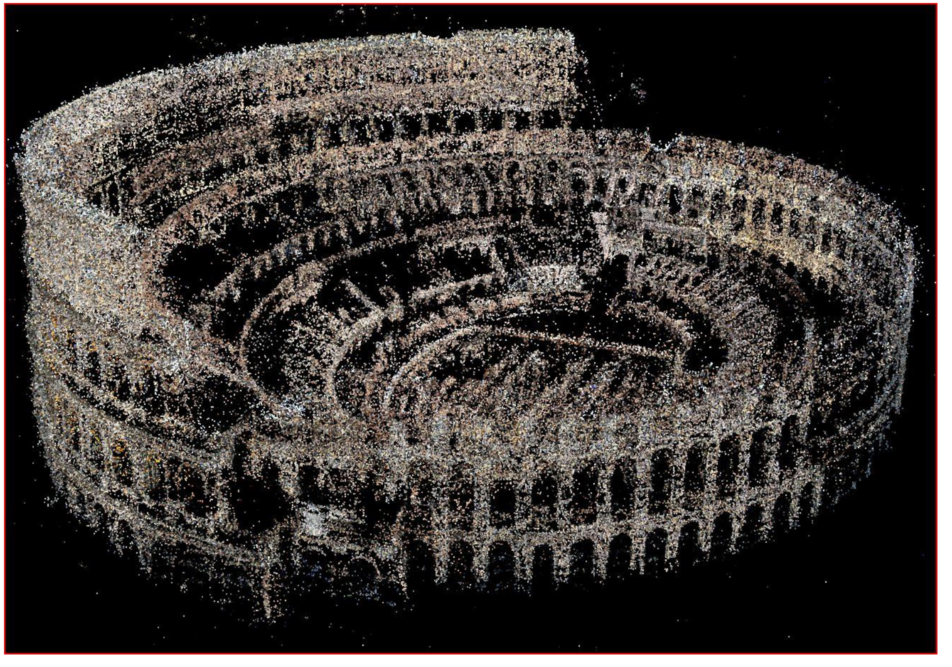
Map Stitching: Example

Multi-Layer Maps

- Maps can contain additional information:
  - semantic information (place names and descriptions)
  - timely information like traffic conditions or road closures
  - hyperspectral images that capture non-visible light
  - thermal images for detecting animals
- GIS (Geographical Information Systems) are used to analyze maps with many layers of data

Point Clouds

- Point Clouds or a set of coordinate with x,y,z values
- Sometimes points will also have color data associated with them.



http://grail.cs.washington.edu/rome/dense.html

Generating Point Clouds

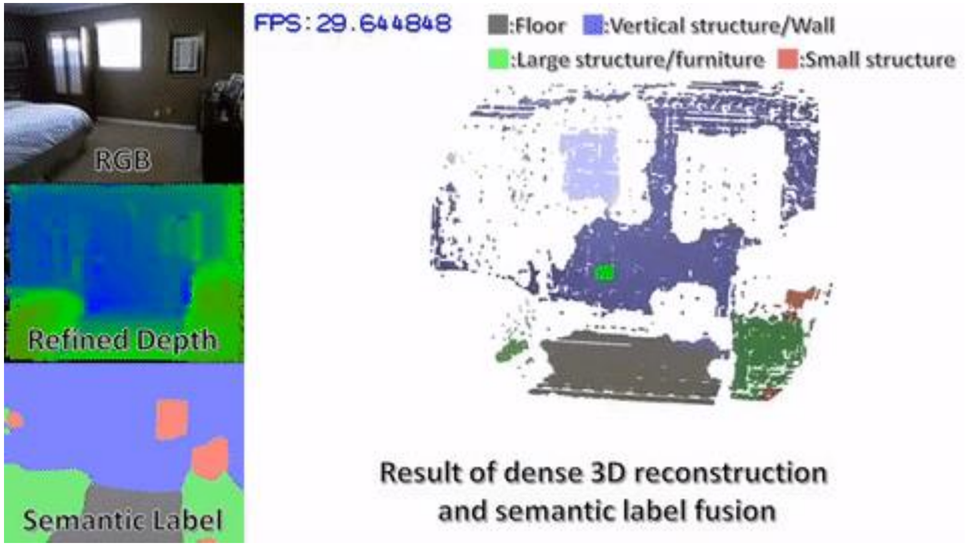
- Stereo Cameras
- LiDAR
- Radar
- Structure From Motion
- Image Alignment and Stitching

SLAM

Simultaneous Localization and Mapping

Simultaneous Localization and Mapping

- SLAM is used by robots that must construct a map of their environment while localizing themselves
- SLAM is used by self-driving cars, agricultural robots, and underwater robots
- GPS can be used to aide SLAM, but GPS alone is insufficient for SLAM.

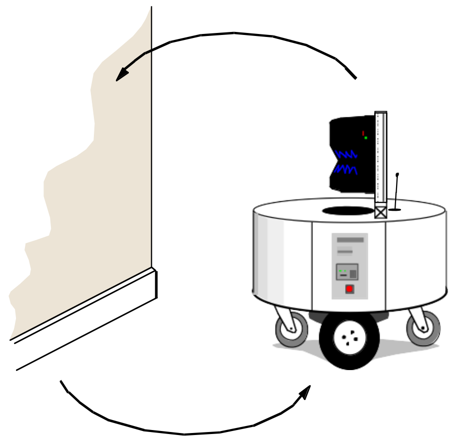


The SLAM Problem

SLAM is the process by which a robot **builds a map** of the environment and, at the same time, uses this map to **compute its location**

- **Localization:** inferring location given a map
- **Mapping:** inferring a map given a location
- **SLAM:** learning a map and locating the robot simultaneously

The SLAM Problem



- SLAM is a **chicken-or-egg problem**:
  - → A map is needed for localizing a robot
  - → A pose estimate is needed to build a map
- Thus, SLAM is (regarded as) a **hard problem** in robotics

The SLAM Problem

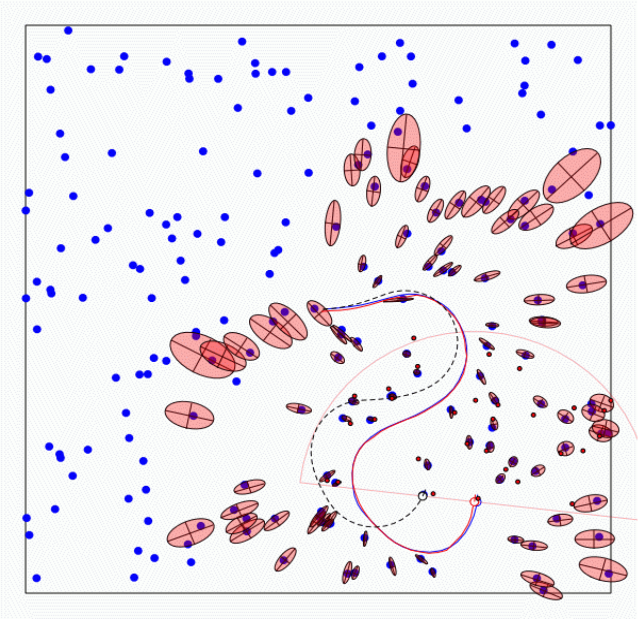
- SLAM is considered **one of the most fundamental problems** for robots to become truly autonomous

- A variety of different approaches to address the SLAM problem have been presented

The SLAM Problem

Given:

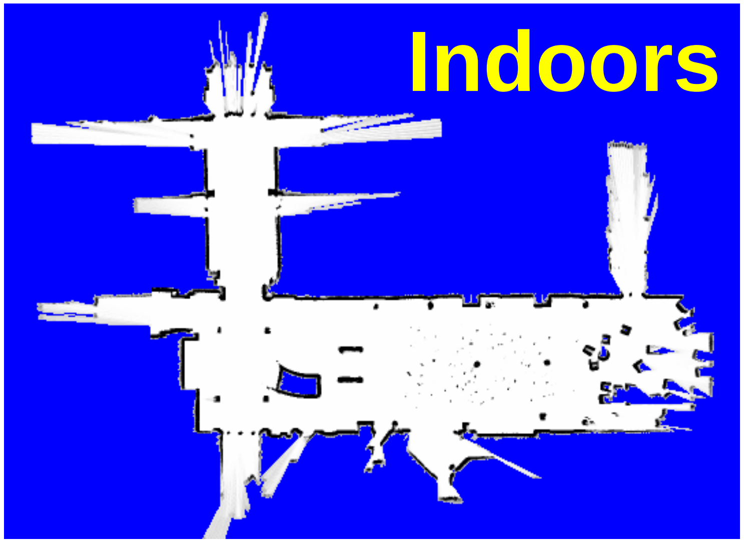
- The robot's controls
  $$\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k\}$$
- Relative observations

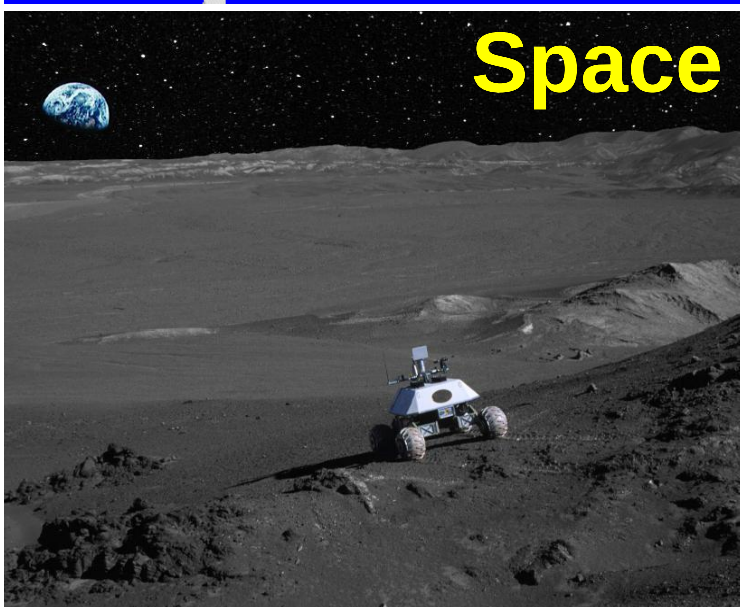


Wanted:

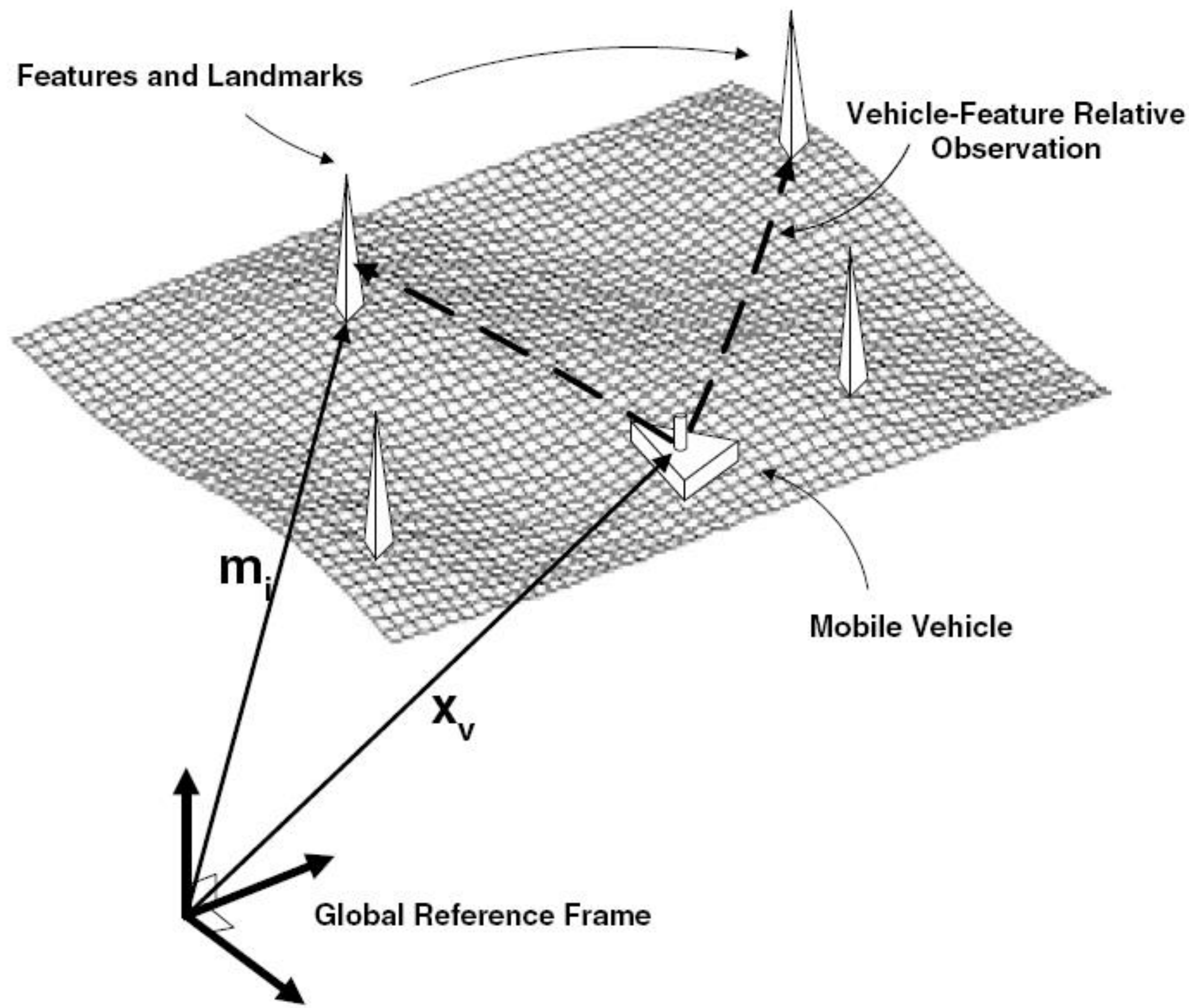
- Map of features
  $$\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n\}$$
- Path of the robot
  $$\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k\}$$

SLAM Applications



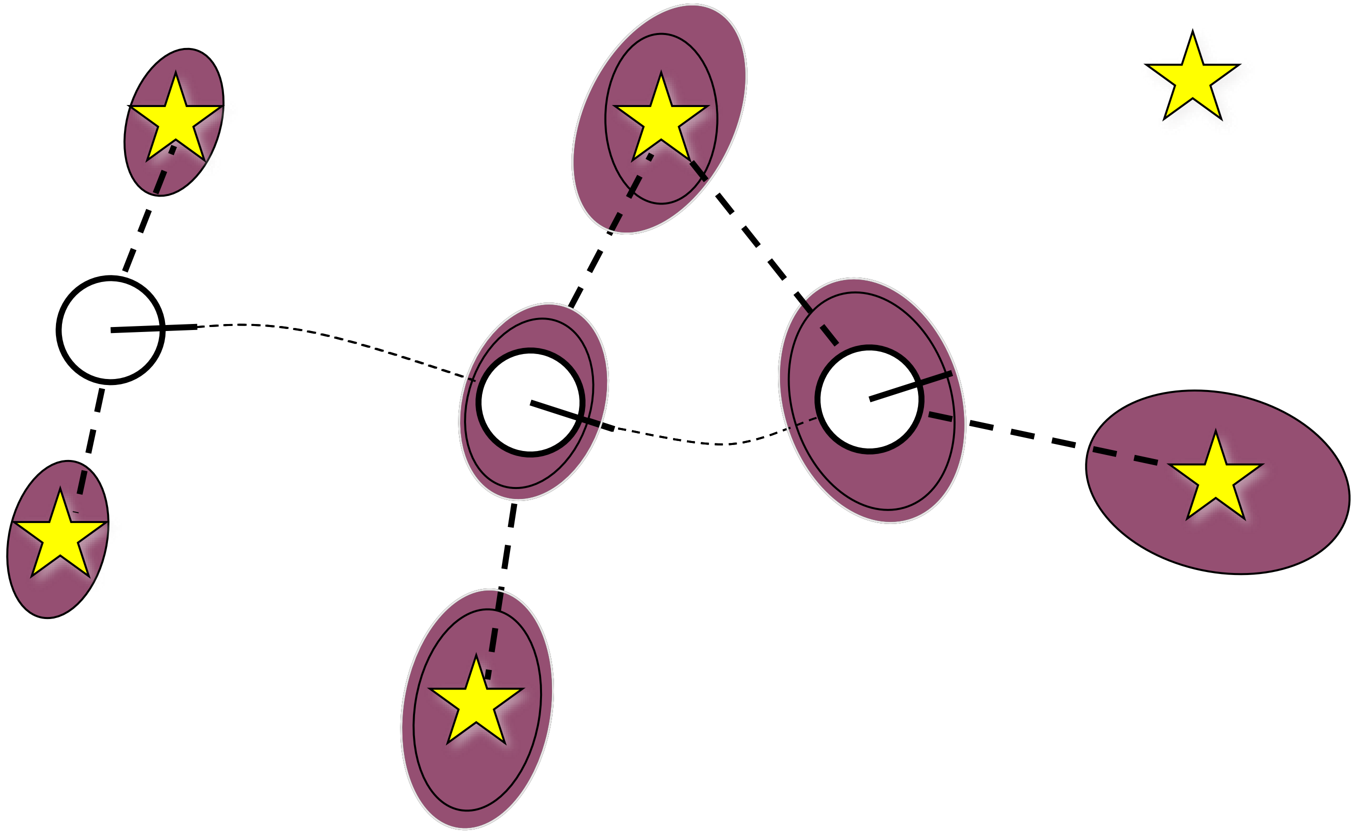


Structure of the Landmark-based SLAM-Problem



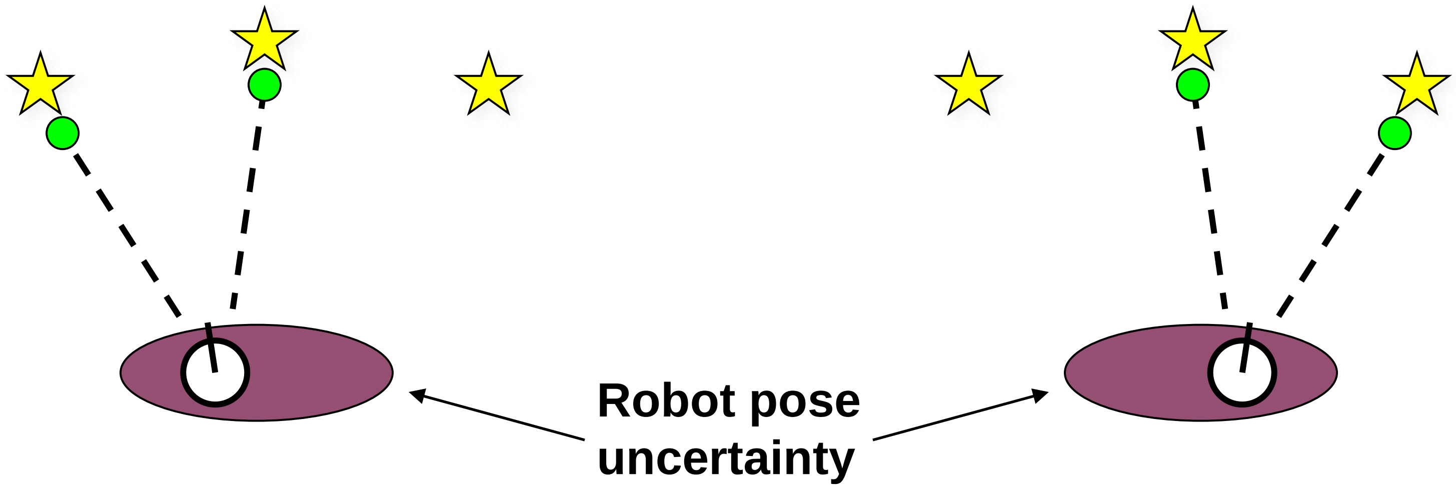
Why is SLAM a hard problem?

SLAM: robot path and map are both **unknown**



Robot path error correlates errors in the map

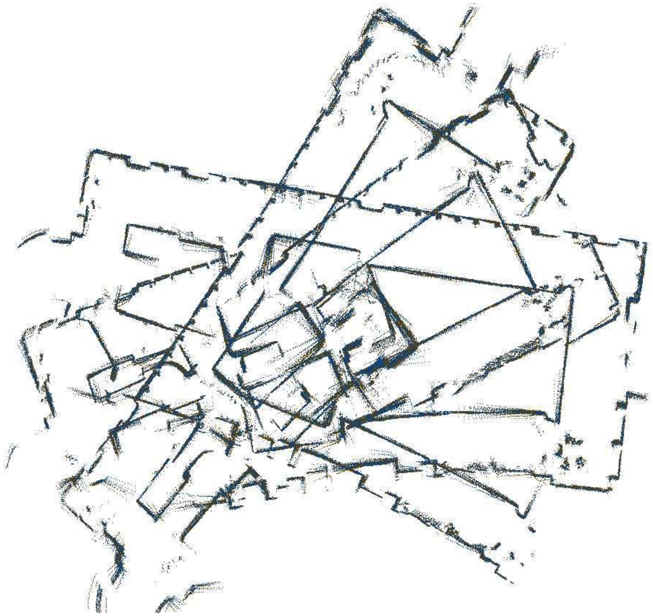
Why is SLAM a hard problem?

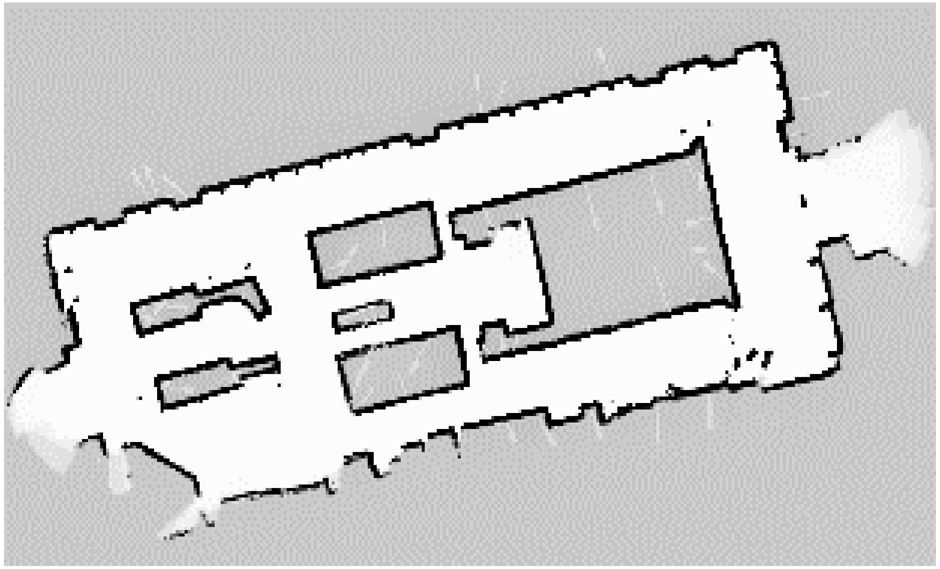


- In the real world, the mapping between observations and landmarks is unknown
- Picking wrong data associations can have catastrophic consequences
- Pose error correlates data associations

Cyclic Environments

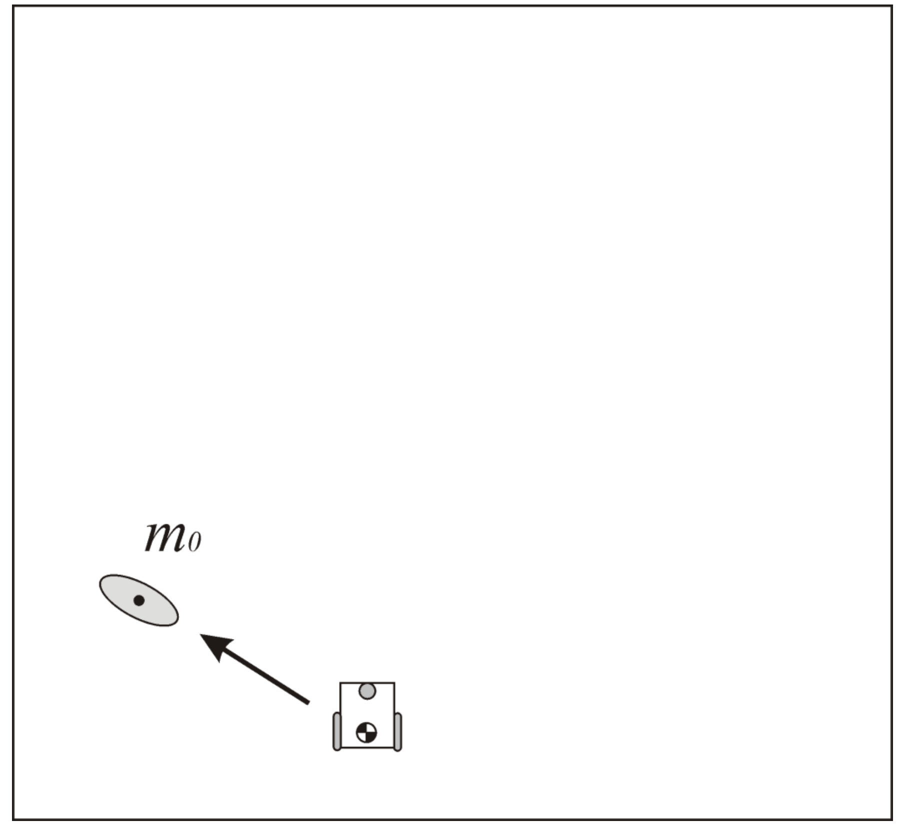
- Small local error accumulate to arbitrary large global errors!
- This is usually irrelevant for navigation
- However, when closing loops, global error does matter





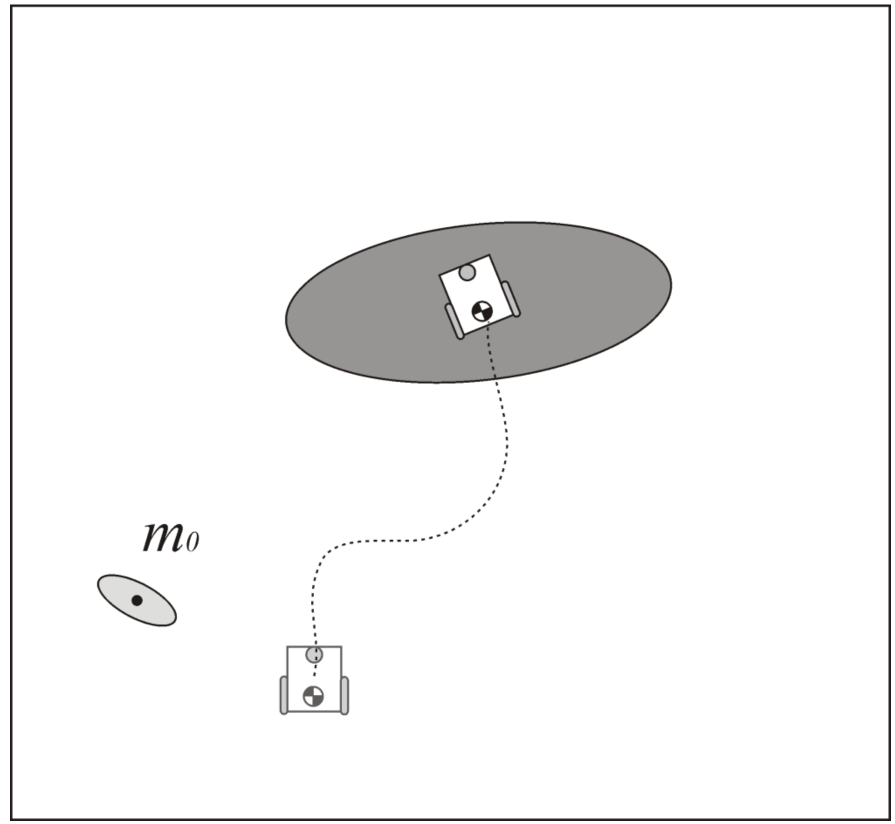
SLAM overview

- Let us assume that the robot uncertainty at its initial location is zero.
- From this position, the robot observes a feature which is mapped with an uncertainty related to the exteroceptive sensor error model



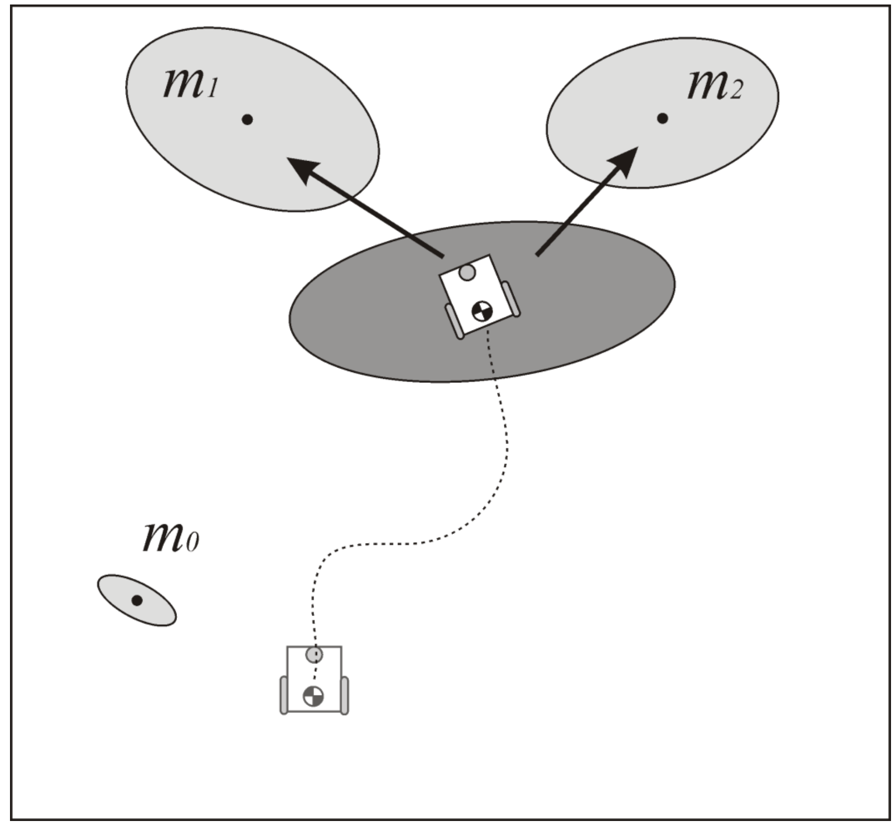
SLAM overview

- As the robot moves, its pose uncertainty increases under the effect of the errors introduced by the odometry



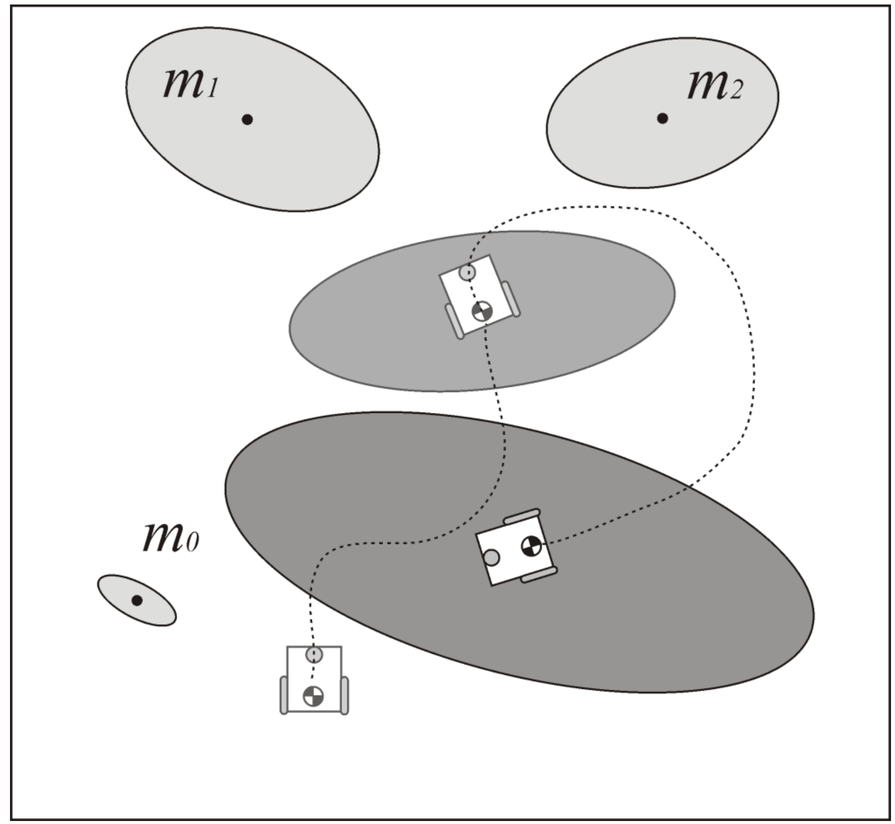
SLAM overview

- At this point, the robot observes two features and maps them with an uncertainty which results from the combination of the measurement error with the robot pose uncertainty
- The map becomes correlated with the robot position estimate.
- If the robot updates its position based on an observation of an imprecisely known feature in the map, the resulting position estimate becomes correlated with the feature location estimate.



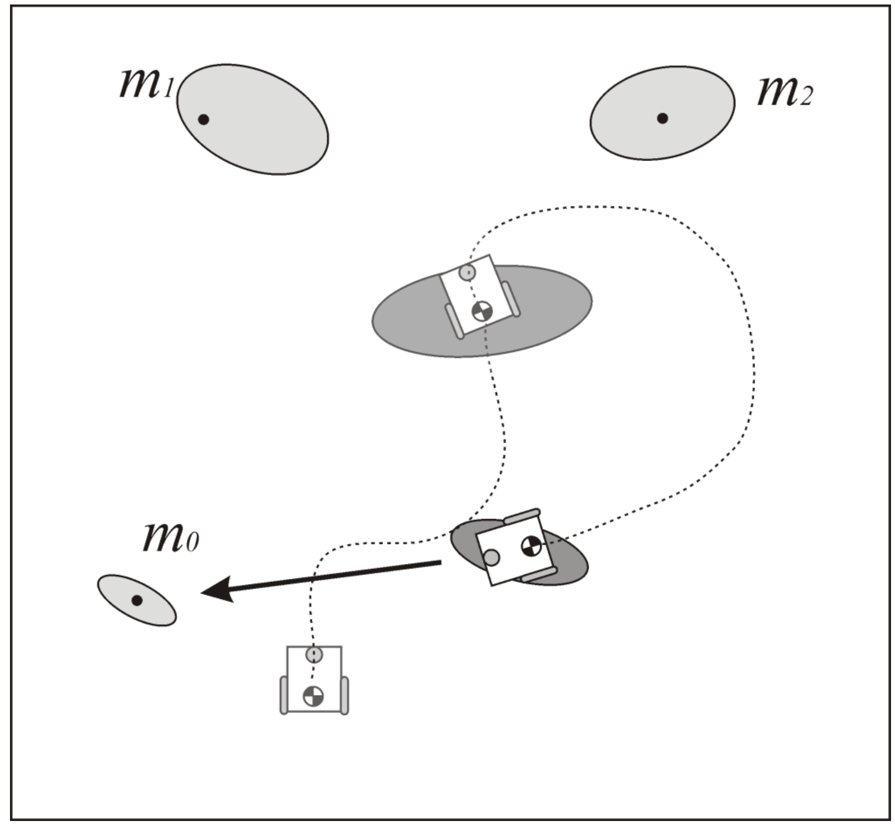
SLAM overview

- The robot moves again and its uncertainty increases under the effect of the errors introduced by the odometry



SLAM overview

- In order to reduce its uncertainty, the robot must observe features whose location is relatively well known. These features can for instance be landmarks that the robot has already observed before.
- In this case, the observation is called *loop closure detection.*
- When a loop closure is detected, the robot pose uncertainty shrinks.
- At the same time, the map is updated and the uncertainty of other observed features and all previous robot poses also reduce



SLAM:
Simultaneous Localization and Mapping

- Full SLAM: Estimates entire path and map!

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$

- Online SLAM:

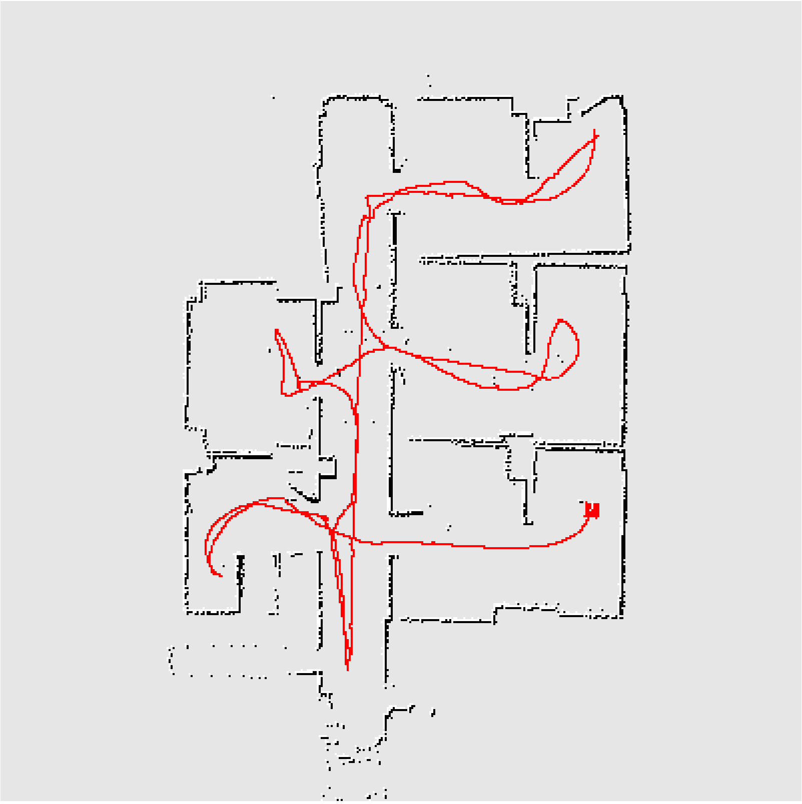
$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int\int\ldots\int p(x_{1:t}, m \mid z_{1:t}, u_{1:t})\, dx_1 dx_2 \ldots dx_{t-1}$$

Integrations typically done one at a time

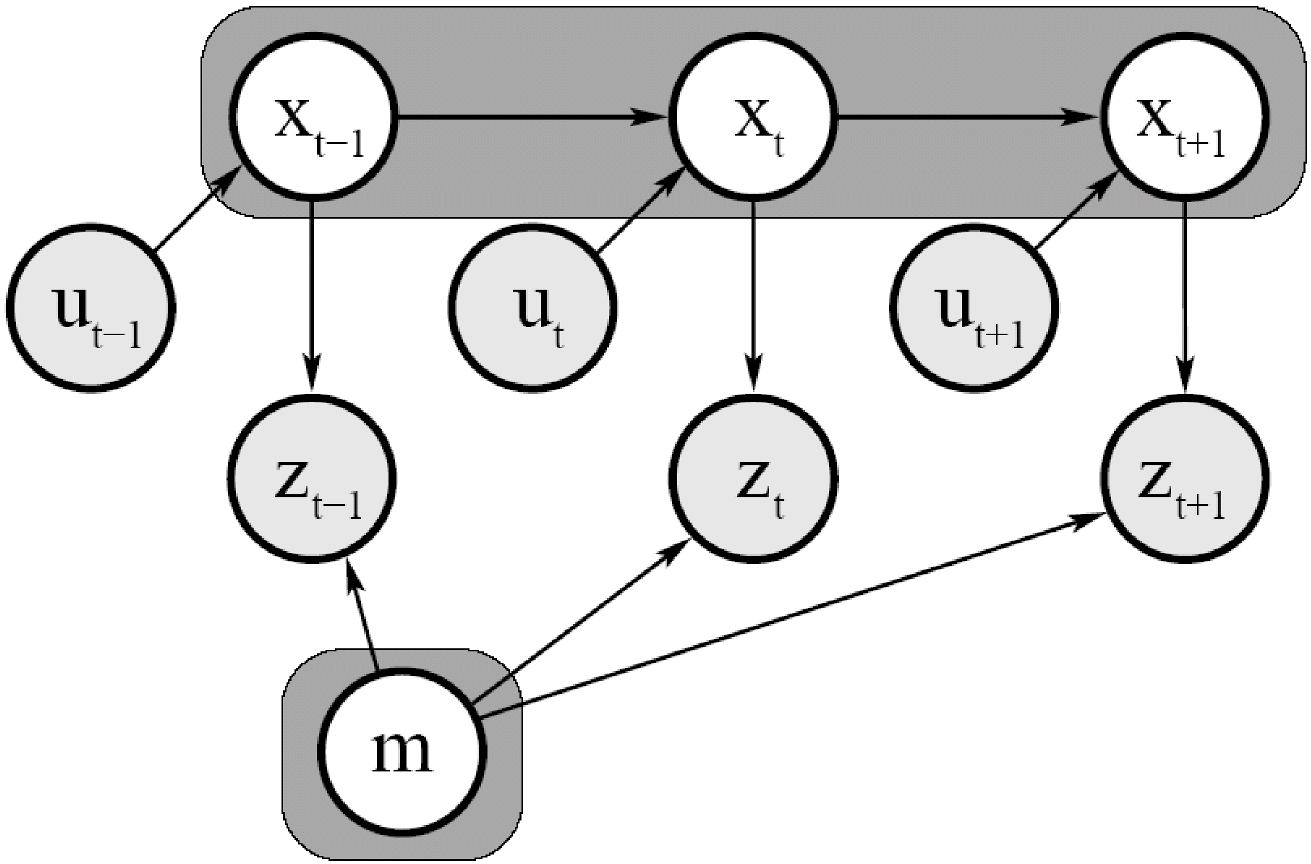
Estimates most recent pose and map!

Solution Likelihood

- SLAM operates by maximizing two likelihoods:
  - likelihood of the map given the pose and sensor readings of the robot
  - likelihood of the pose of the robot given the map and the sensor readings
- Simultaneously optimizing for both of these will let the robot produce a map while estimating its pose in the map

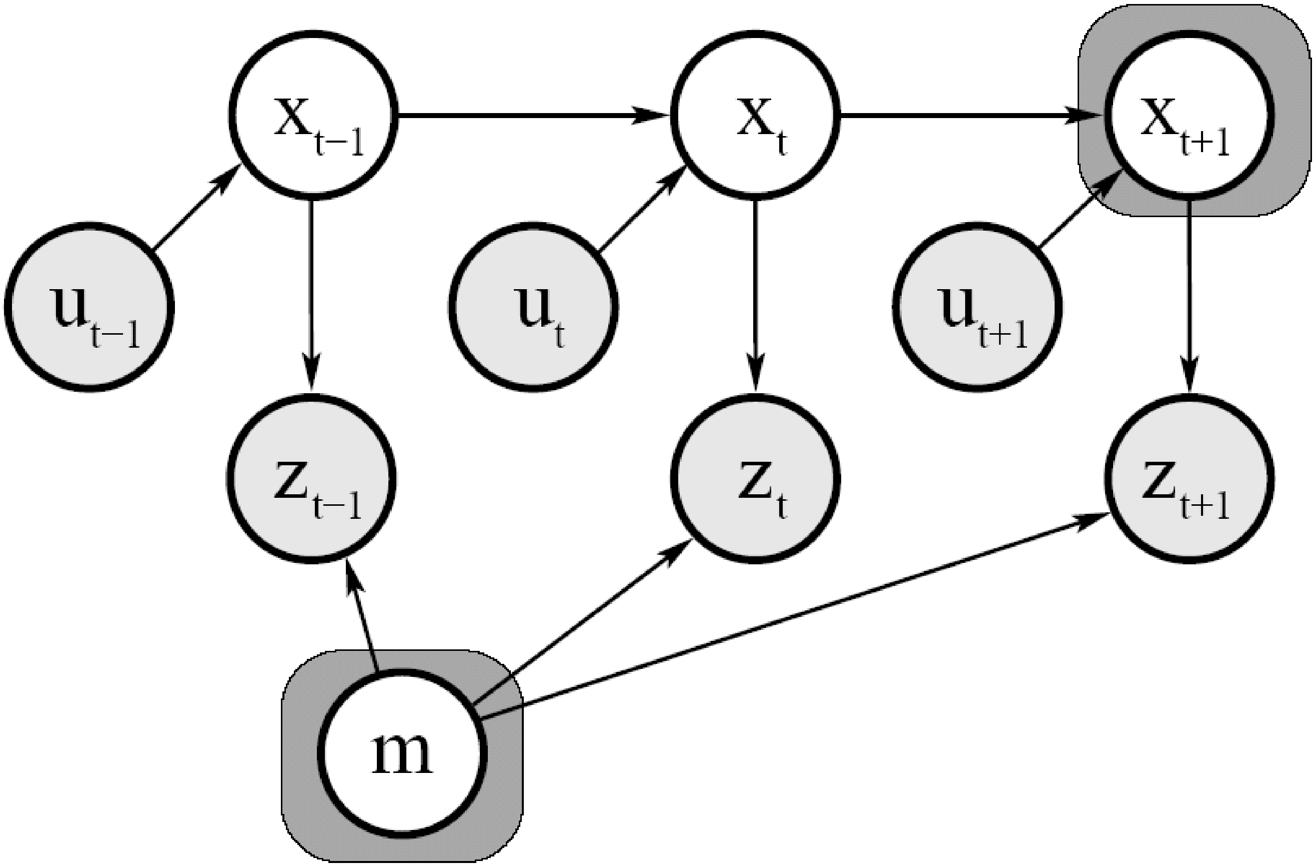


Graphical Model of Full SLAM:



$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$

Graphical Model of Online SLAM:



$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int \int \ldots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t})\, dx_1\, dx_2 \ldots dx_{t-1}$$

The Three SLAM paradigms

- Most of the SLAM algorithms are based on the following three different approaches:
  - Extended Kalman Filter SLAM: (called EKF SLAM)
  - Particle Filter SLAM: (called FAST SLAM)
  - Graph-Based SLAM

Extended Kalman Filter SLAM

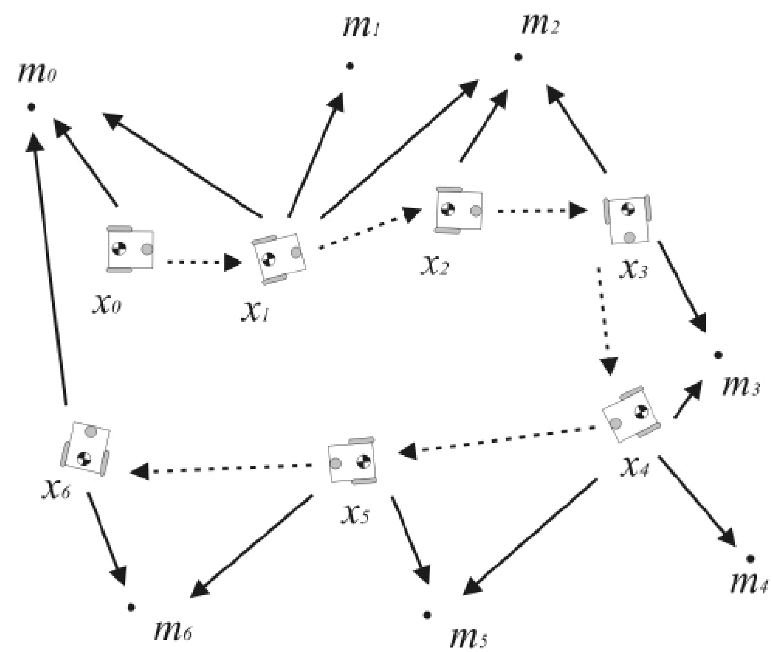
- Extended Kalman Filters can be used to estimate the pose of the robot and the map
- The state vector is the landmarks and the pose of the robot
- The map transforms are nonlinear so the basic Kalman Filter is insufficient
- Complexity is quadratic with number of landmarks

Particle Filter SLAM

- Particle Filters can also be used to estimate the map the pose of the robot
- Particle Filters have been shown to scale to handle larger numbers of landmarks
- FastSLAM is a popular SLAM approach that uses Particle Filters to handle more than 50,000 landmarks.
- Particle Filters can achieve a logarithmic complexity with the number of landmarks.

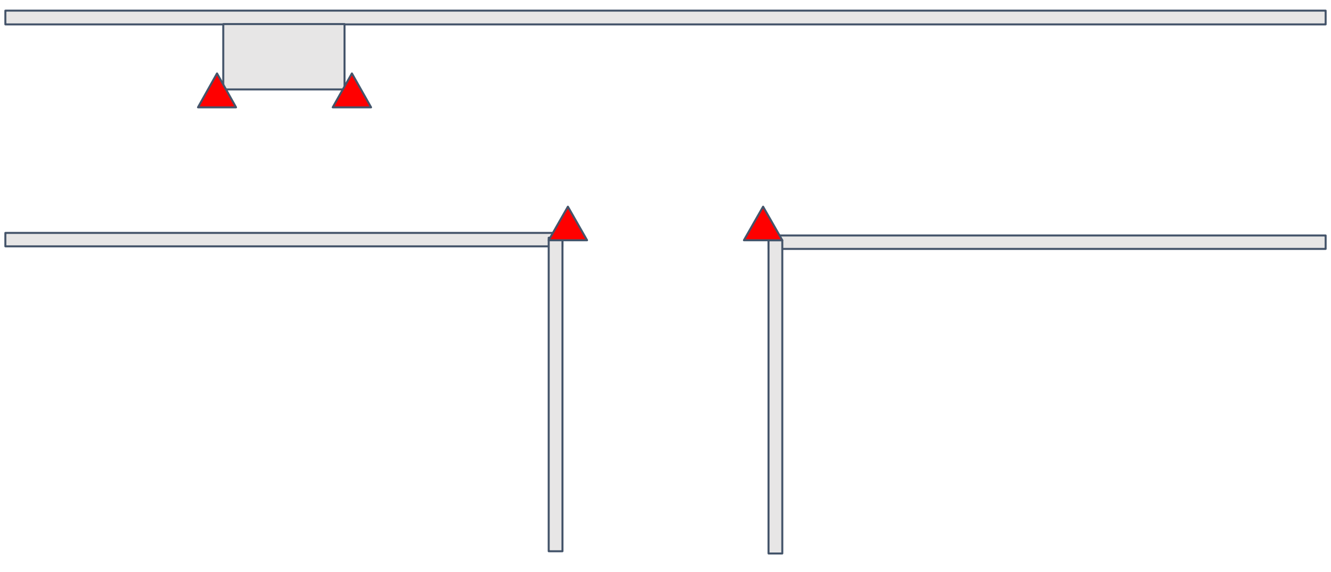
Graph-Based SLAM

- Graph-based SLAM problem can be interpreted as a sparse graph of nodes and constraints between nodes.
- The nodes of the graph are the robot locations and the features in the map.
- The constraints are the relative position between consecutive robot poses, (given by the odometry input ***u***) and the relative position between the robot locations and the features observed from those locations.



Essential: Identifying Landmarks

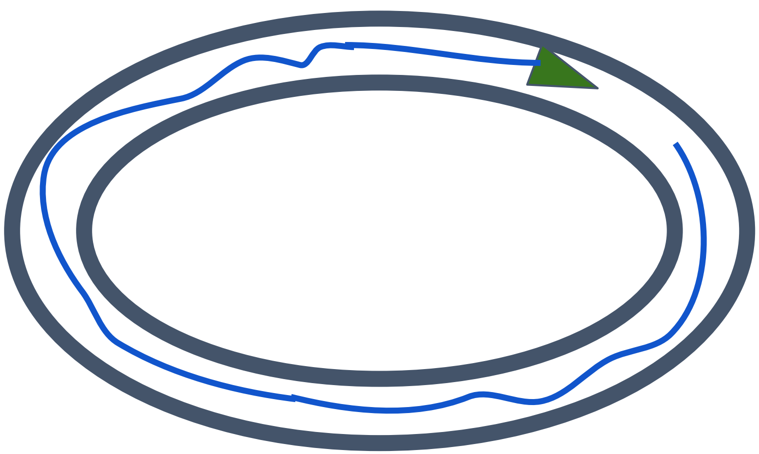
- Landmarks in the map are required for relocalizing
- Landmarks should be distinct from the environment and easy to recognize
- Landmarks should be stable
- Various to feature the environment



Essential: Closing Loops

- In circular environment, small map errors can be compounded to produce incoherent maps.

Real Environment



Map Environment