CSCI444/944
Perception & Planning

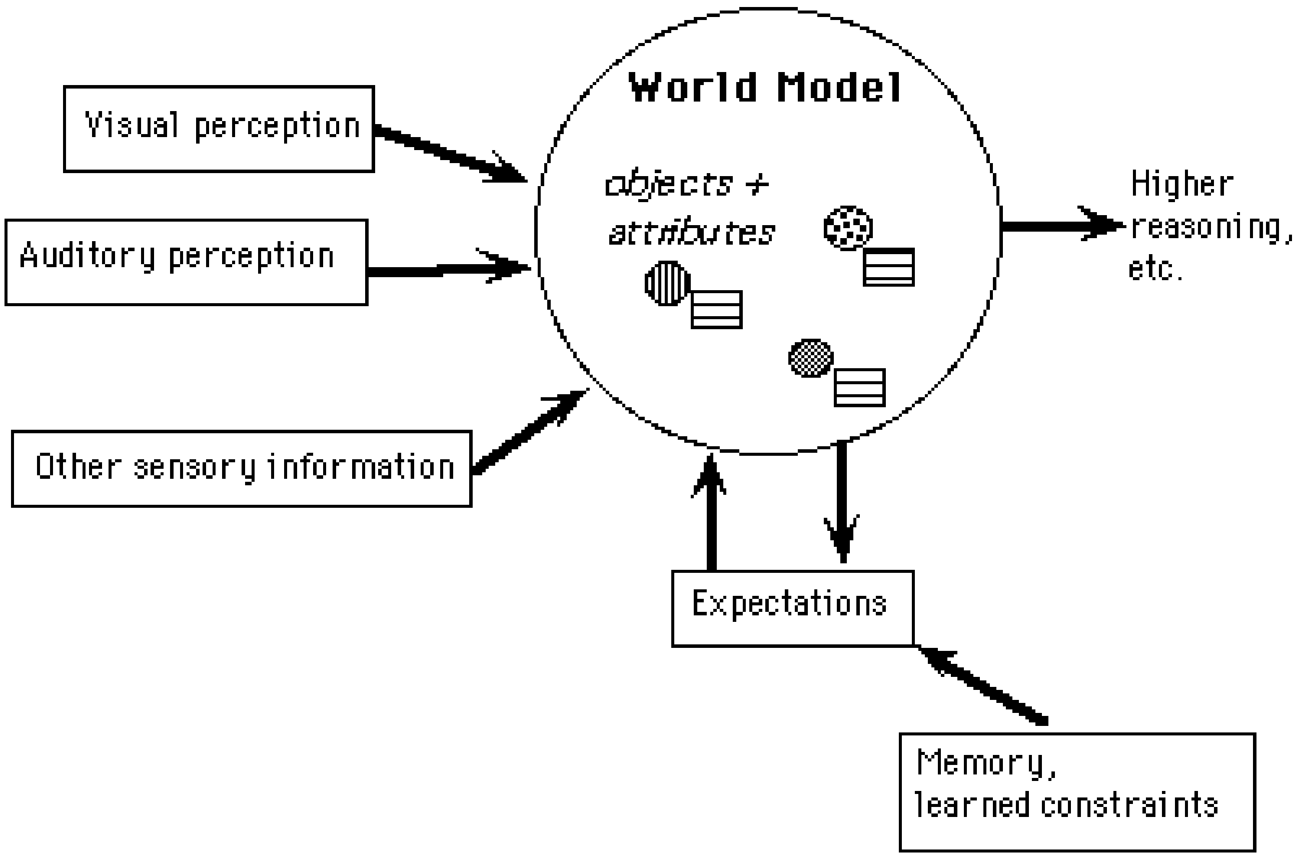
Week 4

Perception

based

Models

Internal World Model



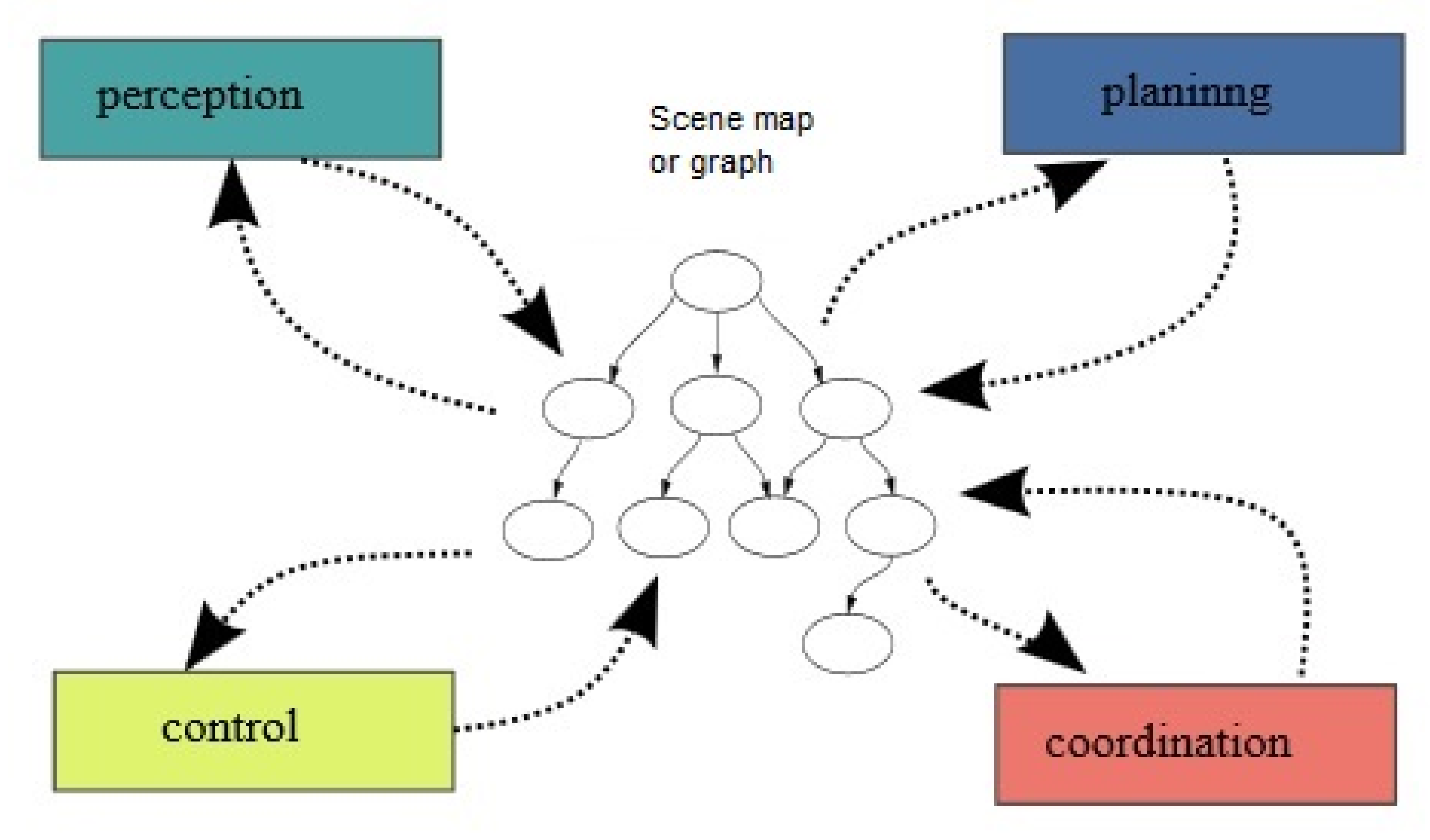
An internal model of the world can facilitate decision making

Internal World Model

- Robots need an internal representation of their surroundings to be able to realise what is in the immediate environment and how to interact with it.
- As real world scenes are dynamic, it is not sufficient to load a static environment model from a file.
- Sensor data needs to be processed and continuously incorporated into the world model of a robotic application.

Internal World Model

- Scene data can be stored in a variety of data structures and used for path planning, task coordination, motion control modules or perception functionality.

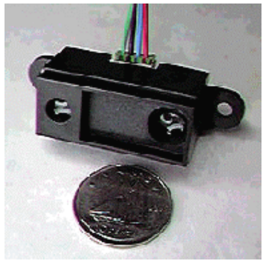


Internal World Model

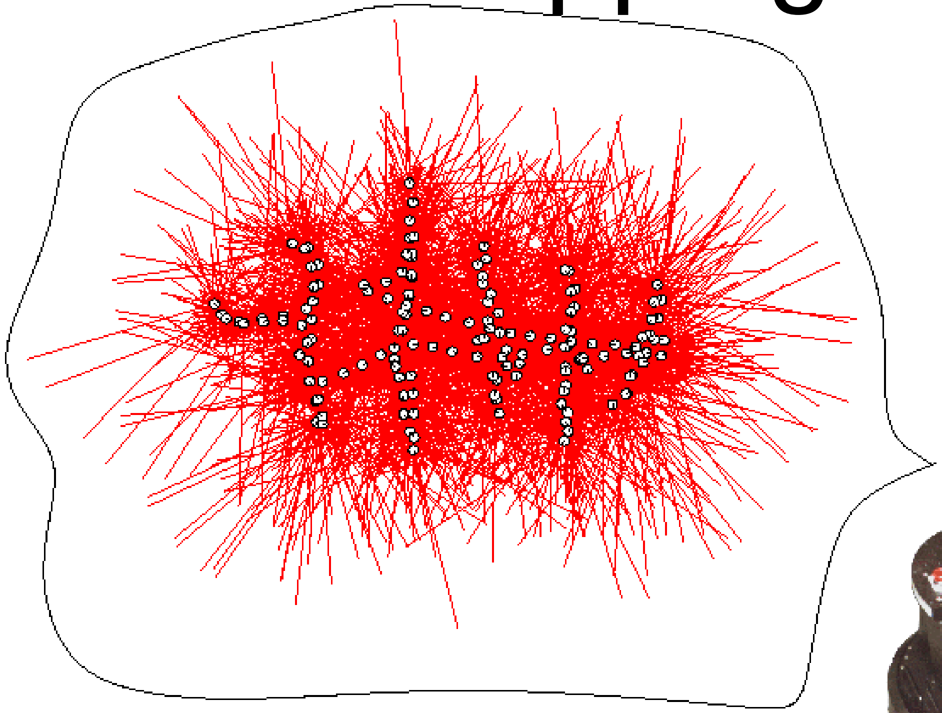
- The main purpose of a world model is to answer questions like:
  - Where am I going?
  - What’s the best way there?
  - Where have I been?
  - Where am I now?
  - *Can I get there?*
  - *. . . etc . . .*

Internal World Model

- For a mobile robot an internal world model can be achieved with a map using range sensors and odometry.



The General Problem of Mapping



What does the environment look like?

The General Problem of Mapping

- Formally, mapping involves, given a series of controls u_i and sensor observations z_i,

$$d = \{u_1, z_1, u_2, z_2, \dots, u_n, z_n\}$$

to calculate the most likely map

$$m^* = \arg\max_m P(m|d)$$

Mapping as a Chicken and Egg Problem

- Mapping involves to simultaneously estimate the pose* of the vehicle while updating the map.
- The general problem is denoted as the simultaneous localization and mapping problem (SLAM).
  - A robot moving through an unknown environment accumulates errors in odometry making it increasingly uncertain where it is.
  - Constructing an accurate map of the environment is difficult if the pose of the robot is uncertain.
- Throughout this section we will describe how to calculate a map given we know the pose of the vehicle.

*pose: (x, y, θ) location, orientation

Problems in Mapping

- Not all mapping problems are equally hard
- Difficulty depends on:
  1. **Size:** the larger the environment the more difficult to aquire a map.
  2. **Noise:** Robot sensors and actuators are never free of noise. The larger the noise the more difficult the problem.
  3. **Perceptual ambiguity:** If different places look alike the harder it is to establish correspondence to different points in a map.
  4. **Cycles:** If a robot can move in cycles the odometric errors can accumulate to huge values.

Problems in Mapping

- Sensor interpretation
  - How do we extract relevant information from raw sensor data?
  - How do we represent and integrate this information over time?

- Robot locations have to be estimated
  - How can we identify that we are at a previously visited place?
  - This problem is the so-called data association problem.

Problems in Mapping

Example:
(a) Raw odometry data obtained from a laser range finder.
(b) The result of a mapping algorithms

Problem formulation:
How to map raw data so as to obtain an accurate map of the environment from raw data?

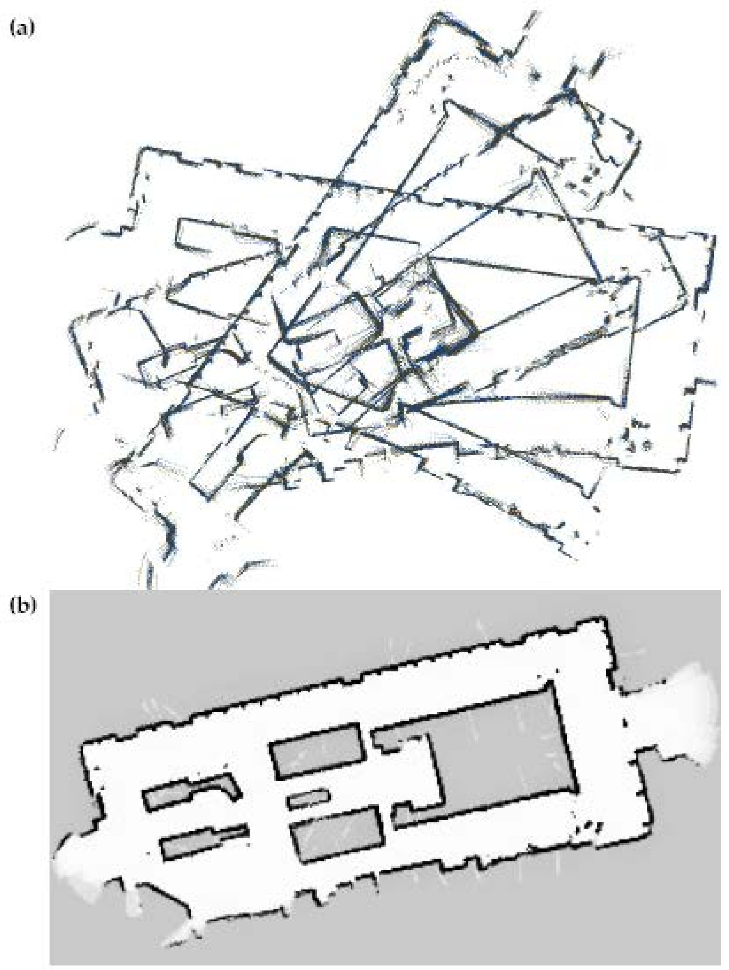
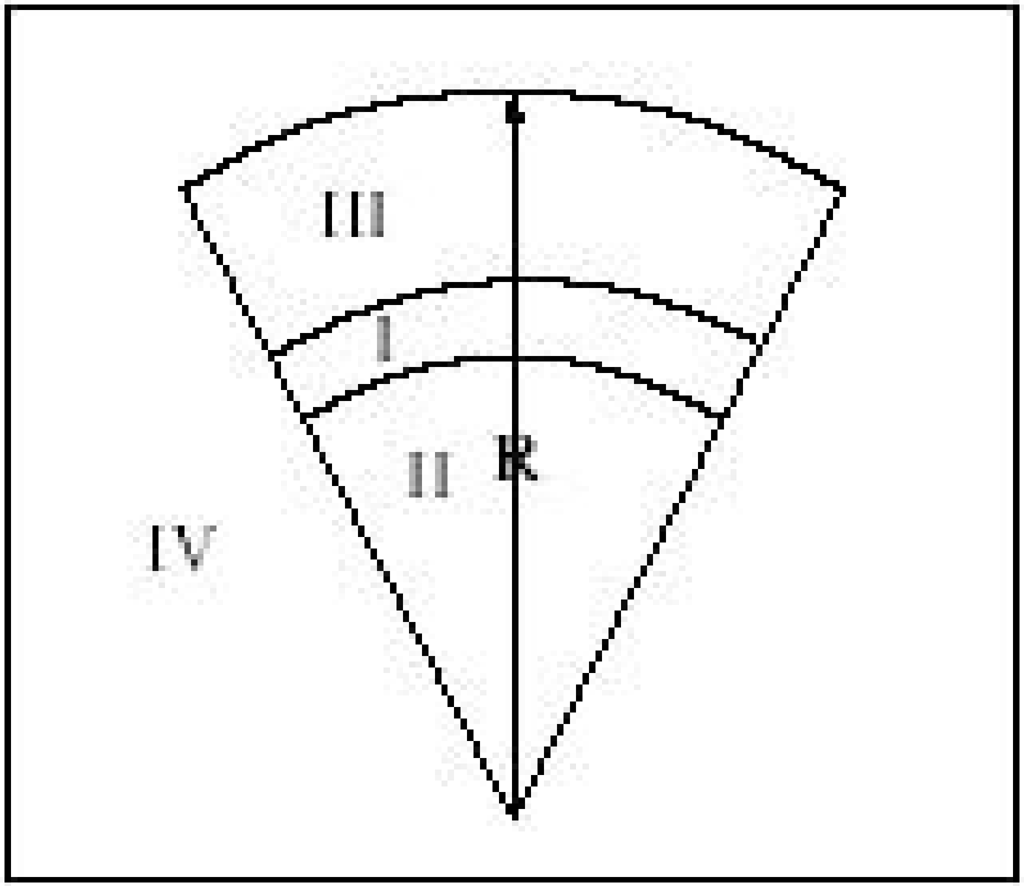
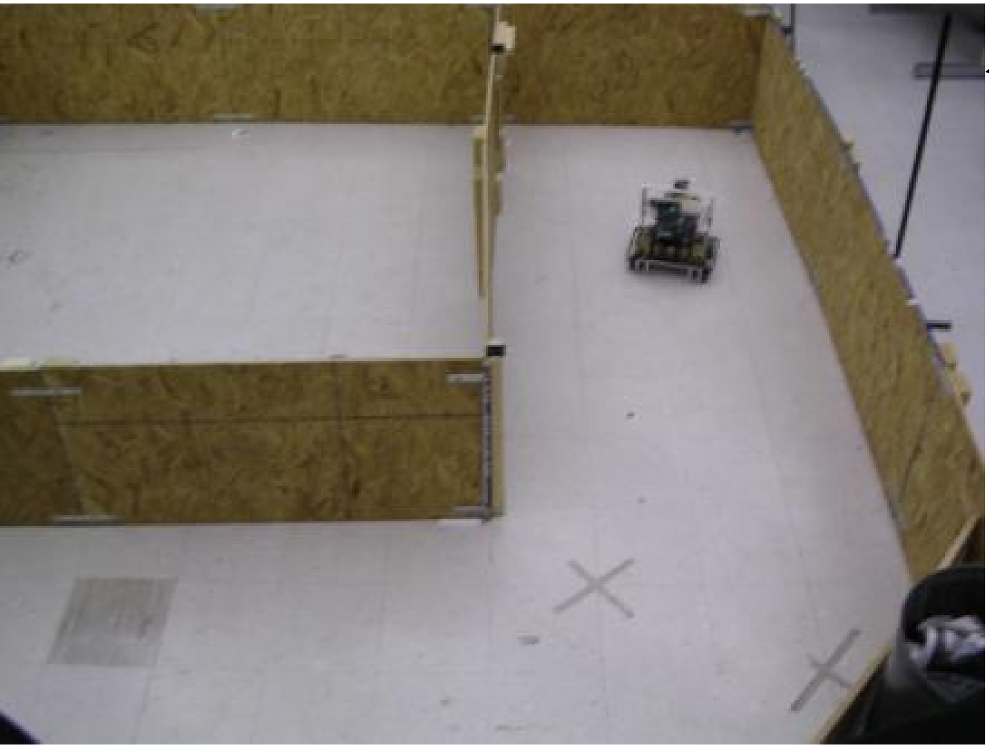


Figure 9.1 (a) Raw range data, position indexed by odometry. (b) Occupancy gric map.

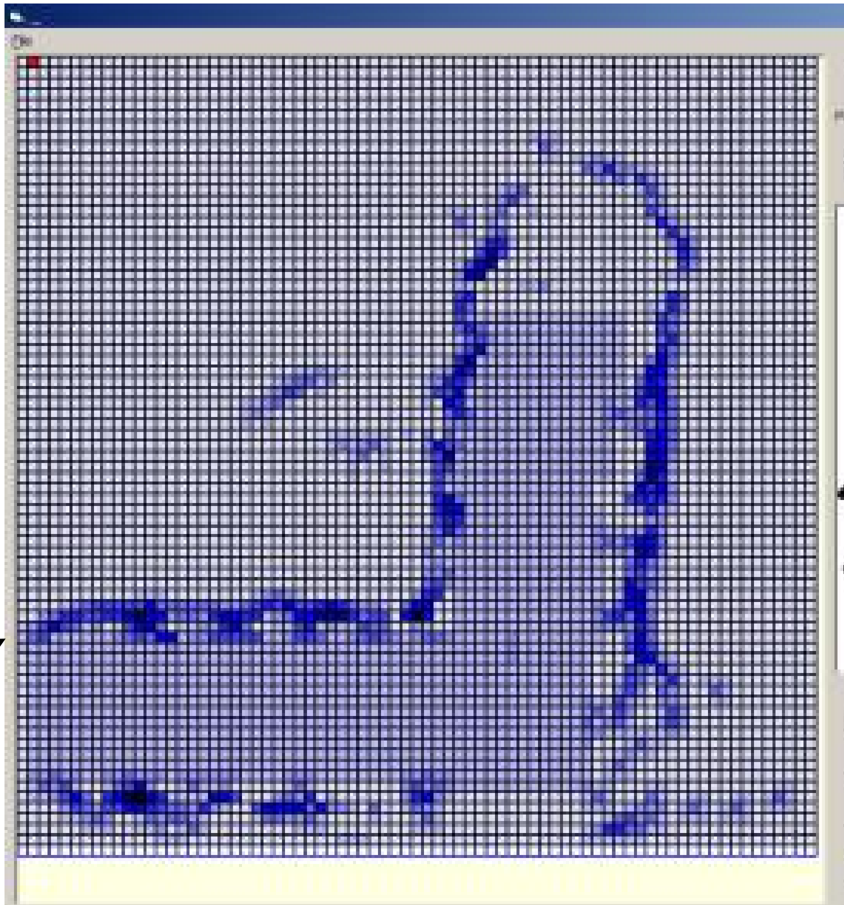
Sonar Sensor Model



Sonar Sensor Model



Actual environment

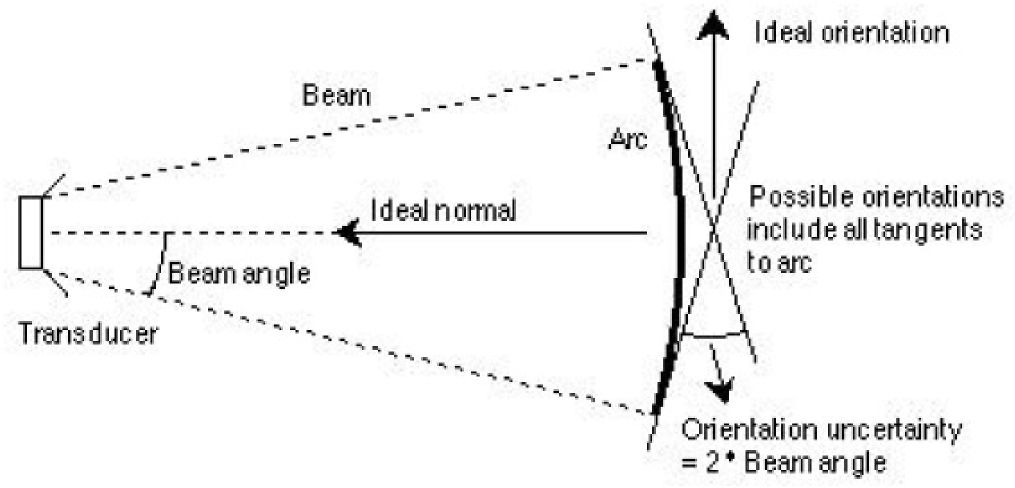


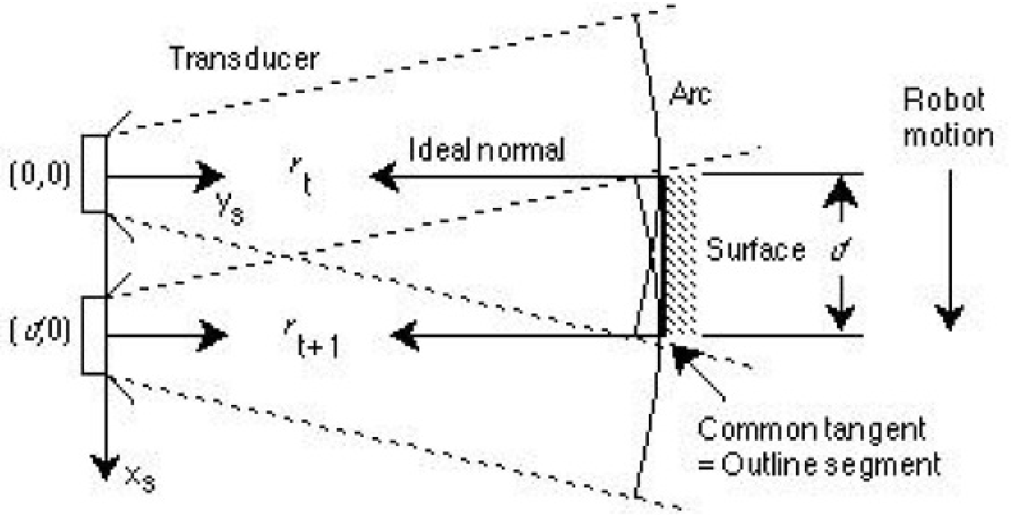
Raw sensory data affected by: Arcs, noise, drift, artefacts (reflection & edge effects). Notice the problems with corners.

Sonar Sensor Model

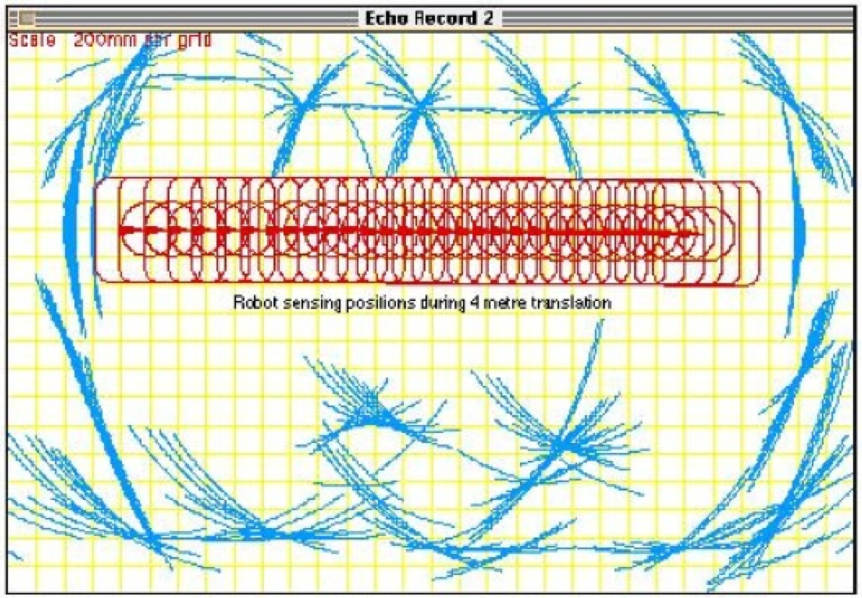
- Sonar models can help with mapping raw data to a map.
- The two most common sonar sensor models are:
  - Geometric representations.
    - e.g. arc models.
  - Occupancy grids.
    - Large grids – computationally expensive.

Arc Sensor Model

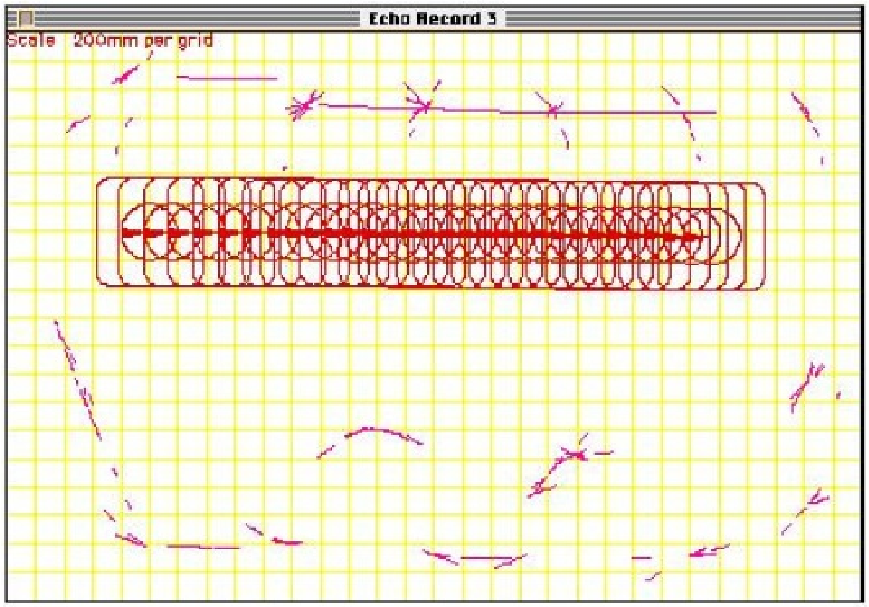

Ideal orientation
Beam
Arc
Ideal normal
Possible orientations include all tangents to arc
Beam angle
Transducer
Orientation uncertainty = 2 * Beam angle



Arc Sensor Model



Environment map showing estimated arc segments from sonars



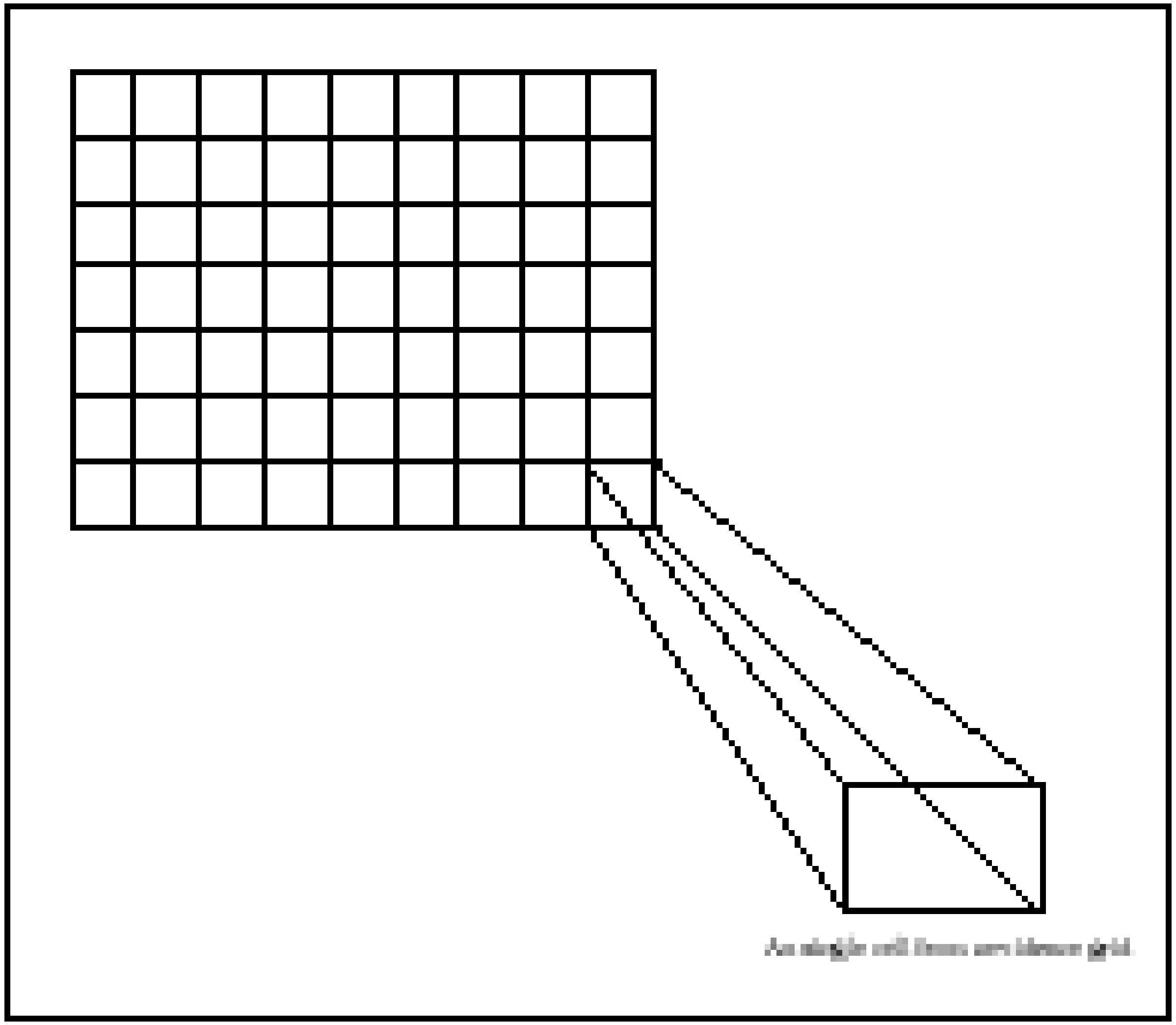
Processed arc segments showing object surfaces

Notice: size of an arc increases with distance due to the spread of signal waves

Occupancy Grids

- A tool to construct an internal model of *static* environments based on sensor data.
- The environment to be mapped is divided into regions.
- Each grid cell is an *element* and represents an area of the environment.
- Are 2D floor plan maps that describe a 2D slice of a 3D world.
- Used when a robot moves on a flat surface & sensors capture only a slice of the world.

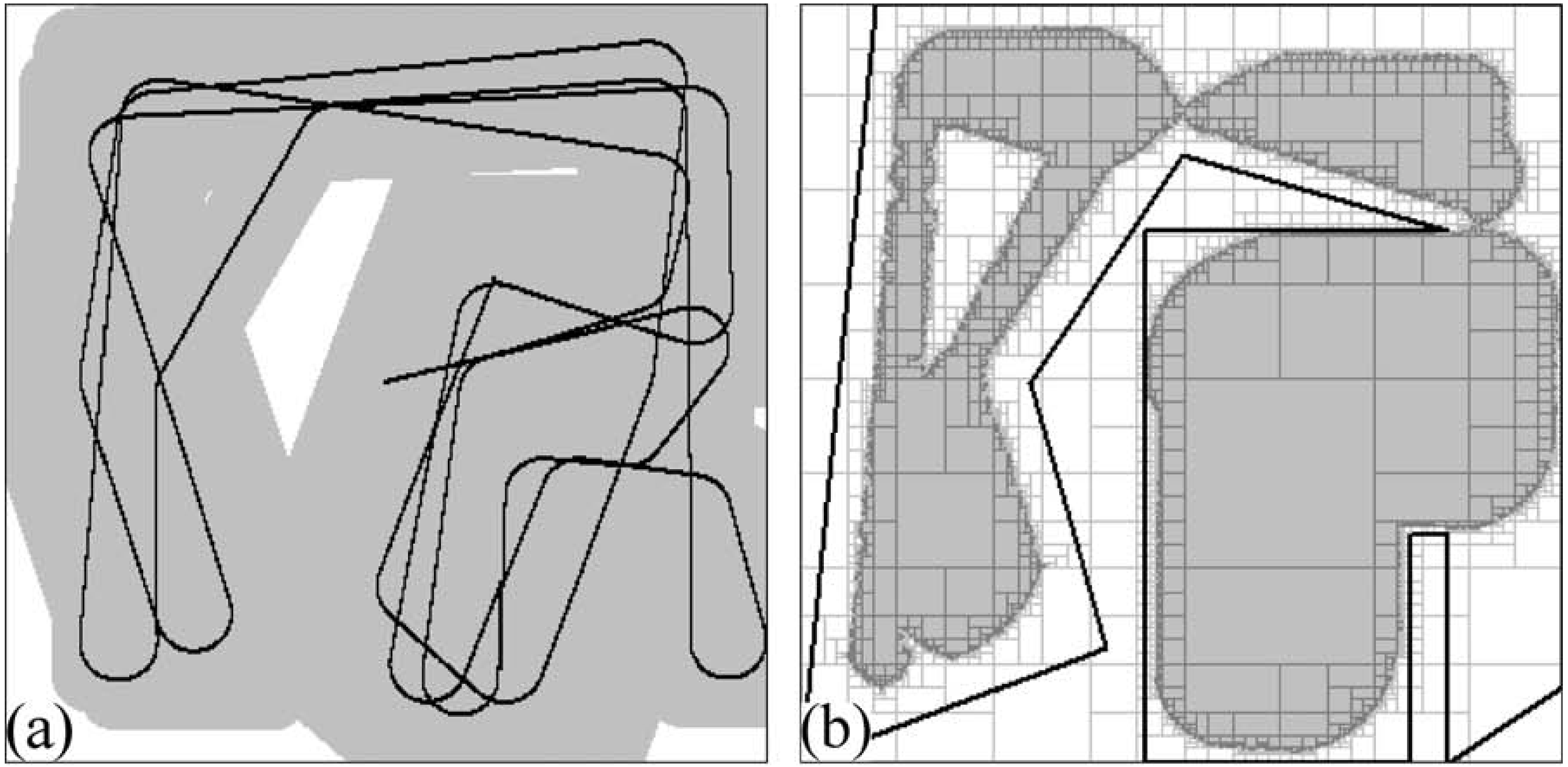
Representation of Occupancy Grids



Occupancy Grids

- Introduced by Moravec and Elfes in 1985
- Represent environment by a grid.
- Estimate the probability that a location is occupied by an obstacle.
- Key assumptions
  - Occupancy of individual cells (m[xy]) are independent
  - Robot position and direction (x,y,θ) is known!
- Maps sensory signals to (x,y)
- Repeated signals for the same location increases probability of an obstacle at that location.

Representation of Occupancy Grids



(a) Robot path, (b) Occupancy Grid

Occupancy Grids

- Pros:
  - Simple
  - Accurate
- Cons:
  - Accuracy is limited by cardinality (resolution) of the grid.
    - Computationally prohibitive if resolution is high; increased memory requirements.
    - Accuracy is low if resolution is low.
  - Require fixed-size environment:
    difficult to update if size of mapped area changes.

Probabilistic Mapping

- Noise in commands and sensors
  - Commands are not executed exactly

    (eg. Slippage leads to odometry errors)
  - Sonars have several error issues
    (eg. cross-talk, foreshortening, specular reflection)
  - Therefore, sonar data gives probabilities of where objects are located.

Probabilistic Mapping

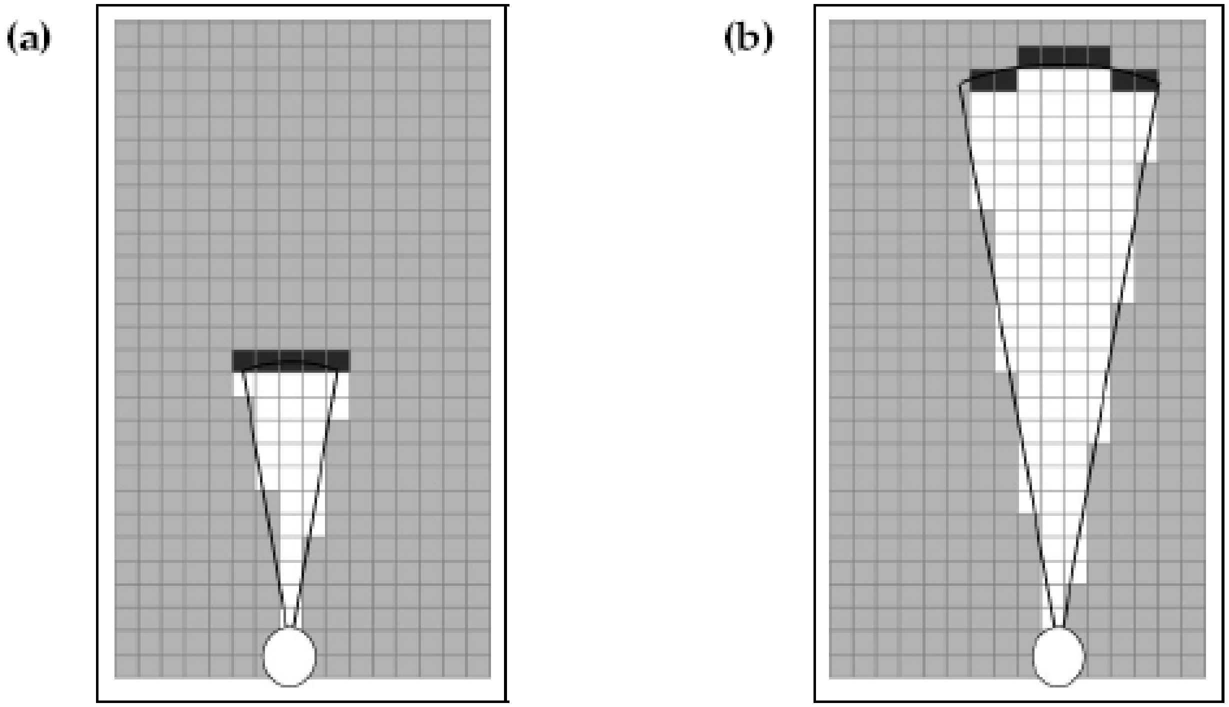
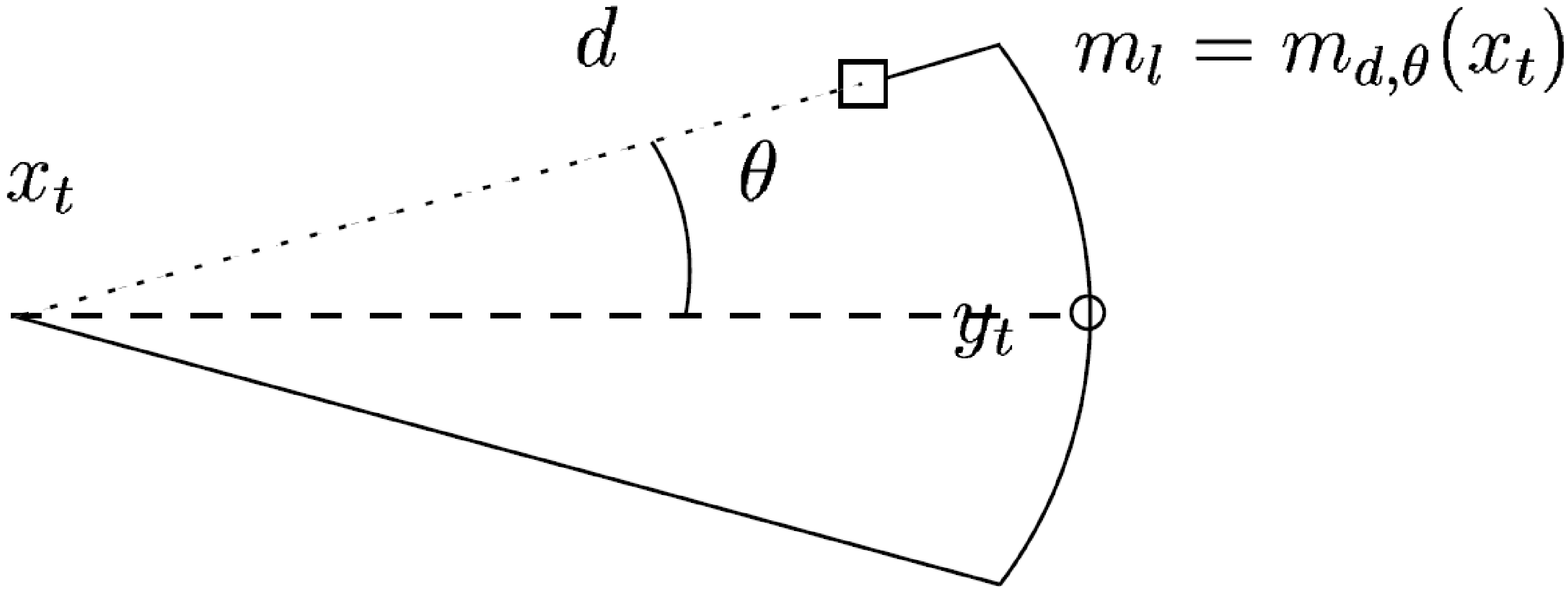
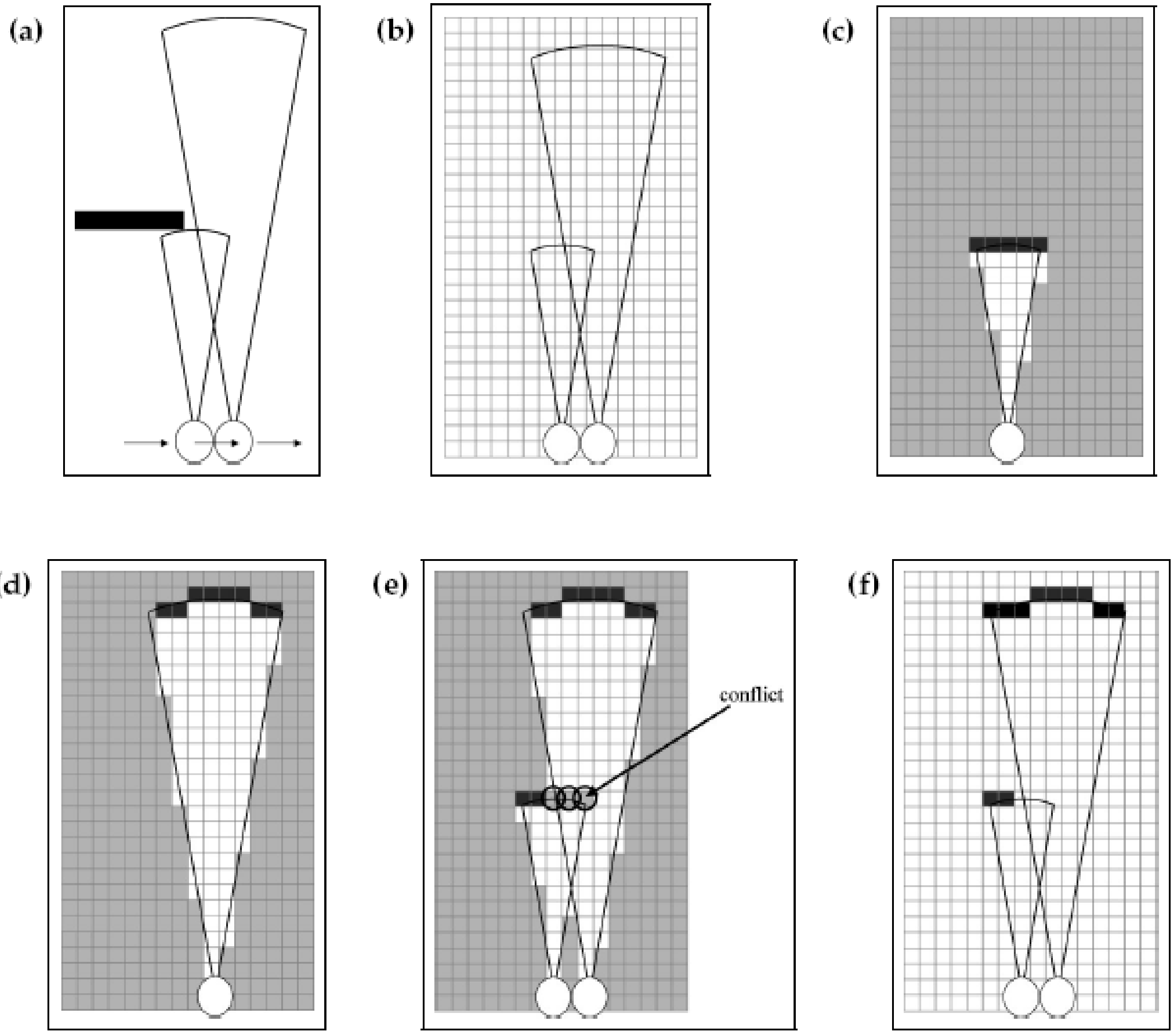


Figure 9.3 Two examples of an inverse measurement model **inverse_range_sensor_model** for two different measurement ranges. The darkness of each grid cell corresponds to the likelihood of occupancy. This model is somewhat simplistic; in contemporary implementations the occupancy probabilities are usually weaker at the border of the measurement cone.

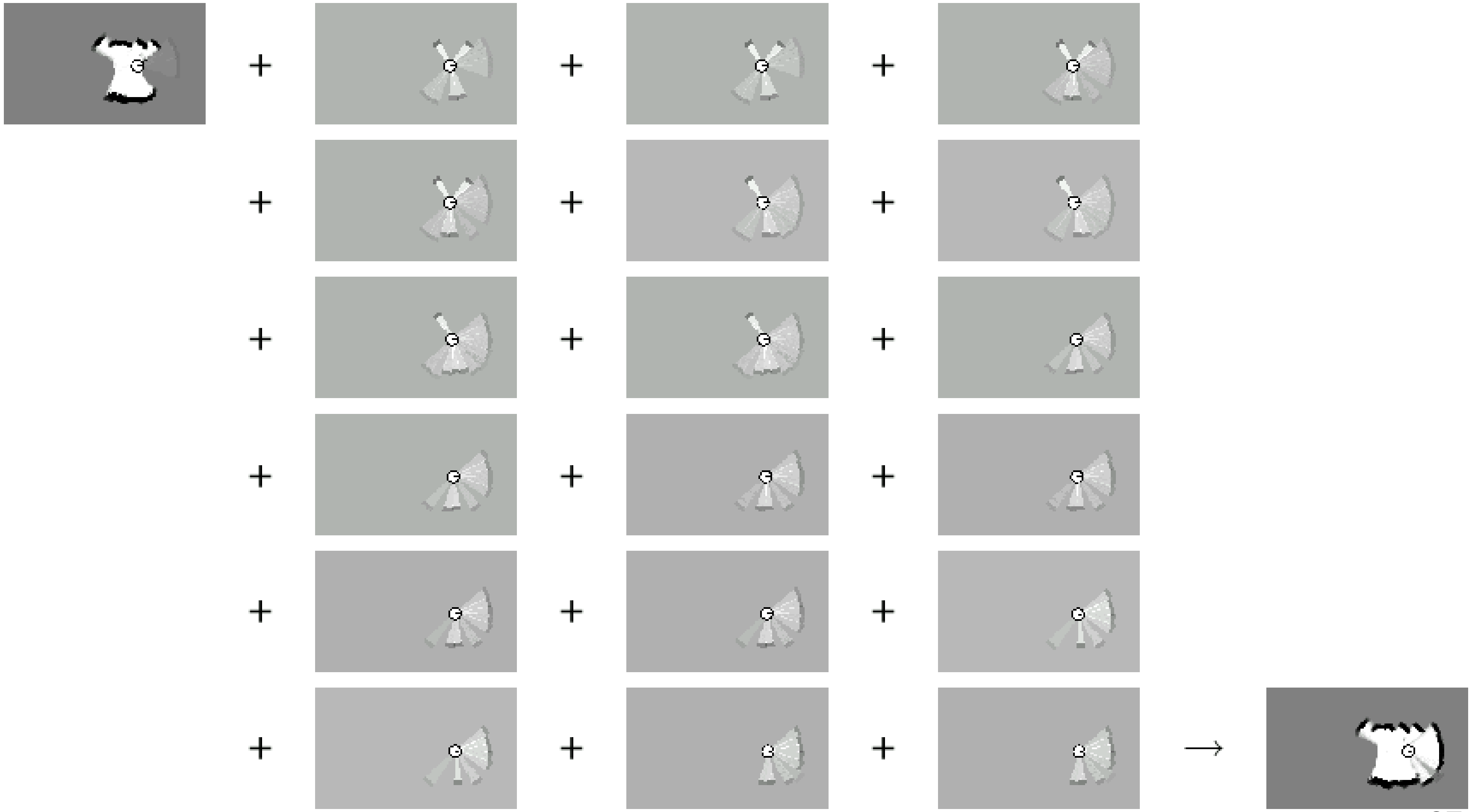
Key Parameters of the Model



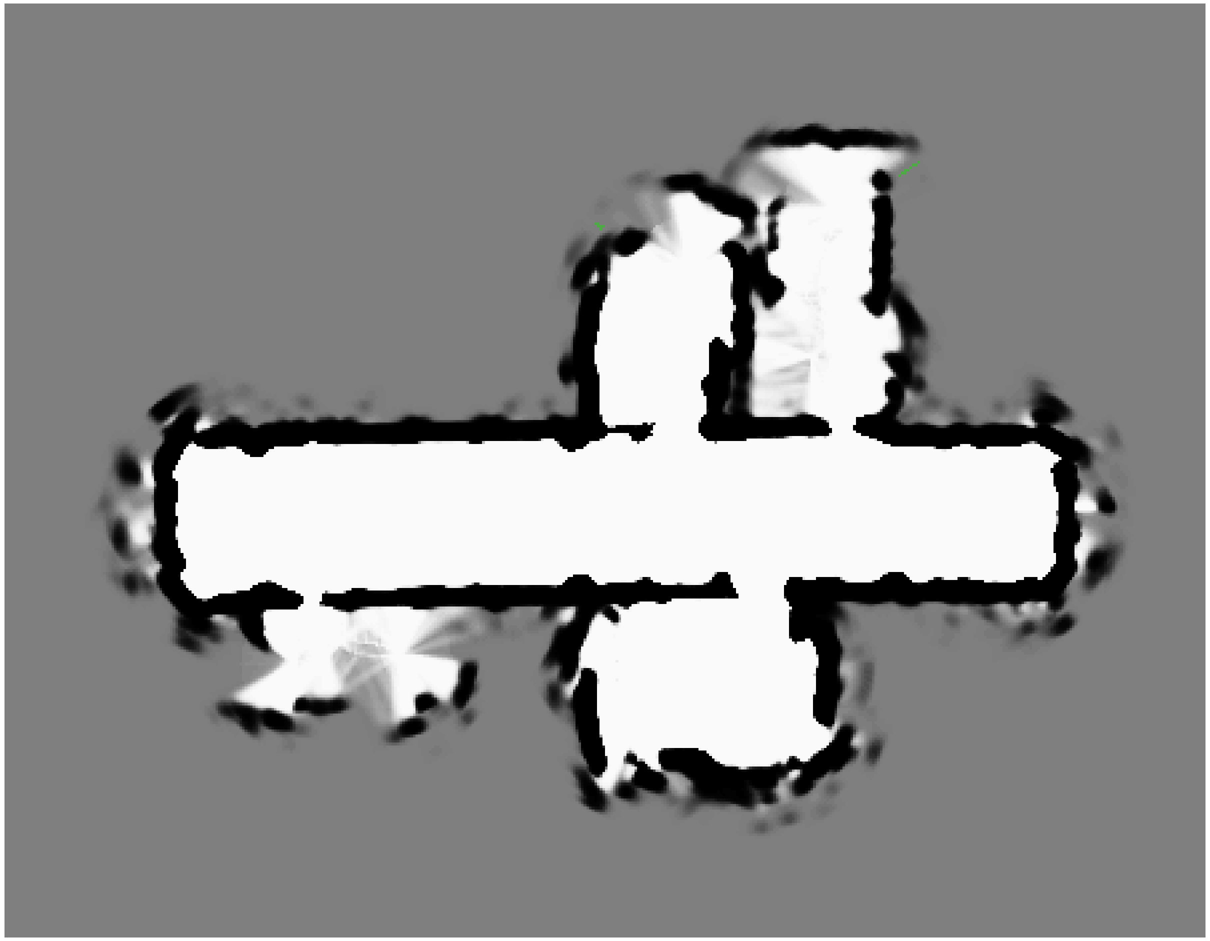
Mapping – Maintaining Dependencies



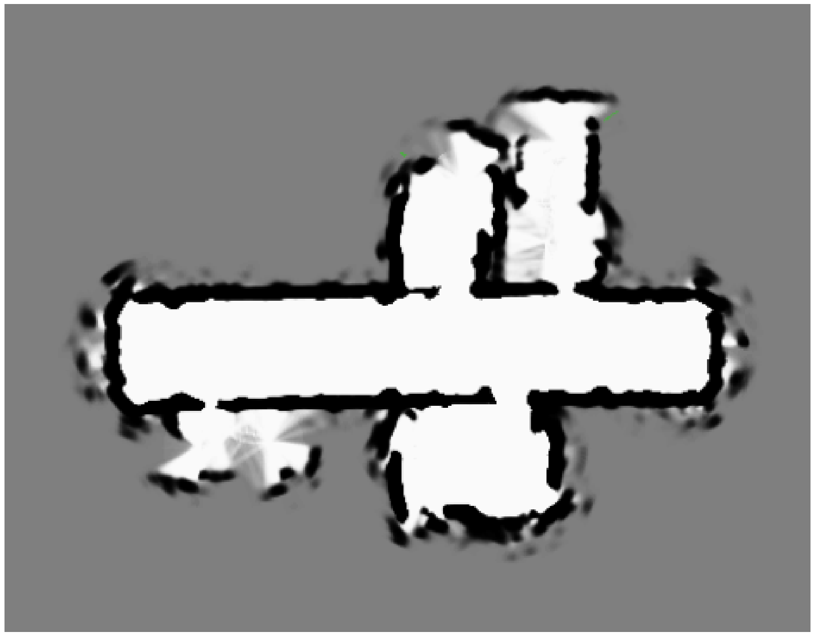
Incremental Updating of Occupancy Grids (Example)

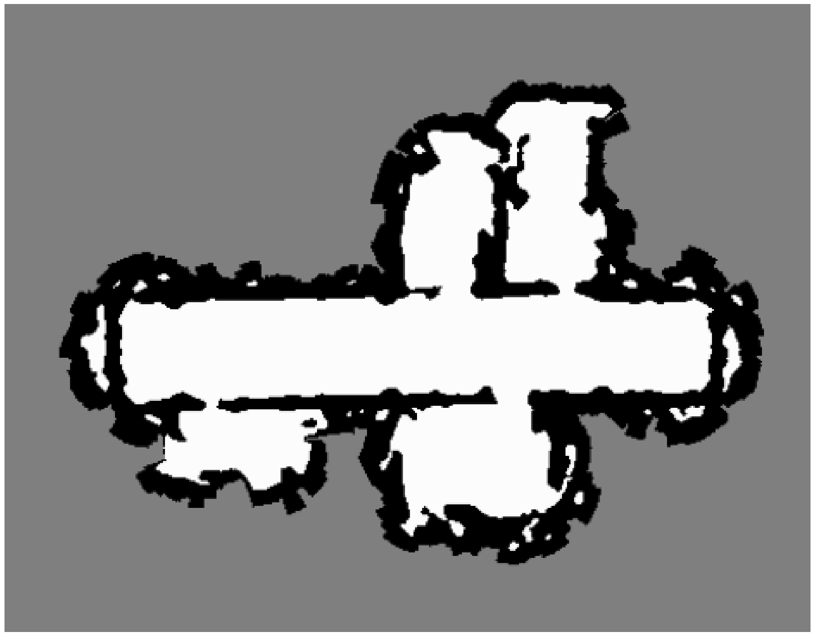


Resulting Map Obtained with Ultrasound Sensors



Resulting Occupancy and Maximum Likelihood Map

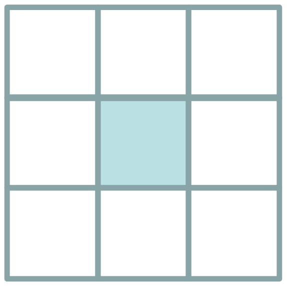

Probabilistic occupancy map


Maximum likelihood Map

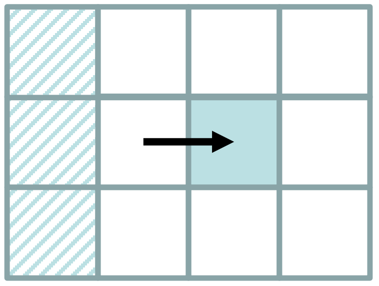
The maximum likelihood map is obtained by clipping the occupancy grid map at a threshold. The resulting map is binary.

Dynamically Expanding Occupancy Grids

- Variable-sized maps.
- Ability to increase size of map, if new areas are added to the environment.

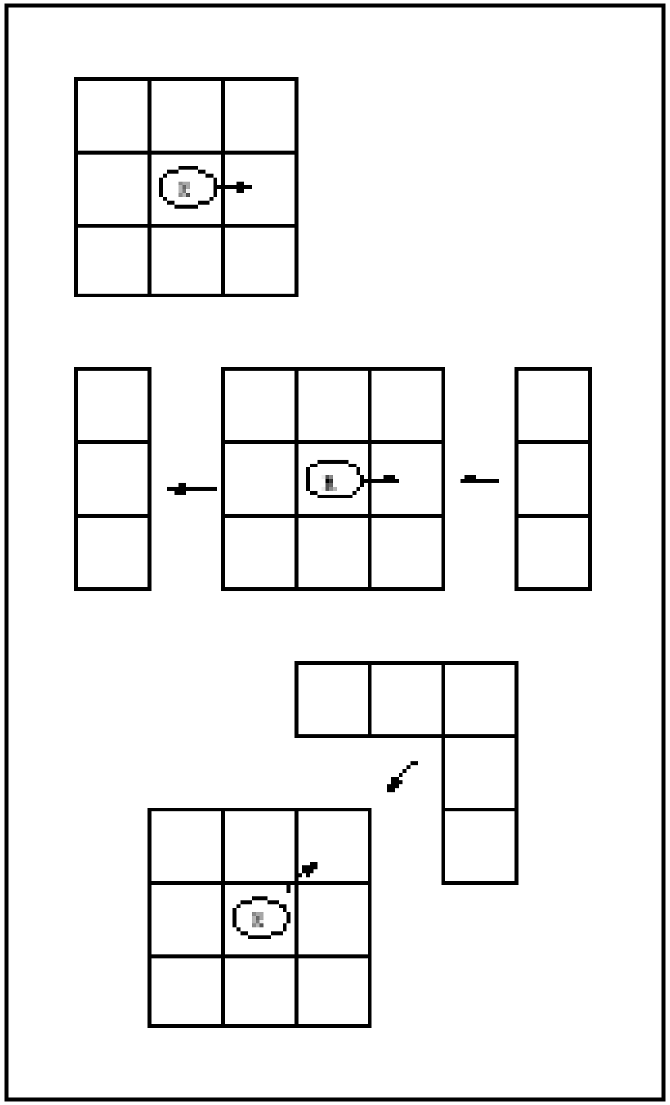


- Start mapping at center of nine-block grid.



- As robot explores, new cells are added.
  - Number of new cells depend on reach of sensor
  - Global map is stored outside the RAM in a file or a database.

Adding Cells to a DEOG



```
Algorithm_Grid ()
{
  initialize_grid ()
  initialize_position ()
  do forever
  {
    get_sensor_data ()
    update_grid ()
    manvoeur_robot ()
    calculate_position ()

    if new position outside grid block
    {
      add_cells ()
    }
  }
}
```

Dynamically Expanding Occupancy Grids

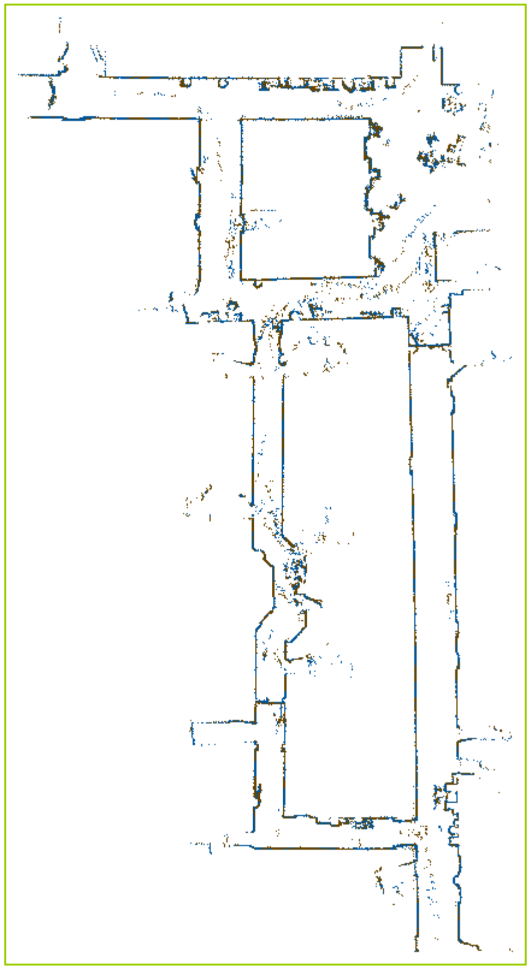
Pros:

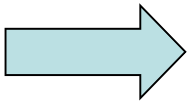
- Offer a solution for mapping changing environments.
- Saves RAM.
- Other useful information can be stored in the map.

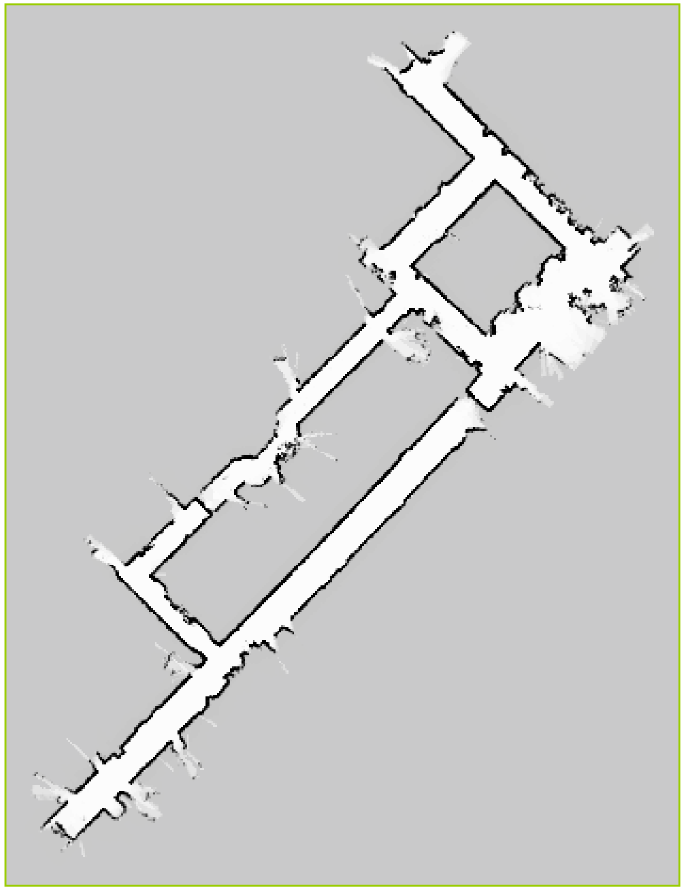
Cons:

- More complicated to program than regular occupancy grids.

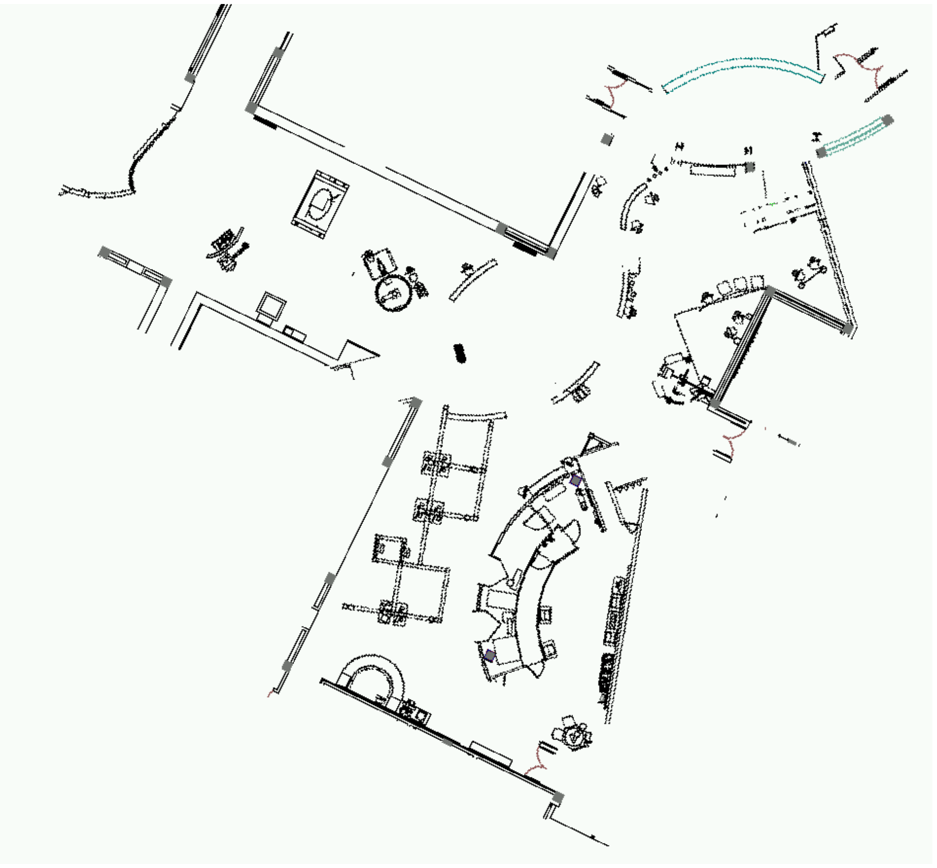
Occupancy Grids: From scans to maps



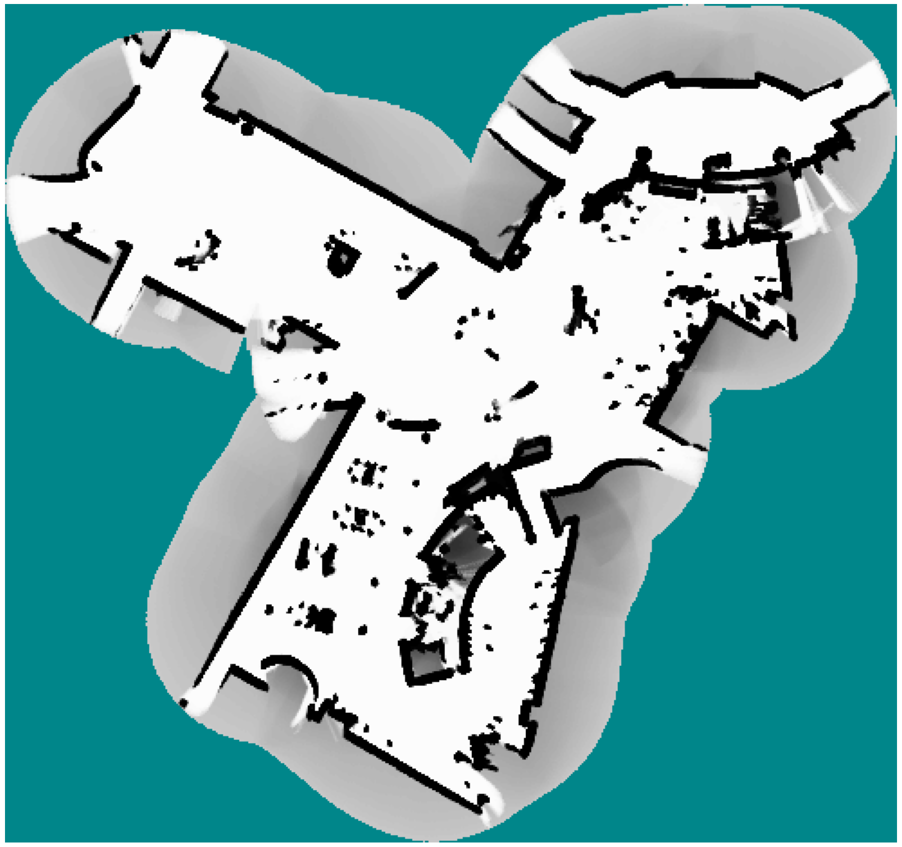




Tech Museum, San Jose



CAD map



Occupancy grid map

Localization

Where am I?

Methods:

- ❑ Iconic
- ❑ Feature-based
- ❑ Monte-Carlo

Iconic Localization

1. Use raw sensor data.
2. Use occupancy grids.
3. Compare current map with original map.
   - If original map has errors, localization can be very inaccurate.
   - Localization errors accumulate over time.

Approach:

- "pose": (x, y, θ)
  location, orientation
- Compare small local occupancy grid with stored global occupancy grid.
- Best fit pose is assumed to be the correct pose.

Feature-based Localization

- Compare currently extracted features with features marked in a map.
- Requires presence of easily extractable features in the environment.
- If features are not easily distinguishable, may mistake one for the other.

Monte Carlo Localization

- Probabilistic
  1. Start with a uniform distribution of possible poses (x, y, Θ)
  2. Compute the probability of each pose given current sensor data and a map
  3. Normalize probabilities
     - Throw out low probability points
- Performance
  - Excellent in mapped environments
  - Need non-symmetric geometries

Simultaneous Localisation And Mapping (SLAM)

- What is SLAM?
  - The problem has 2 stages
    - Mapping
    - Localization
  - The paradox:
    - In order to build a map, we must know our position
    - To determine our position, we need a map!
  - SLAM is like the chicken-egg problem
  - Solution is to alternate between the two steps.

SLAM – Multiple parts

- Landmark extraction
- Data association
- State estimation
- State update
- Landmark update

There are many ways to solve each of the smaller parts.

Hardware

- Mobile Robot - (with odometry)
- Range Measurement Device
  - Laser scanner – (Can’t be used underwater)
  - Sonar – (Not accurate)
  - Vision – (Cannot be used where not enough light is available)

The goal of the process

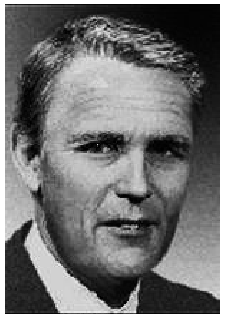
The SLAM process consists of number of steps.

- Use environment to update the position of the robot. Since the odometry of the robot is often erroneous we cannot rely directly on the odometry.
- We can use laser scans of the environment to correct the position of the robot.
- This is accomplished by extracting features from the environment and re observing when the robot moves around.

Extended Kalman Filter

An EKF (Extended Kalman Filter) is the heart of the SLAM process.

- Can filter inaccuracies (noise) in both state transitions and measurements/observations.
- Is a de facto standard in navigation systems (i.e. GPS).
- It is responsible for updating where the robot thinks it is based on the Landmarks (features).
- The EKF keeps track of an estimate of the uncertainty in the robots position and also the uncertainty in these landmarks it has seen in the environment.


Rudolf E. Kalman

Extended Kalman Filter

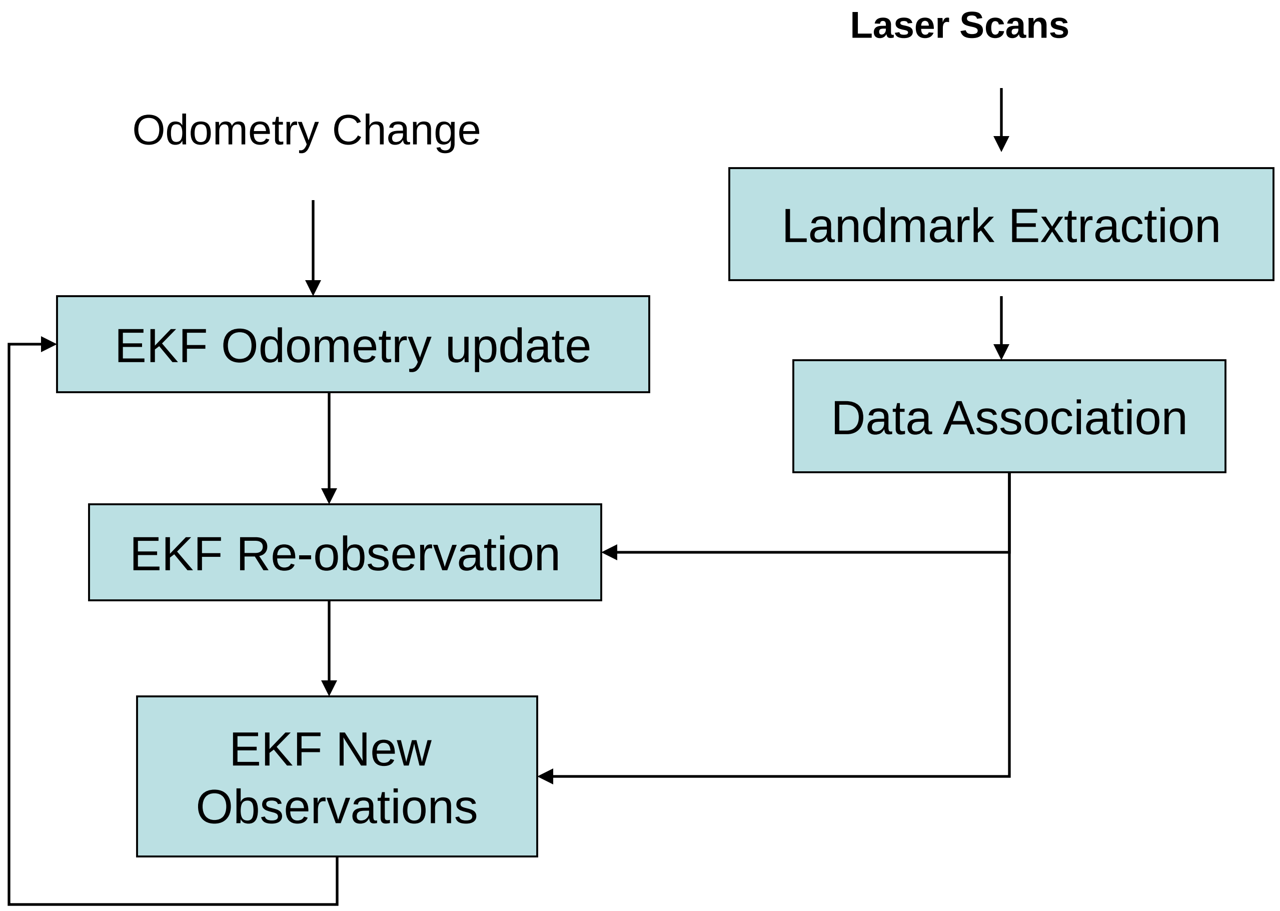
- Also known as linear quadratic estimation (LQE),
- Uses a series of measurements observed over time, containing statistical noise and other inaccuracies.
- By producing estimates of unknown variables Kalman Filtering tends to be more accurate than measurements based on a single measurement alone.
- Uses estimate of a joint probability distribution over the variables for each timeframe.

$$x_k = f\left(x_{\{k-1\}}, u_k\right) + w_k,\ z_k = h(x_k) + v_k$$

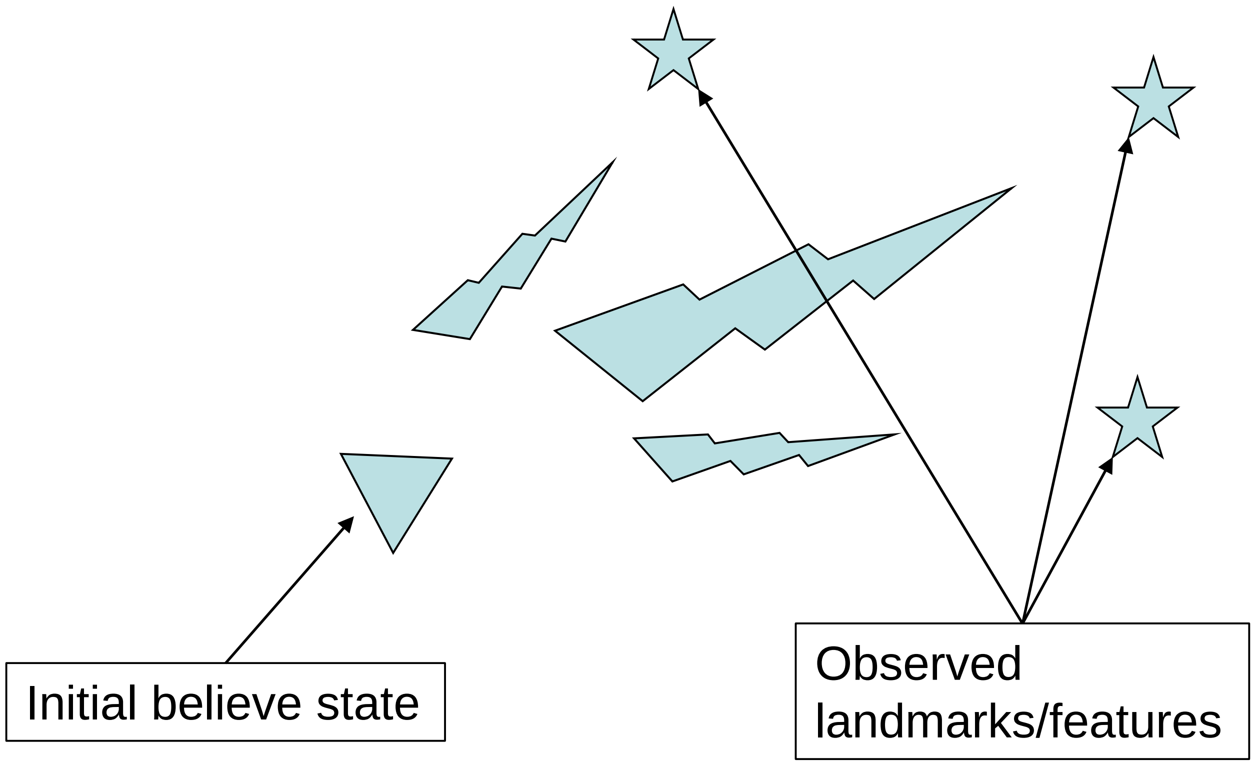
w_k and v_k are action and observation noises, u_k is action, x_k is state at time k.

Details i.e. in: *https://www.cs.cmu.edu/~robosoccer/cmrobobits/lectures/Kalman.ppt*

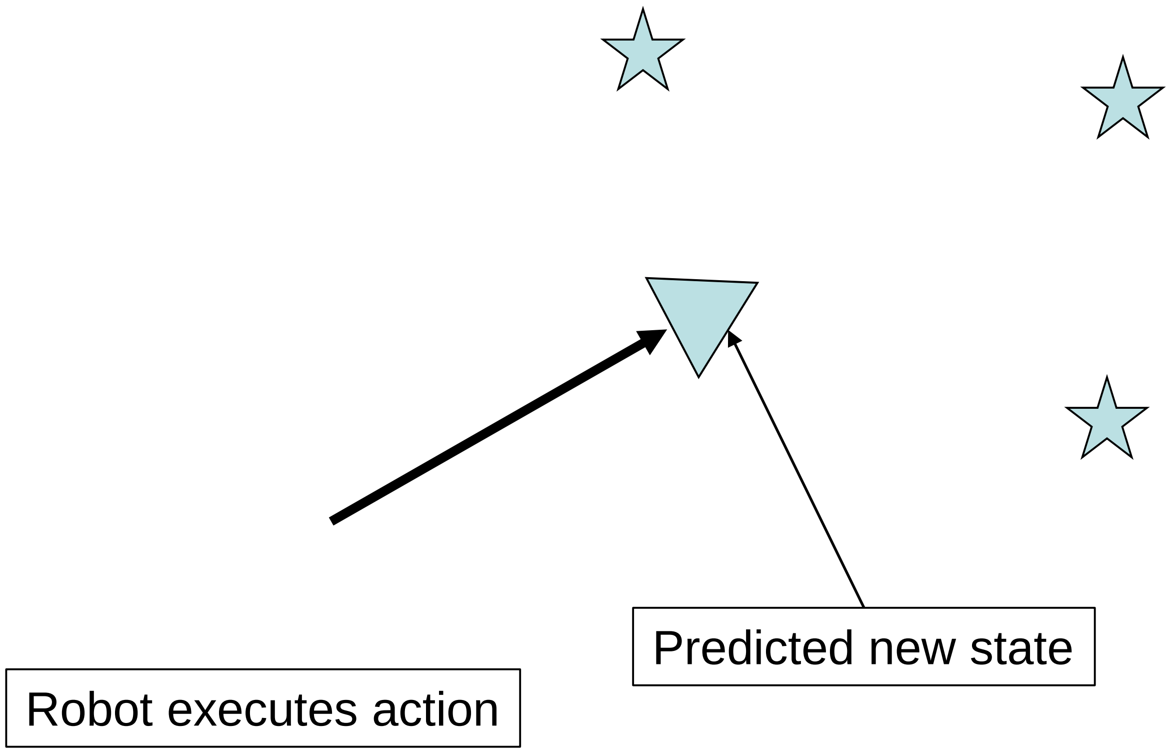
Overview of process



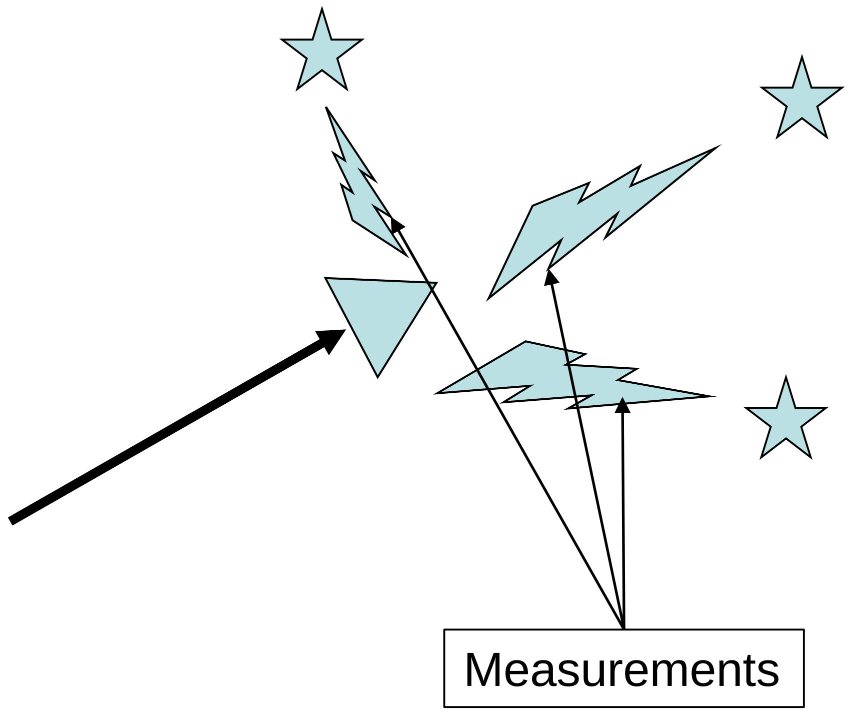
Extended Kalman Filter



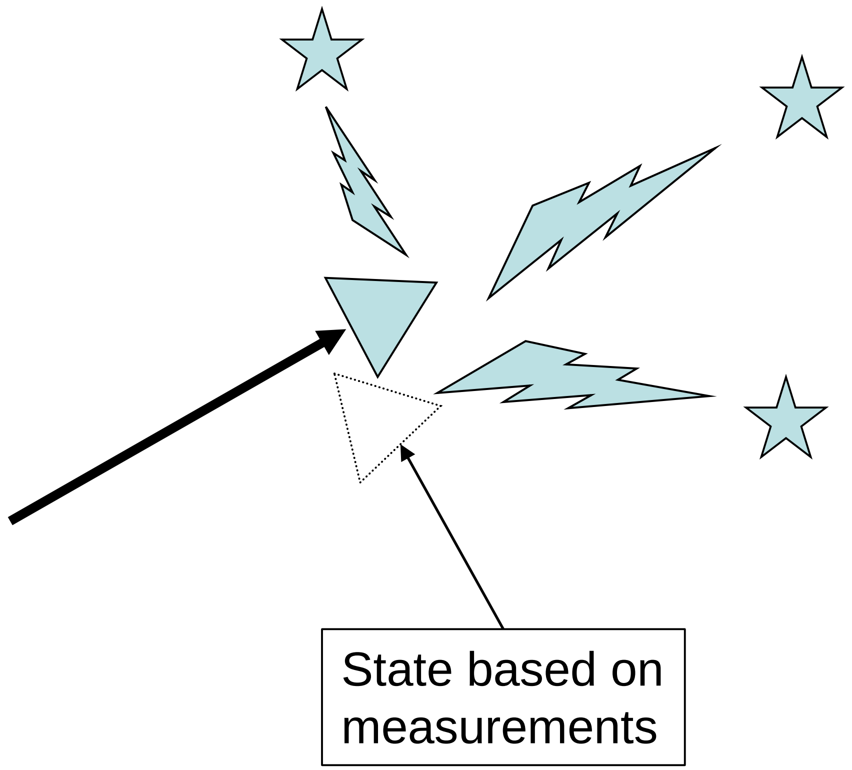
Extended Kalman Filter



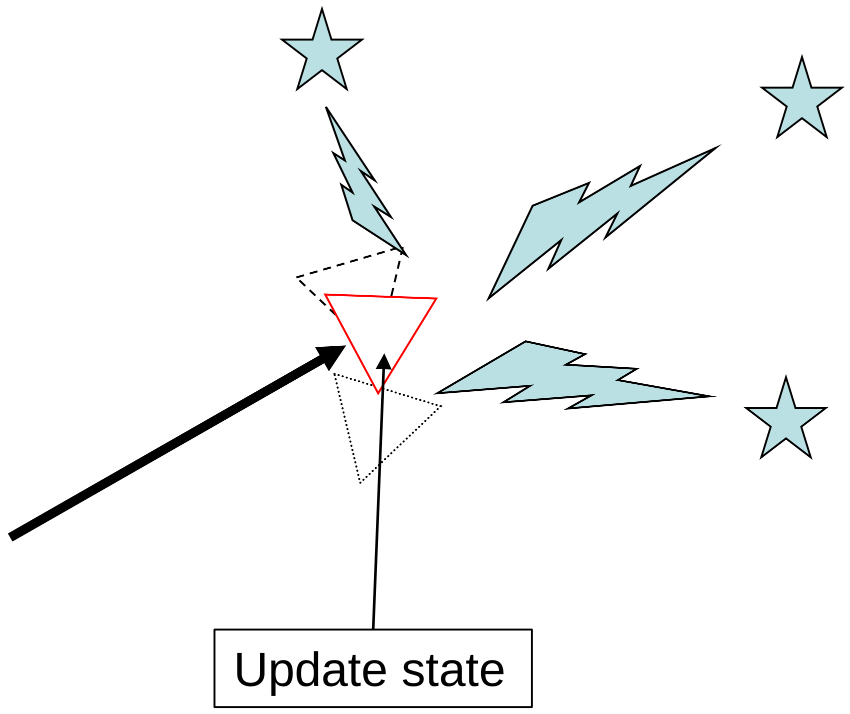
Extended Kalman Filter



Extended Kalman Filter



Extended Kalman Filter



Laser & Odometry data

- Laser data is the reading obtained from the scan
- The goal of the odometry data is to provide an approximate position of the robot
- The difficult part about the odometry data and the laser data is to get the timing right.

Laser & Odometry data

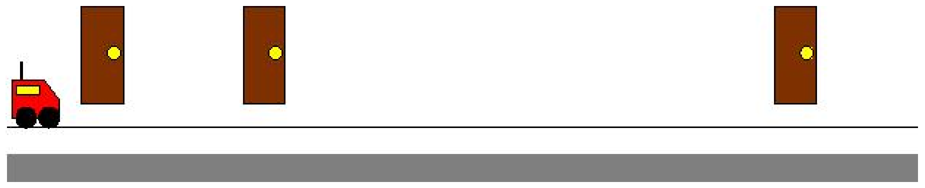
The Velodyne LiDAR system consists of 4 HDL-32E LiDAR sensors mounted on the top of a car.
https://www.youtube.com/watch?v=EBgdskiWlO8

Landmarks

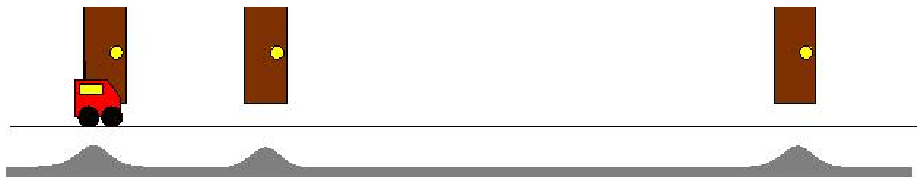
- Landmarks are features which can easily be re-observed and distinguished from the environment.
- These are used by the robot to find out where it is (to localize itself).

Landmarks

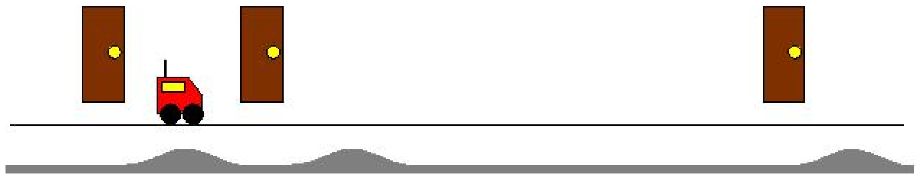
Initial state
detects nothing:



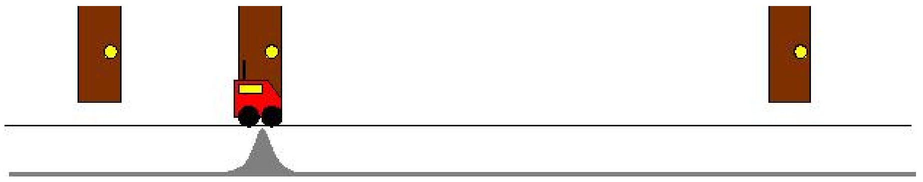
Moves and
detects landmark:



Moves and
detects nothing:



Moves and
detects landmark:



The key points about suitable Landmarks

- Landmarks should be easily re-observable.
- Individual landmarks should be distinguishable from each other.
- Landmarks should be plentiful in the environment.
- Landmarks should be stationary.

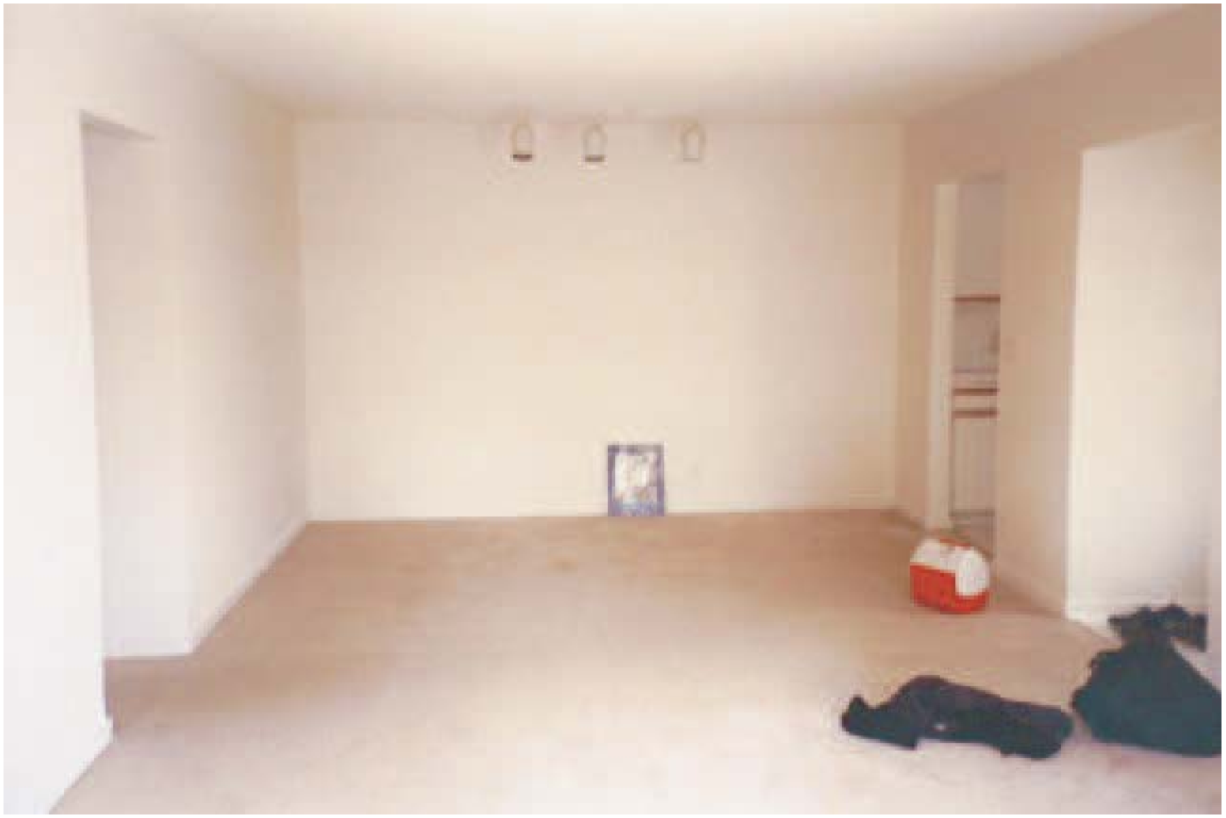
Landmark Extraction

- Once we have decided on what landmarks a robot should utilize we need to be able to somehow reliably extract them from the robots sensory inputs.
- The 2 basic Landmark Extraction Algorithms used are Spikes and RANSAC

Spike

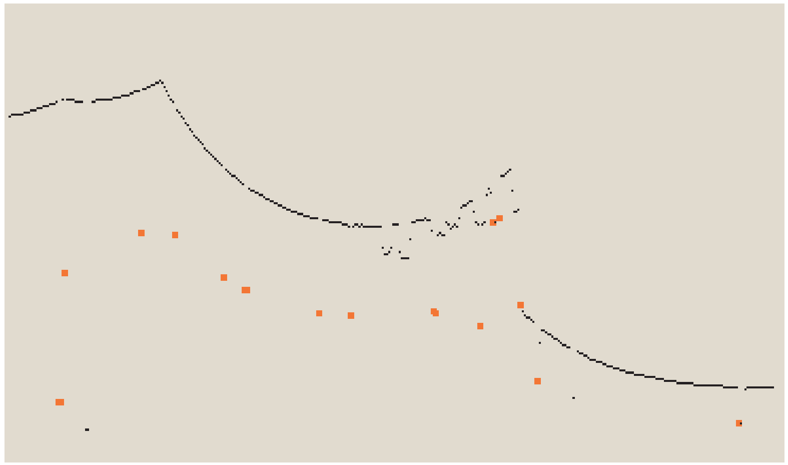
- The spike landmark extraction uses extrema to find landmarks.
- When some of the laser scanner beams reflect from a wall and some of the laser scanner beams do not hit this wall, but are reflected from other things.

Spike Example



In an indoor environment such as that used by our robot there are many straight lines but also well defined corners and objects between observer and wall. These could all be used as landmarks.

Spike

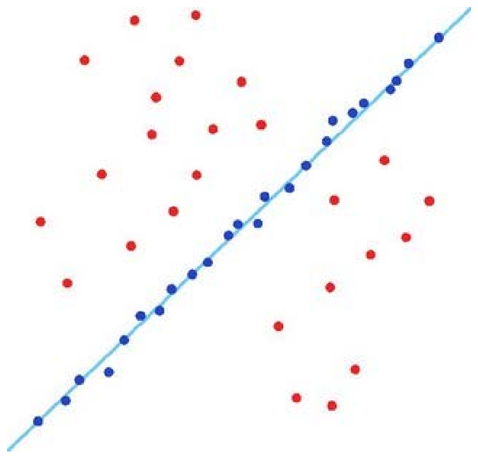


Spike landmarks. The red dots are table legs, spikes are corners, extracted as landmarks.

Spike landmarks rely on the landscape changing a lot between two laser beams. This means that the algorithm will fail in smooth environments.

RANSAC (Random Sampling Consensus)

- This method can be used to extract lines from a laser scan that can in turn be used as landmarks.
- RANSAC finds these line landmarks by randomly taking a sample of the laser readings and then using a least squares approximation to find the best fit line that runs through these readings.



Consensus

Data Association

- The problem of data association is that of matching observed landmarks from different (laser) scans with each other.

- Problems in Data Association
  - You might not re-observe landmarks every time.
  - You might observe something as being a landmark but fail to ever see it again.
  - You might wrongly associate a landmark to a previously seen landmark.

Algorithm – Nearest Neighbour Approach

- When you get a new laser scan use landmark extraction to extract all visible landmarks.
- Associate each extracted landmark to the closest landmark we have seen more than N times in the database.
- Pass each of these pairs of associations (extracted landmark, landmark in database) through a validation gate.
  - If the pair passes the validation gate it must be the same landmark we have re-observed so increment the number of times we have seen it in the database.
  - If the pair fails the validation gate add this landmark as a new landmark in the database and set the number of times we have seen it to 1.

Overview of the process

- Update the current state estimate
- Update the estimated state from re-observing landmarks.
- Add new landmarks to the current state.

Example

- Cisco 3D mapping device Zebedee: https://www.youtube.com/watch?v=gKPp2MYBYX0

Final Review – Open Areas

- There is the problem of closing the loop. This problem is concerned with the robot returning to a place it has seen before. The robot should recognize this and use the new found information to update the position.
- Furthermore the robot should update the landmarks found before the robot returned to a known place, propagating the correction back along the path.

References:

- Introduction to AI Robotics
  Dr. Robin Murphy
- Dynamically Expanding Occupancy Grids
  Bharani K. Ellore
- Multi-agent mapping using dynamic allocation utilizing a storage system
  Laura Barnes, Richard Garcia, Todd Quasny, Dr. Larry Pyeatt
- Robotic Mapping: A survey
  Sebastian Thrun
- Slam for dummies, by Soren Riisgaard & Morten Rufus Blas
- Wikipedia - Slam
- Minimal Slam for Efficient Floor-Planning, by Stephen Pfetsch