Texas Tech University Computer Science

Monte Carlo Localization using Dynamically Expanding Occupancy Grids

Karan M. Gupta

Agenda

- Introduction
- Occupancy Grids
- Sonar Sensor Model
- Dynamically Expanding Occupancy Grids
- Monte Carlo Localization
- Monte Carlo Localization with DEOGs

Introduction

- **An intelligent robot is a mechanical creature which can function autonomously.**

❑ **Intelligent – the robot does not do things in a mindless, repetitive way.**

❑ **Function autonomously – the robot can operate in a self-contained manner, under reasonable conditions, without interference by a human operator.**

Introduction

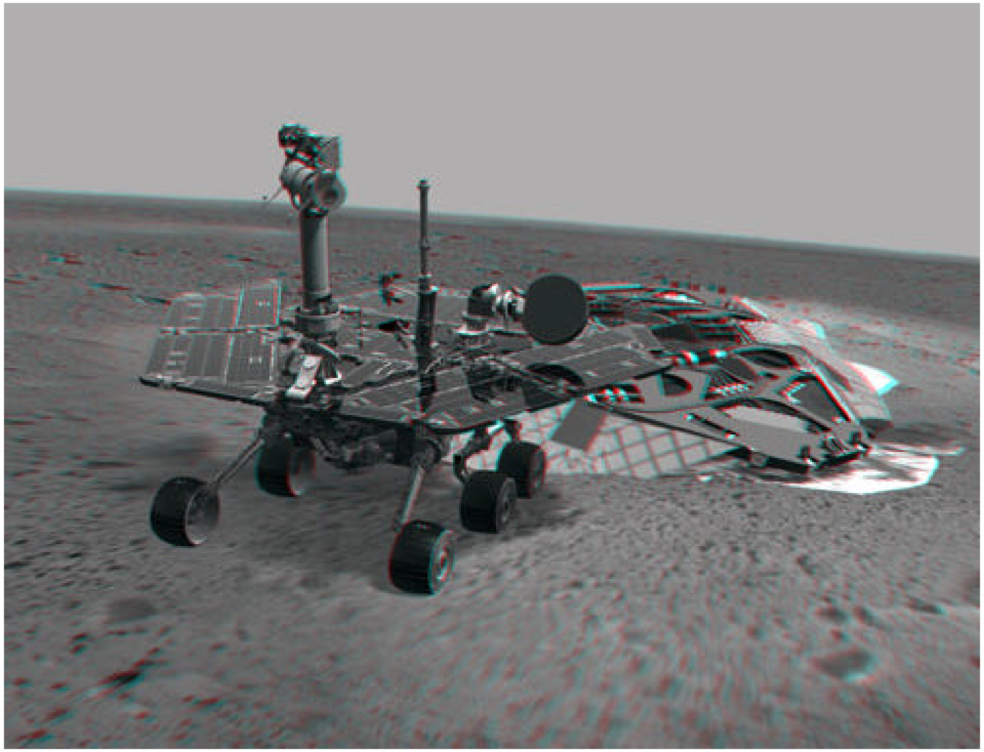
- **Robots in Museums**

Introduction

- **Personal Robots**

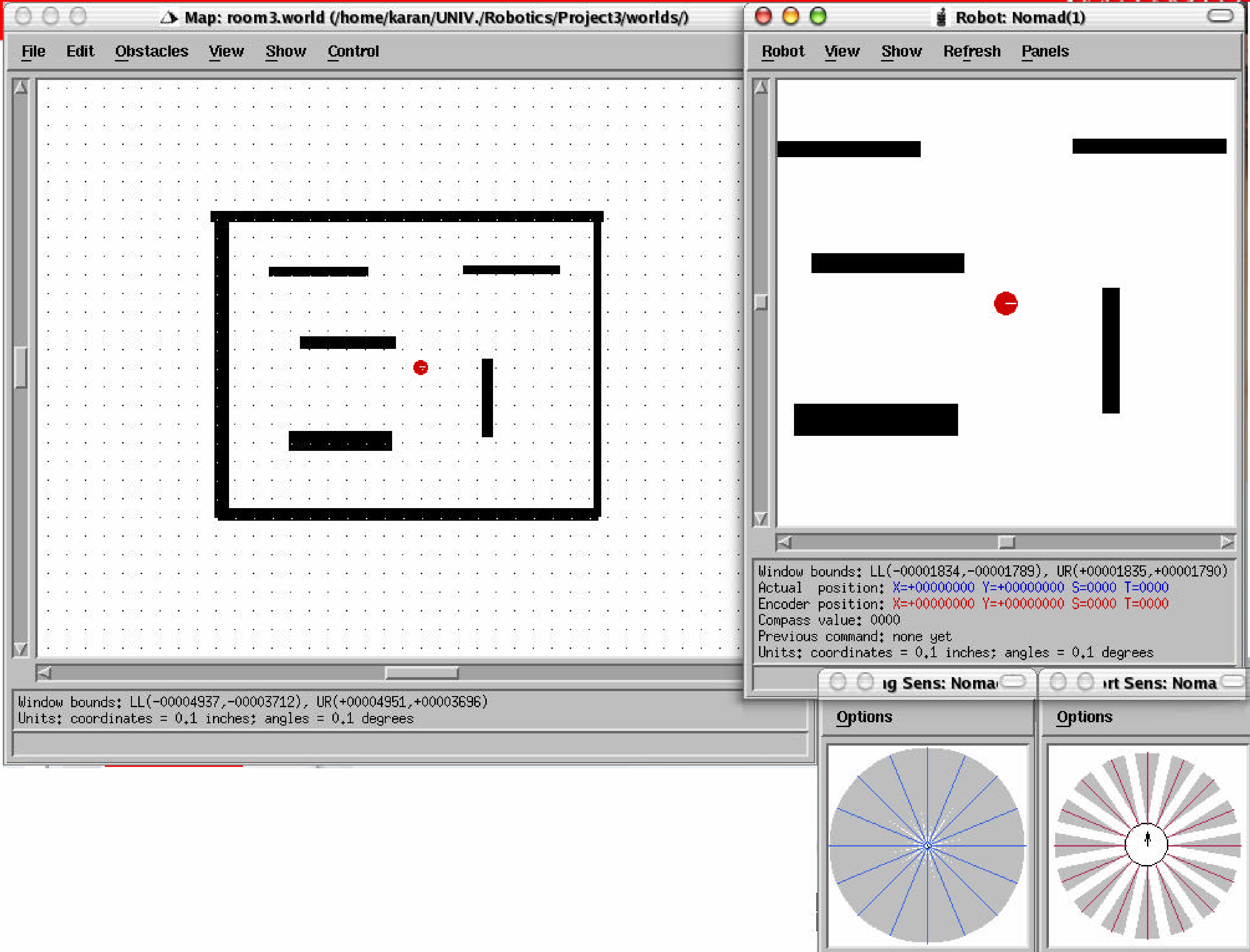
Introduction

- **Robots in Space**



Introduction

- **The problem of Navigation:**
  - **Where am I going?**
  - **What's the best way there?**
  - **Where have I been?**
  - **Where am I?**
  - **How am I going to get there?**

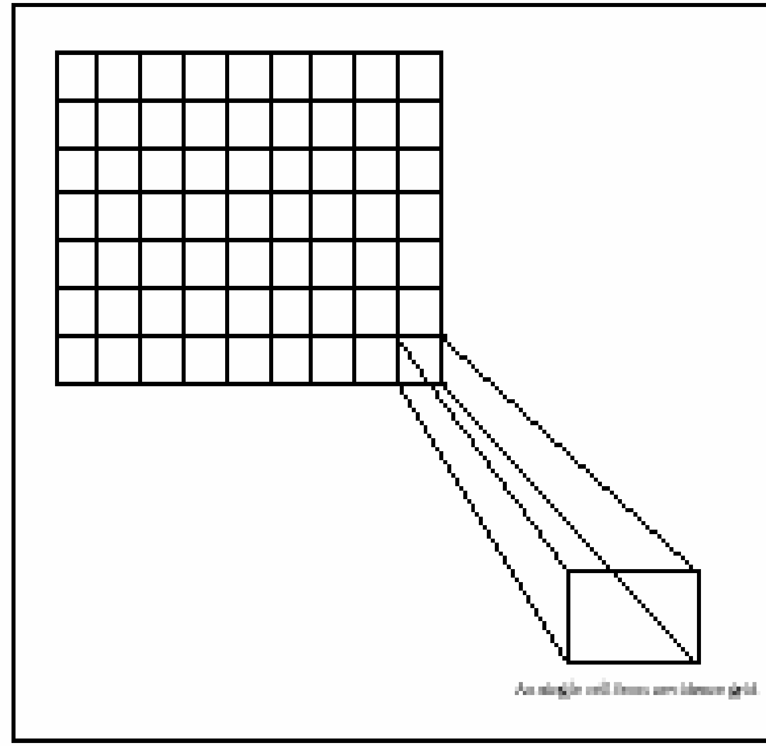

Map: room3.world (/home/karan/UNIV./Robotics/Project3/worlds/)
File
Edit
Obstacles
View
Show
Control
Window bounds: LL(-00004937,-00003712), UR(+00004951,+00003696)
Units: coordinates = 0.1 inches; angles = 0.1 degrees
Robot: Nomad(1)
Robot
View
Show
Refresh
Panels
Window bounds: LL(-00001834,-00001789), UR(+00001835,+00001790)
Actual position: X=+00000000 Y=+00000000 S=0000 T=0000
Encoder position: X=+00000000 Y=+00000000 S=0000 T=0000
Compass value: 0000
Previous command: none yet
Units: coordinates = 0.1 inches; angles = 0.1 degrees
Options
Options

Occupancy Grids

- **Originally proposed by Moravec and Elfes: based on ultrasonic range measurements.**
- **A tool to construct an internal model of *static* environments based on sensor data.**
- **Creates map incrementally using belief values**
- **Can be directly applied to localization, path planning and navigation**
- **The environment to be mapped is divided into regions.**
- **Each grid cell is an *element* and represents an area of the environment.**

Occupancy Grids

- **Two-dimensional grid of cells**
- **Each cell represents a small discrete region of the world**
- **Each cell contains a value that indicates if the cell (and corresponding region in the environment) is either *occupied* or *empty***
- **Pros**
  - **Simple**
  - **Accurate**
- **Cons**
  - **Require fixed-size environment: difficult to update if size of mapped area changes.**

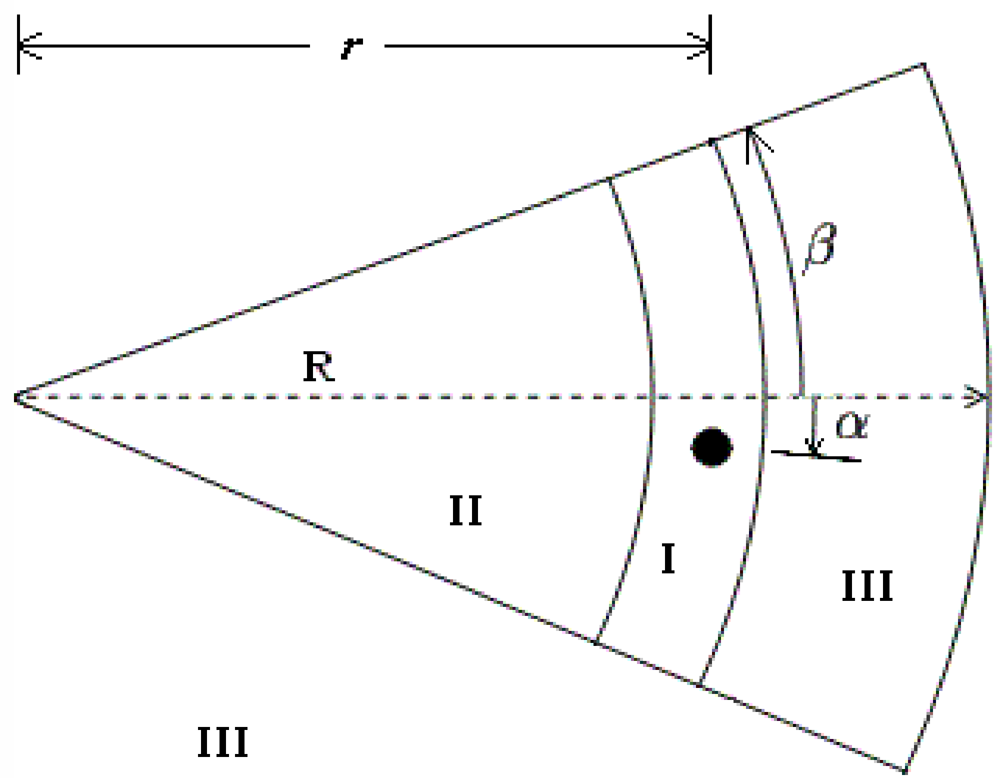


Sonar Sensor Model

Region I: The sonar reading indicates a reflection from an object lying in this region. So this region is *probably* occupied.

Region II: No reading was returned from anything in this region. So this is the area that is *probably* empty.

Region III: The reading has been returned by Region II, so this is the area that is *undetected* by current sonar reading.



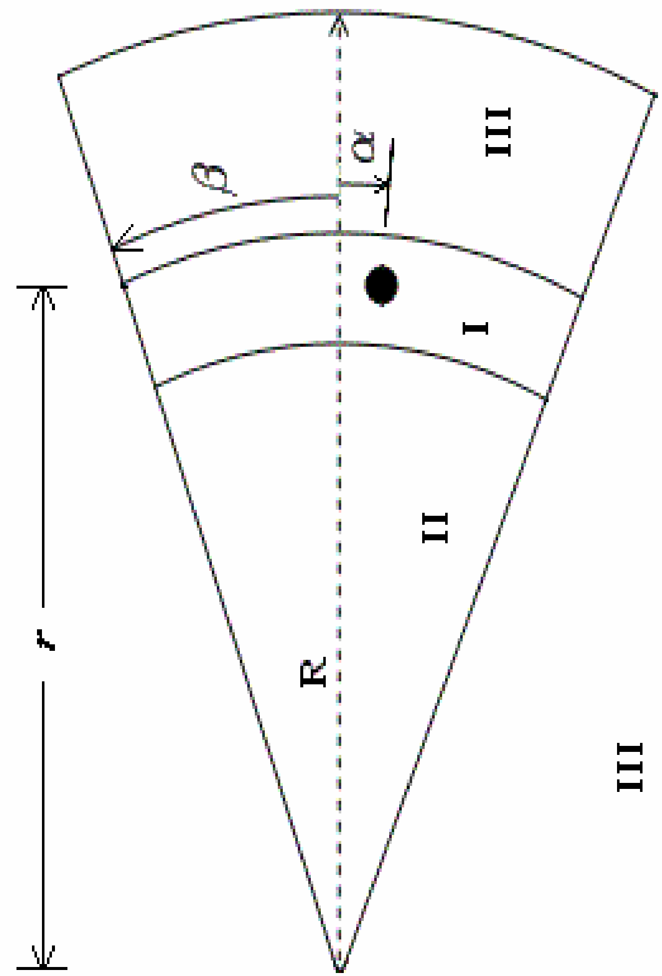
Sonar Sensor Model

Region I:

$$P(Occ) = \frac{\frac{R-r}{R} + \frac{b-a}{b}}{2} \text{ x } MaxOcc$$

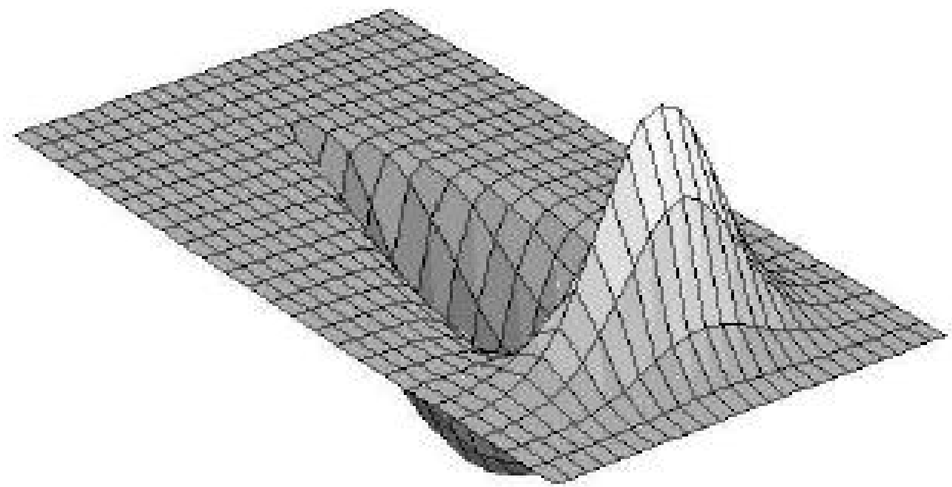
Region II:

$$P(Emp) = \frac{\frac{R-r}{R} + \frac{b-a}{b}}{2}$$



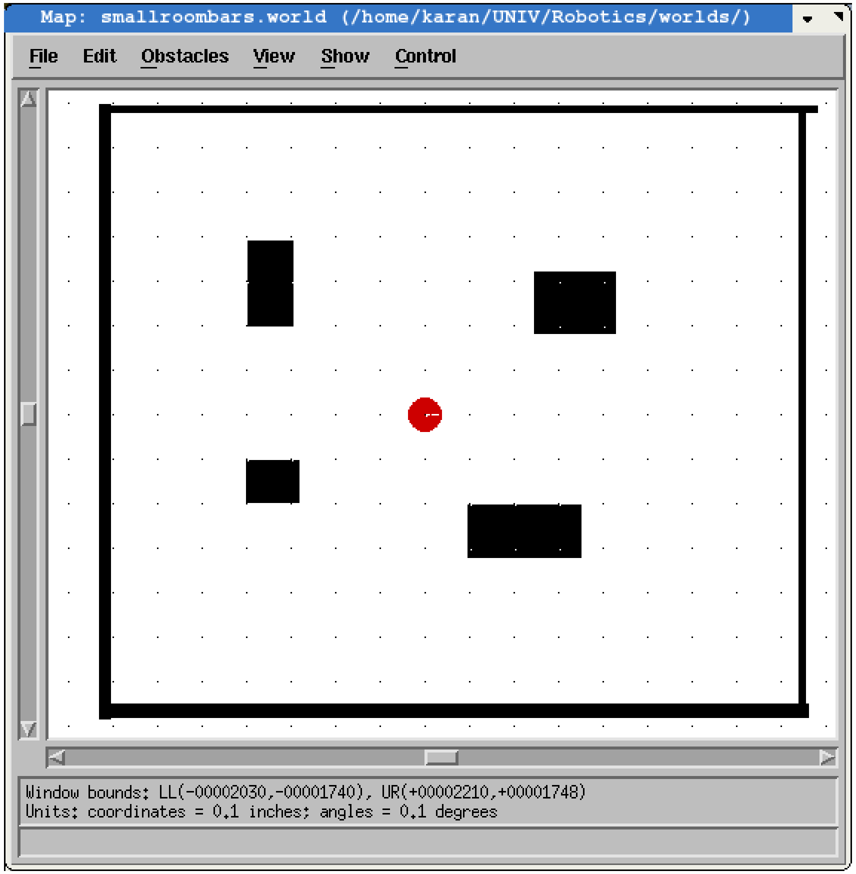
Sonar Sensor Model

- **Why 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)

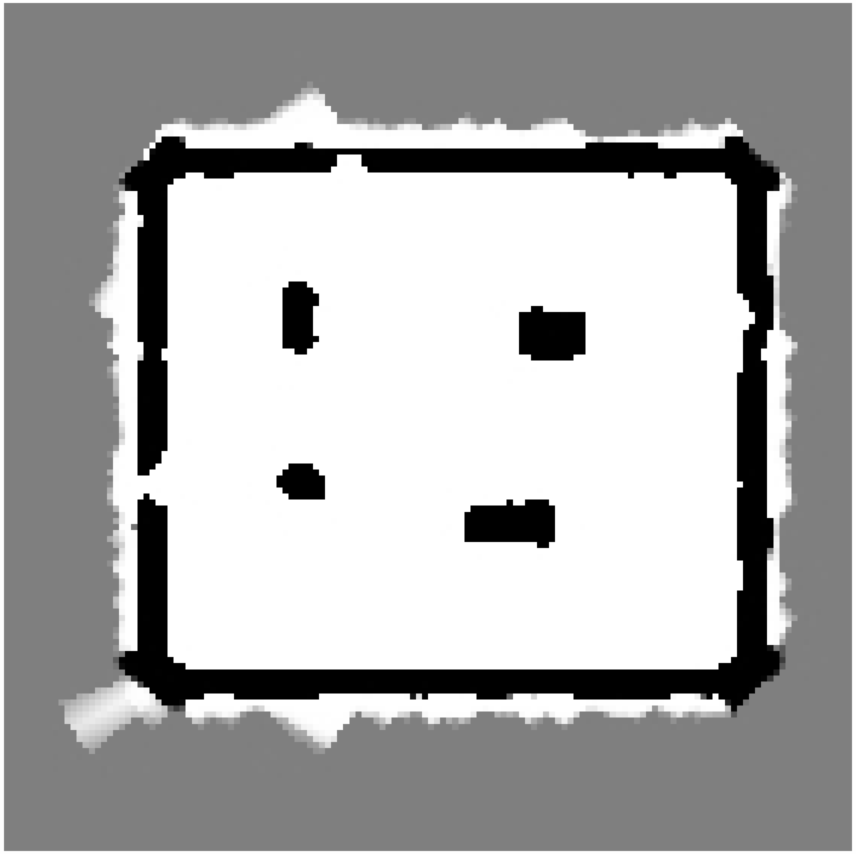


Occupancy Grids

Given Map

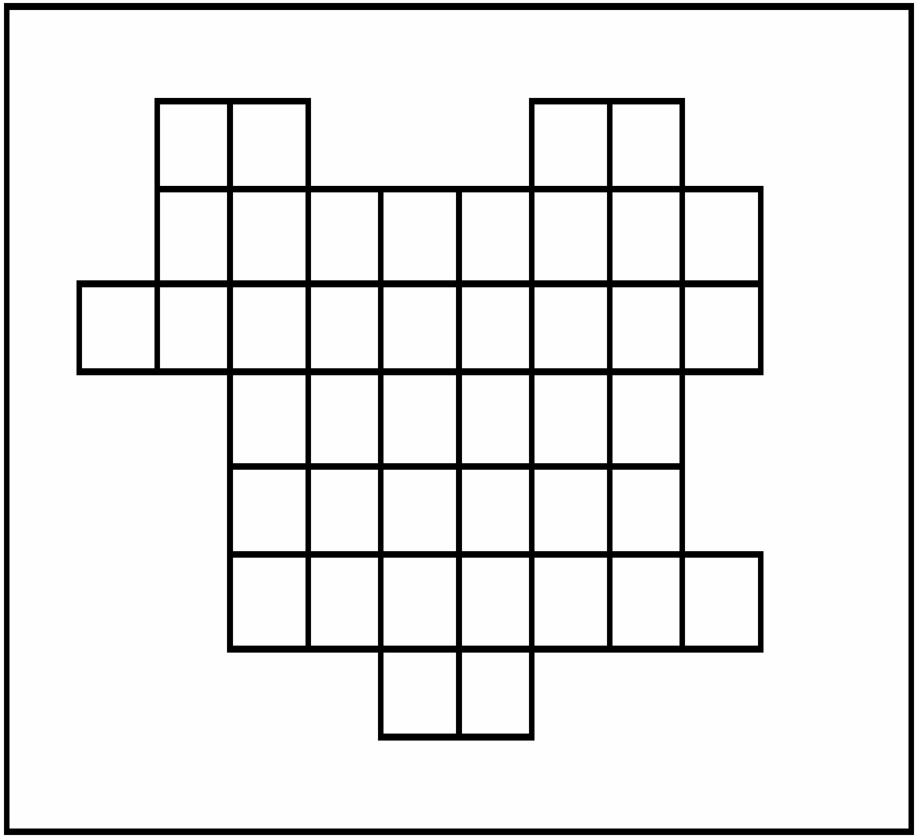


Created Map



Dynamically Expanding Occupancy Grids

- Variable-sized maps
- Ability to increase size of map, if new areas are added to the environment
- As robot explores, new cells are added
- Global map is stored outside the RAM in a Centralized Storage System
- Implemented using a Centralized Storage System



Dynamically Expanding Occupancy Grids

- Best (the only?) solution for mapping changing environments.
- Saves RAM
- Other useful information can be stored in the map
- More complicated to program than regular occupancy grids

Monte Carlo Localization

- **To navigate reliably, a mobile robot must know where it is.**
- **Robot's *pose*:**

  X = (location, orientation) = [x, y, ?]
- **Mobile robot localization: the problem of estimating a robot's pose relative to its environment.**
- ***"the most fundamental problem to providing a mobile robot with autonomous capabilities"* – IEEE Transactions on Robotics and Automation.**

Monte Carlo Localization

- **Three Flavors:**
  - Position tracking
    - » Robot knows its initial pose.
    - » As the robot moves, its pose changes.
    - » The problem is to compensate small, incremental errors in a robot's odometry (x, y, ?).
  - Global localization problem
    - » Robot does not know its initial pose.
    - » The problem is to look at the surroundings and make multiple distinct hypotheses about its location.
    - » More challenging problem than Position Tracking.
  - Kidnapped robot problem
    - » A well-localized robot is teleported to some other place without being told!
    - » Tests robot's ability to recover from catastrophic localization failures.

Monte Carlo Localization

- Remember: The Navigation Problem
  - Where am I going?
  - What's the best way there? —— Path Planning
  - Where have I been? —— Mapping
  - Where am I? —— Localization
- Global Localization
  - Enables robot to make use of existing maps, which allows it to plan and navigate reliably in complex environments.
- Position Tracking (Local Tracking)
  - Useful for efficient navigation and local manipulation tasks.

Monte Carlo Localization

The Concept:

- ❑ **Compare small local occupancy grid with stored global occupancy grid.**
- ❑ **Best fit pose is correct pose.**

Probabilistic

- **1. Start with a uniform distribution of possible poses (x, y, Q)**
- **2. Compute the probability of each pose given current sensor data and a map**
- **3. Normalize probabilities**
  - **Throw out low probability points**

Monte Carlo Localization

Bayesian Approach

- We want to estimate pose of robot at *k*, given knowledge about the initial state and all movements Z^k up to current time.
- k = current time-step
- $Z^k = \{z_k,\ i = 1..k\}$
- $\mathbf{x} = [x,\ y,\ ?]^T$ ——— the current state of the robot
- Find the *posterior density* $= p(\mathbf{x}_k | Z^k)$ = probability of being in $\mathbf{x}$ at time *k*, if Z^k takes place.
- To localize the robot we need to recursively compute $p(\mathbf{x}_k | Z^k)$ at each time-step.

Monte Carlo Localization

Bayesian Approach

- Two phases to compute $p(\mathbf{x}_k | Z^k)$:
- Prediction Phase:
  - Predict current position using only the history of the robot's movements.

$$p(\mathbf{x}_k | Z^{k-1}) =)\ p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_{k-1})\ p(\mathbf{x}_{k-1} | Z^{k-1})\ d\mathbf{x}_{k-1}$$

- Update Phase:
  - Incorporate information from sensors (compare what is observed to what is on the map).

$$p(\mathbf{x}_k | Z^k) = \frac{p(\mathbf{z}_k | \mathbf{x}_k)\ p(\mathbf{x}_k | Z^{k-1})}{p(\mathbf{z}_k | Z^{k-1})}$$

- Repeat the process for every time-step
- Use an estimate function: maximum or mean etc. to get the current position.

Monte Carlo Localization

- **Represent the posterior density $p(\mathbf{x}_k|Z^k)$ by a set of N random samples (*particles*) that are drawn from it.**
- **Set of particles = $S_k = \{s^i_k;\ i = 1..N\}$**
- **Density is reconstructed from the samples using an estimator, e.g. histogram.**
- **New localization goal:**
  - **Recursively compute at each time-step *k*, the set of samples S^k that is drawn from $p(\mathbf{x}_k|Z^k)$.**

Monte Carlo Localization

- **Prediction Phase:**
  - **Start from set of particles S_{k-1} computed in previous iteration; apply motion model to each particle s^i_{k-1} by sampling from the density $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$:**

```
for each particle s^i_{k-1} :
    draw one sample s'^i_k from p(x_k|s^i_{k-1}, u_{k-1})
```

  - **We have a new set S'_k that approximates a random sample from the predictive density $p(\mathbf{x}_k|Z^{k-1})$.**
  - **The prime in S'_k indicates that we have not yet applied any sensor readings at time k.**

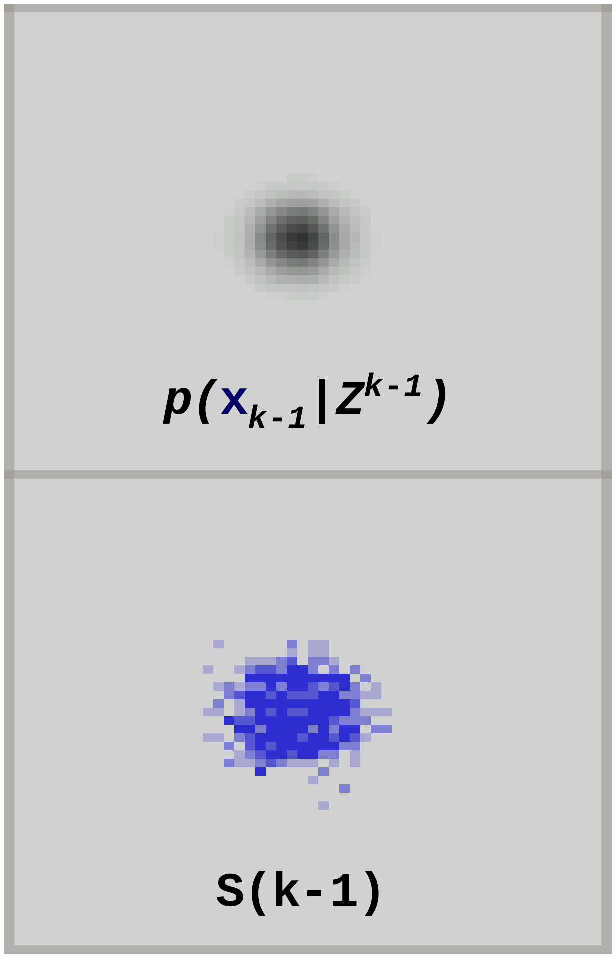
Monte Carlo Localization

- **Update Phase:**
  - We take sensor readings $\mathbf{Z}_k$ into account.
  - Weight each sample in S'_k by a weight which is the likelihood of s'^i_k given $\mathbf{Z}_k$.
  - Weight = $m^i_k = p(\mathbf{Z}_k \mid s'^i_k)$
  - Obtain S_k by resampling from this weighted set:

```
for j = 1..N:
    draw one S_k sample s^j_k from {s'^i_k, m^i_k}
```

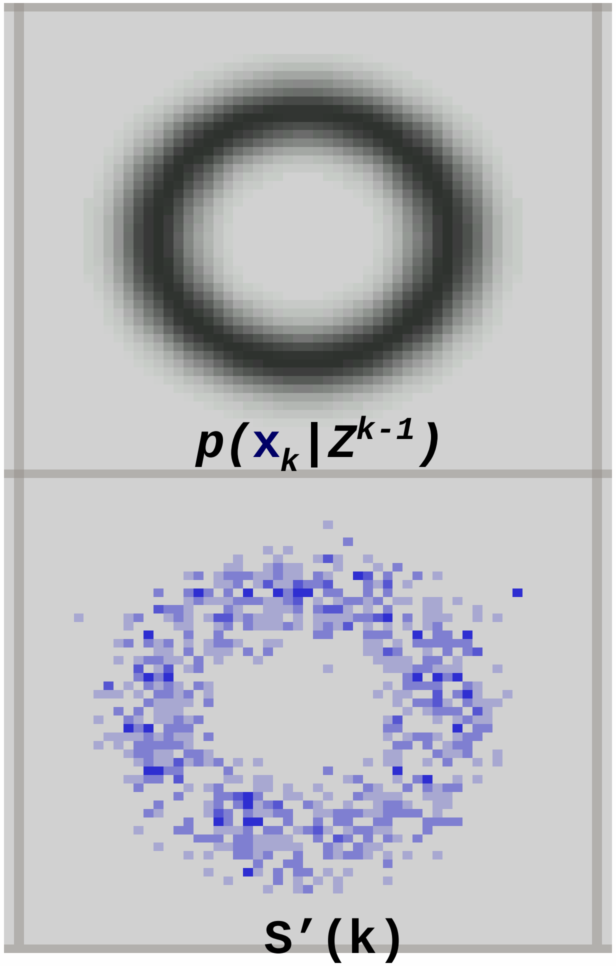
  - This resampling selects with higher probability samples s'^i_k that have a high likelihood associated with them.
  - The new set S_k approximates a random sample from $p(\mathbf{x}_k \mid Z^k)$.

Monte Carlo Localization
A Graphical Example



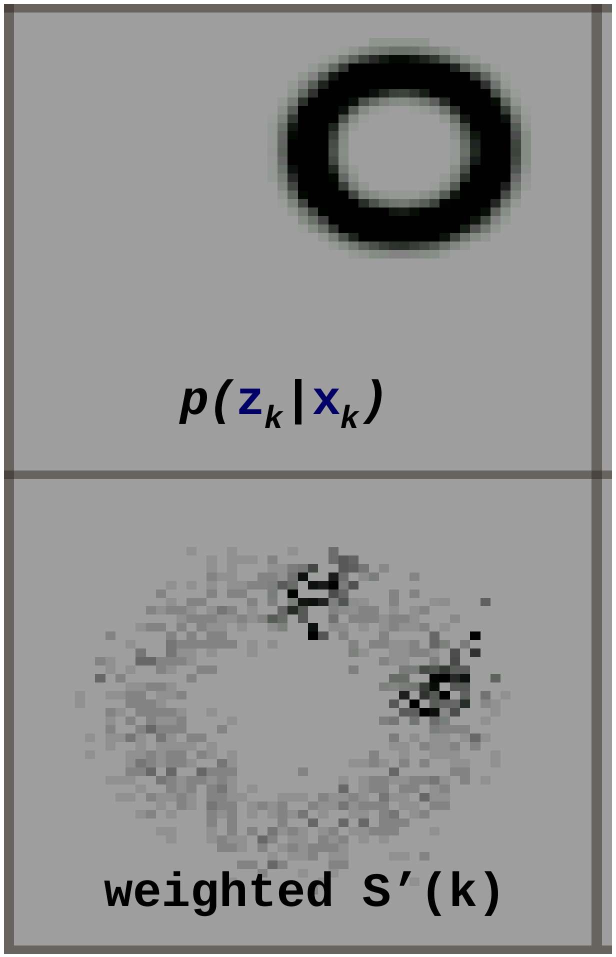
- Initially, the location of the robot is known, but the orientation is unknown.
- The cloud of particles S_{k-1} represents our uncertainty about the robot’s position.

Monte Carlo Localization
A Graphical Example



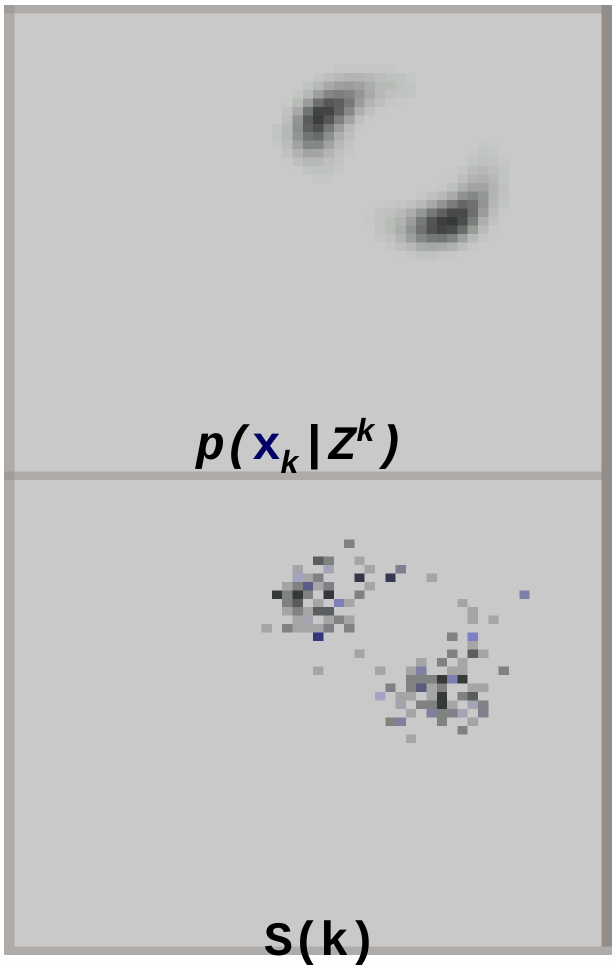
- Robot has moved 1 meter since last time-step.
- We deduce that robot is now on a circle of 1m radius around the previous location.
- Our belief state changes to reflect this.
- At this point we have applied only the motion model.

Monte Carlo Localization
A Graphical Example



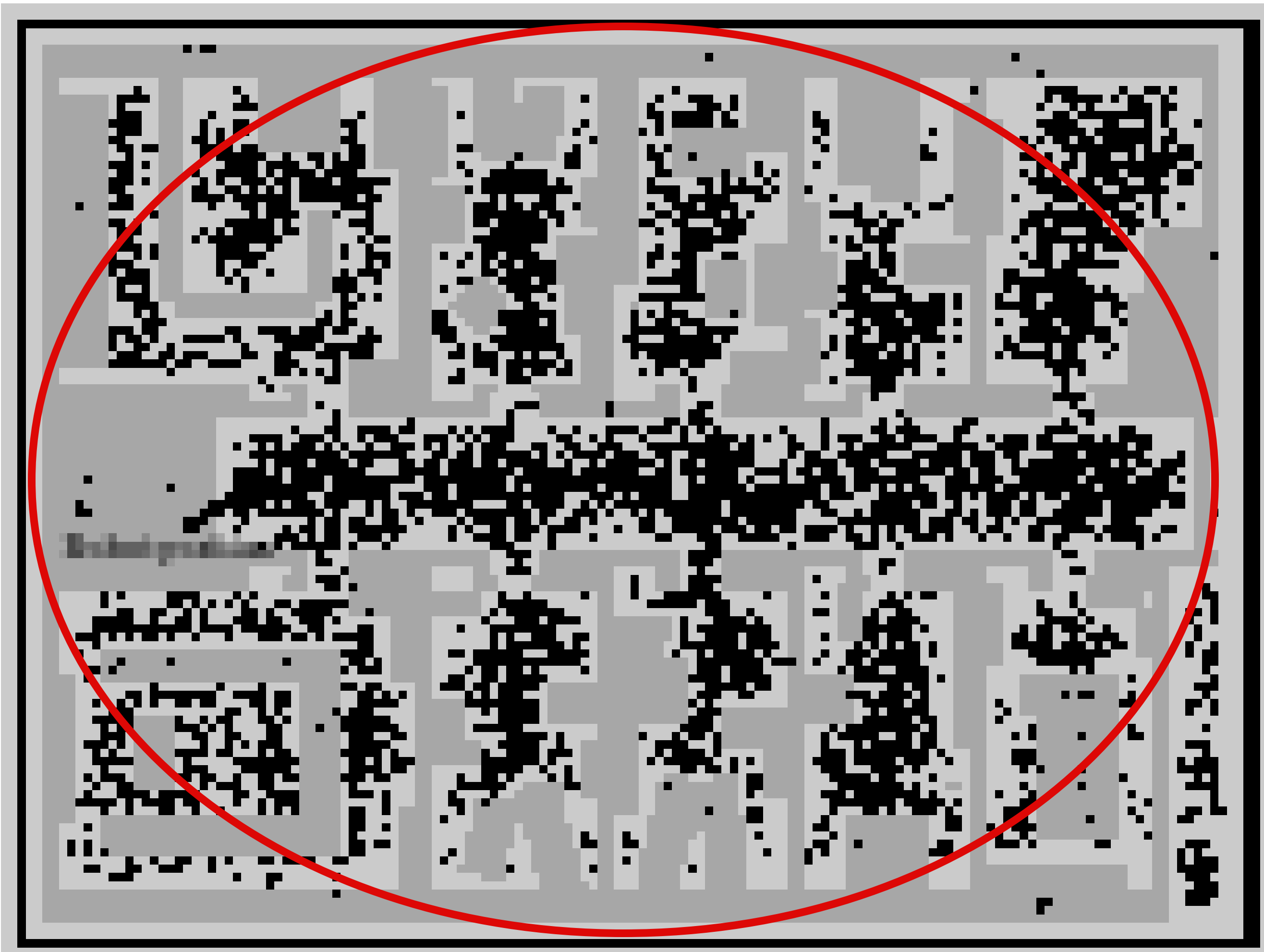
- We now take sensor readings into account.
- A landmark is observed 0.5m away somewhere in the top-right corner.
- We apply weighting to the samples to reflect that the robot is more likely to be in the top-right corner.

Monte Carlo Localization
A Graphical Example



- The weighted set is resampled to give the new set of points where the robot is most likely to be.
- This new set is the starting point for the next iteration.

Monte Carlo Localization

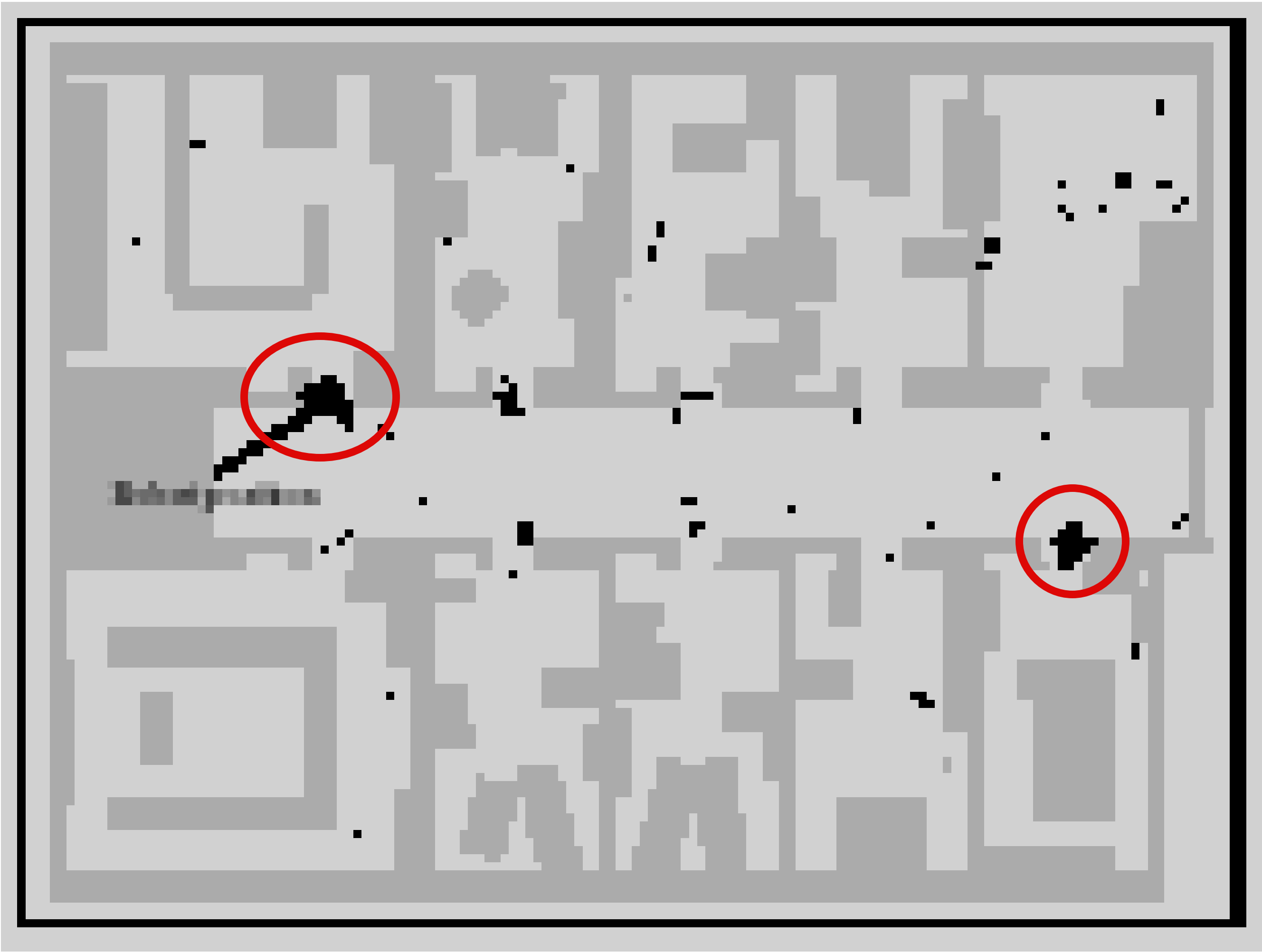


STEP 1

Global Localization

Robot does not know initial pose – every possible pose has a certain probability of being the correct location of the robot.

Monte Carlo Localization



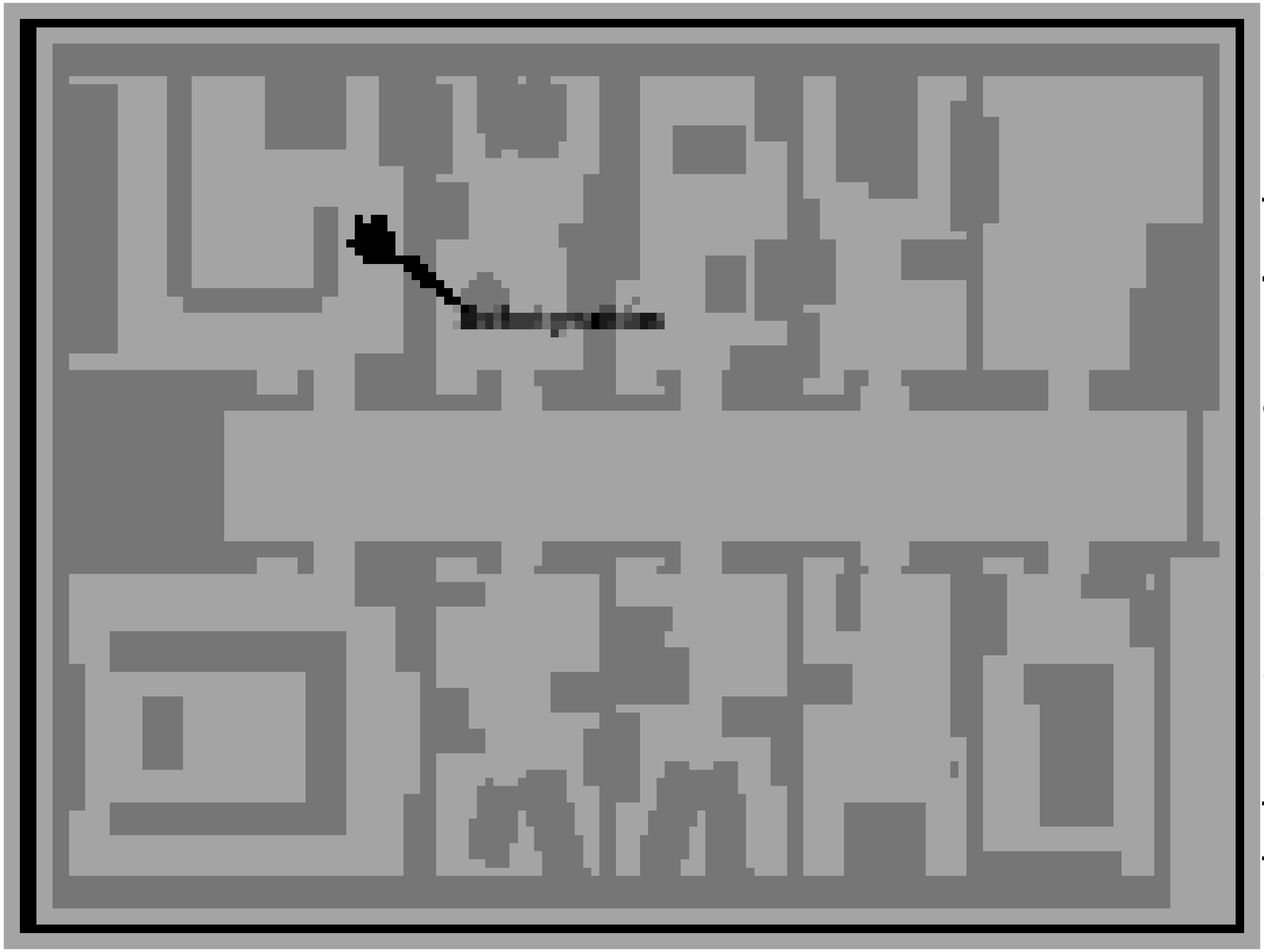
STEP 2

Global Localization

Robot observes the world (sensor readings) – the problem is reduced to choosing between two most likely poses – map has similar symmetry at both locations.

Some scattered samples survive here and there.

Monte Carlo Localization



STEP 3

Global Localization

The robot moves a little more and is able to observe (sensor readings) some unique symmetry which is not at another point on the map.

Robot is globally localized.

Monte Carlo Localization

- **Properties**
  - **Combined the advantages of grid-based Markov localization with the efficiency and accuracy of Kalman filter based techniques.**
  - **Since the MCL-method is able to represent probability densities over the robot's entire state space, it is able to deal with ambiguities and thus can *globally localize* the robot.**
  - **By concentrating the computational resources (*the samples*) on only the relevant parts of the state space, MCL-method can efficiently and accurately estimate the *position* of the robot.**
    - **Excellent in mapped environments**
    - **Need non-symmetric geometries**

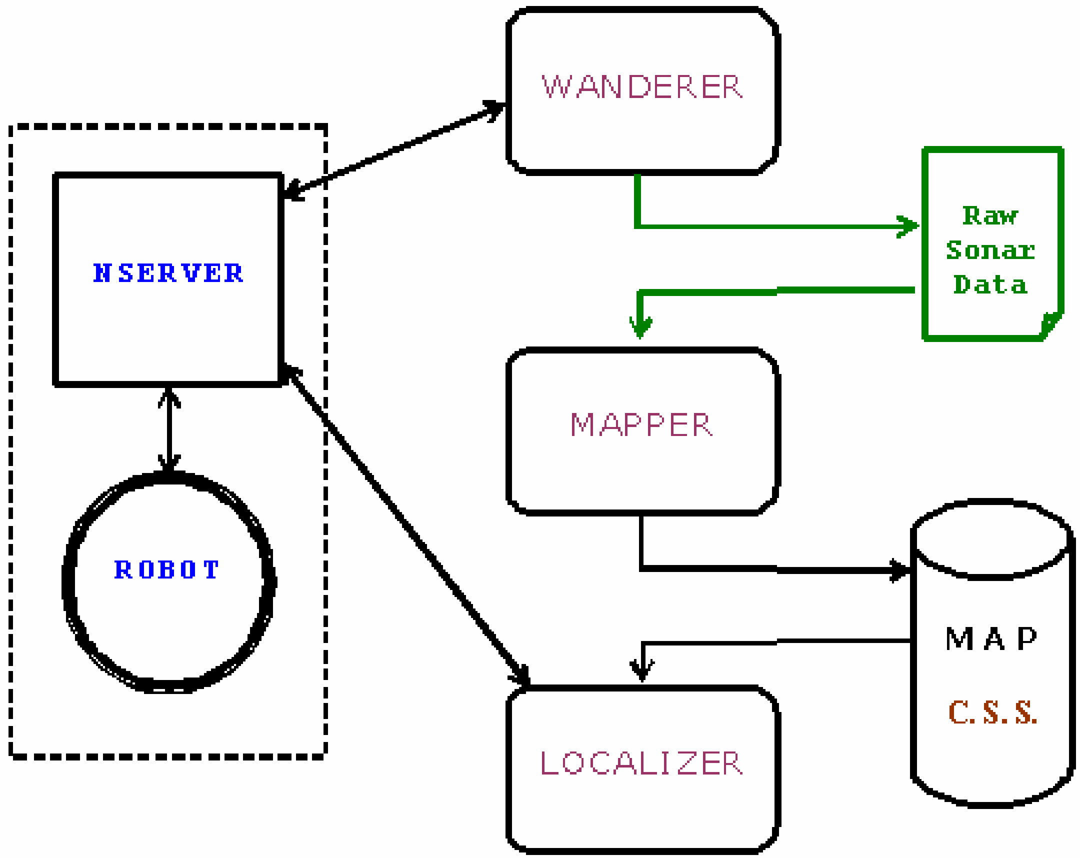
Monte Carlo Localization + DEOGs Implementation

Wanderer: explore the environment and collect information

Mapper: process the data collected by the wanderer

Localizer: use the map to pinpoint the location of the robot when requested

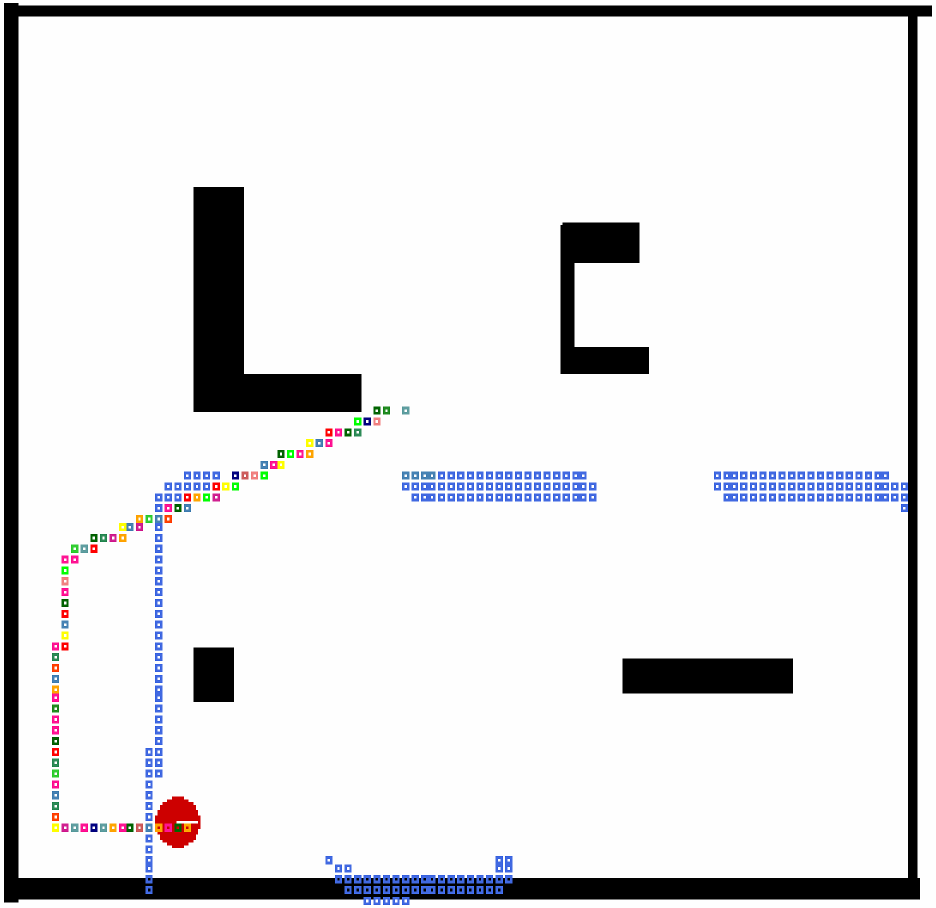
Central Storage System (CSS): stores the map, allows for expansion of the map, quick retrieval of map data



Monte Carlo Localization + DEOGs

	Mean Error	S. Dev.
X	8.59055	1.1919
Y	5.22283	3.49318
Theta	0.0	0.0

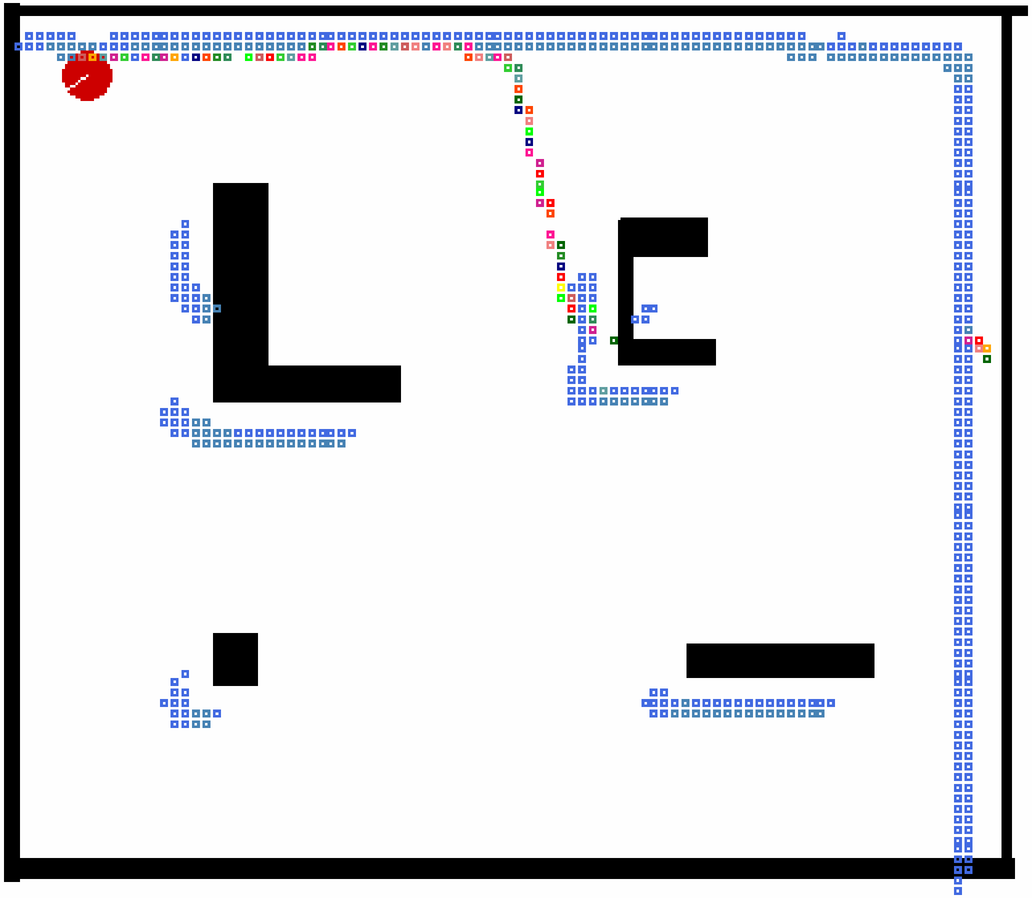
x, y are measured in inches, theta is in degrees



Monte Carlo Localization + DEOGs

	Mean Error	S. Dev.
X	3.82836	3.44103
Y	8.15522	1.01169
Theta	0.0	0.0

x, y are measured in inches, theta is in degrees

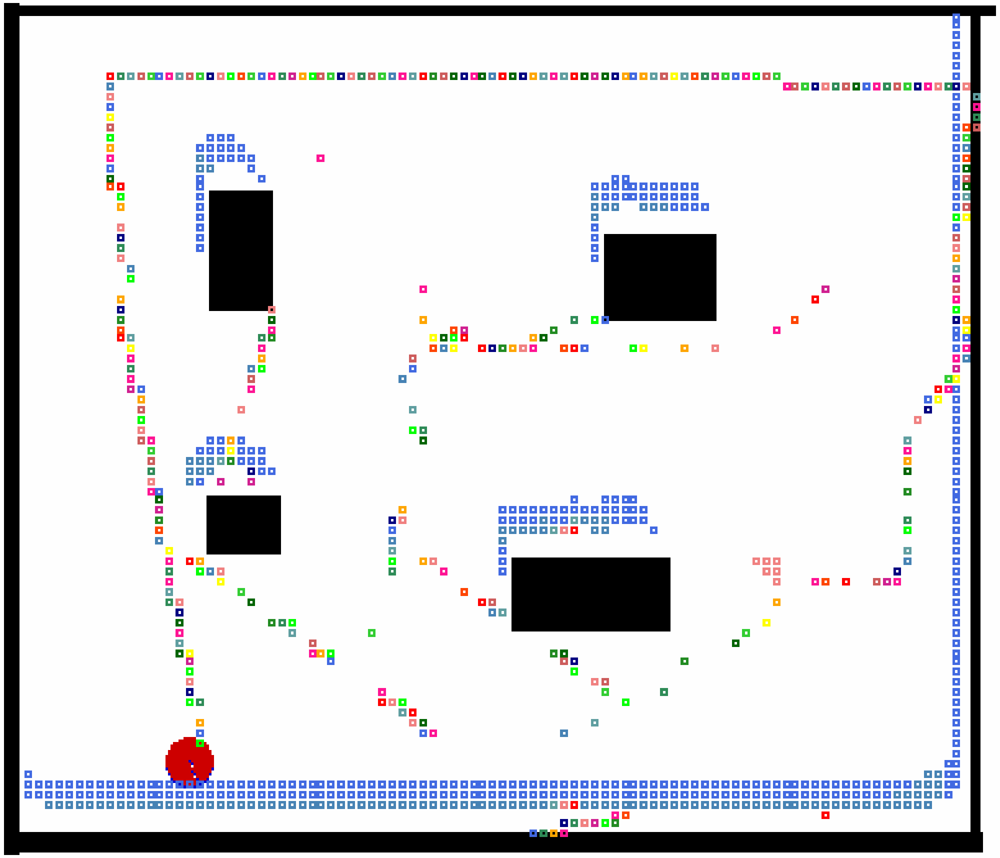


Monte Carlo Localization + DEOGs

The robot was kidnapped several times, MCL was finally able to localize onto the correct position of the robot

	Mean Error	S. Dev.
X	7.73698	1.57169
Y	3.57057	2.8353
Theta	0.0	0.0

x, y are measured in inches, theta is in degrees



Monte Carlo Localization + DEOGs

- **Conclusions:**
  - We have seen that a Monte Carlo Localization method works successfully with Dynamically Expanding Occupancy Grids. This virtually removes any limit on the environment size for nearly any robot system.
  - Now that Mapping and Localization has been tried and tested on a DEOG system, once Path-planning is also tested, a complete DEOG Robotic System can be built, that will work on an environment of any size.

References

- Monte Carlo Localization for Mobile Robots -- *F. Dellaert, D. Fox, W. Burgard, S. Thrun*
- Monte Carlo Localization: Efficient Position Estimation for Mobile Robots -- *F. Dellaert, D. Fox, W. Burgard, S. Thrun*
- Robust Monte Carlo Localization for Mobile Robots -- *F. Dellaert, D. Fox, W. Burgard, S. Thrun*
- Introduction to AI Robotics -- *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*

- CYE www.prorobotics.com
- The Honda Asimo http://asimo.honda.com
- Mars Rover http://marsrovers.jpl.nasa.gov/home/