Robotics

Introduction to Artificial Intelligence

G. Lakemeyer

Winter Term 2016/17

The Freiburg Mobile Robot Obelix

RoboCup

"By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team."

http://www.aldebaran-robotics.com/

http://youtube.com/AllemaniACs

Robots in our Lab (KBSG)

Carl

1997

AtHome

since 2006

MidSize

2007

SPL (Nao)

since 2008

Robotics

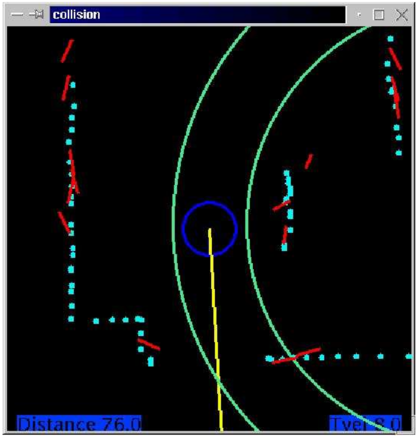
Topics: Mapping the environment
Path planning
Self-localization

The Robot Carl

- Stereo Cameras
- 24 Sonars
- Laser range finder (180°)
- Infrared sensors
- Tactile sensors
- 2 PCs
- Wireless LAN

„Seeing“ with Distance Sensors

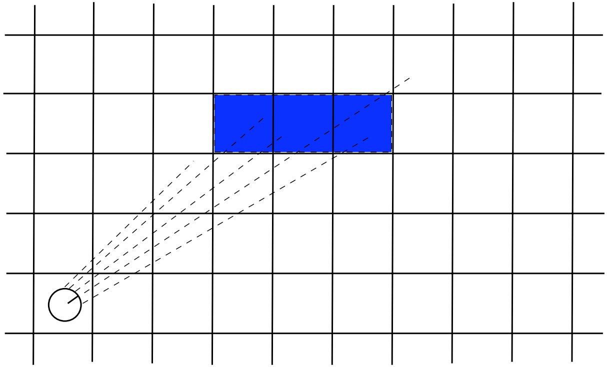
Using a camera for navigation is difficult (too much information)
Hence restrict to sonars and lasers.



Red:	Sonar
Blue:	Laser

Building Geometric Maps

Geometric maps are arrays of grids cells (15×15 cm^2), where each cell $\langle x, y\rangle$ is assigned a value $P(occ_{x,y})$ which is interpreted as the probability that that the cell is occupied. (Exact values are unrealistic because of noisy senors like sonars or lasers.)



Task: Assuming the robot's position is $\langle x_0, y_0\rangle$ and given n sensor readings, determine for any grid cell $\langle x, y\rangle$ whether the cell is occupied.

Estimating $P(occ_{x,y})$

Initialize the occupancy values of the grid to 0.5, that is, initially the robot has maximum uncertainty about the occupancy values.

Let hit mean that the sensor beam is reflected within $\langle x, y \rangle$.
Let miss mean that the sensor beam is reflected beyond $\langle x, y \rangle$.
(Ignore short readings.) Then

$$P(occ_{x,y}) = \begin{cases} \#hits/(\#hits + \#misses) & (\#hits + \#misses) > 0 \\ 1/2 & \text{otherwise.} \end{cases}$$

Estimating $P(occ_{x,y})$

Initialize the occupancy values of the grid to 0.5, that is, initially the robot has maximum uncertainty about the occupancy values.

Let hit mean that the sensor beam is reflected within $\langle x, y\rangle$.
Let miss mean that the sensor beam is reflected beyond $\langle x, y\rangle$.
(Ignore short readings.) Then

$$P(occ_{x,y}) = \begin{cases} \#hits/(\#hits + \#misses) & (\#hits + \#misses) > 0 \\ 1/2 & \text{otherwise.} \end{cases}$$

Problem:

Works only for sensors with an extremely “thin” beam, like a **laser range finder**. Does not work at all for sonars which return a single distance value taken from a 15° cone. Sonars are also extremely **noisy** (reflections, absorption, cross talk).

Sonar Interpretation I

Wanted: A mapping from (very noisy) sonar data into occupancy values of the grid cells of a geometric map.

Method: Learn the mapping using a neural net:

6 input units:
- 4 units for 4 sonars closest to $\langle x, y\rangle$
- 2 units for polar coordinates of $\langle x, y\rangle$
  (angle and distance to the first sensor)

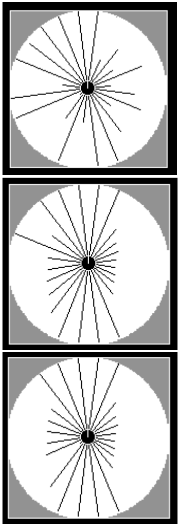
6 hidden units

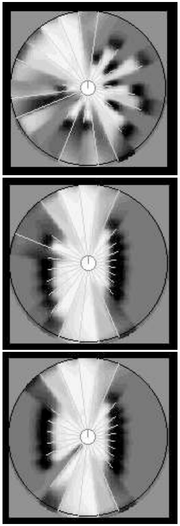
1 output unit

ideal function: Output 1 if $\langle x, y\rangle$ is occupied, 0 otherwise. The net was trained in known environments. 1 Scan is usable for many grid cells.
(It was sufficient to generate examples using a robot simulator.)

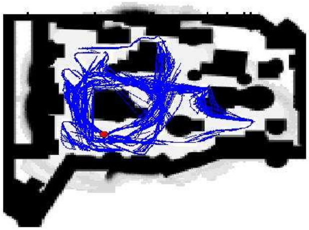
The actual output of the network is a number from $[0, 1]$, which can be interpreted as the probability of the cell being occupied by an object.

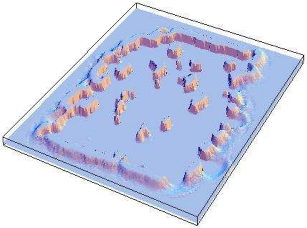
Sonar Interpretation II



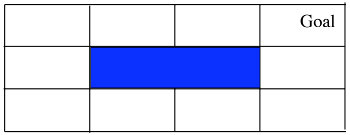


Example Deutsches Museum Bonn





Path Planning (1)



Let $C(\langle x, y\rangle)$ be the cost to traverse cell $\langle x, y\rangle$.

Wanted: For each $\langle x, y\rangle$ compute the cost $V_{x,y}$ to move from $\langle x, y\rangle$ to the goal location.

Initialization:

$$V_{x,y} = \begin{cases} 0 & \text{if } \langle x, y\rangle \text{ is the goal.} \\ \infty & \text{otherwise.} \end{cases}$$

Then iterate the following until no more changes for free locations other than the goal:

$$V_{x,y} = min\{V_{x+\delta,y+\epsilon} + C(\langle x+\delta, y+\epsilon\rangle)\} \text{ for } \delta, \epsilon \in \{-1, 0, +1\}.$$

For occupancy maps choose $C(\langle x, y\rangle) = P(occ_{x,y})$.

Method called Value Iteration (Russell and Norvig, 17.2).

Path Planning (2)

After $V_{x,y}$ is computed by value iteration, the robot can move from any point on the map to the goal by steepest descent, i.e. always move to the cell with the smallest V-value.

Advantages:

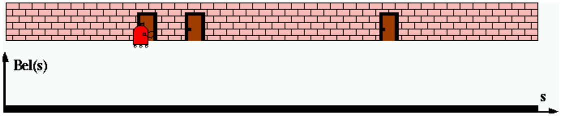
- Any time algorithm.
  The robot can start moving before value iteration has converged.
- The optimal path is calculated for every location on the map to the goal. Thus there is no need to replan if the ideal path must be abandoned (e.g. due to unforeseen obstacles).

Note: When the map changes, usually only some V-values need to be recomputed. The idea is to locally (where the map has changed) reset V-values to ∞ and restart value iteration.

How does a robot know where it is?

Probabilistic Localization

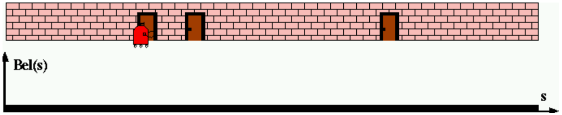
I:Initial Belief



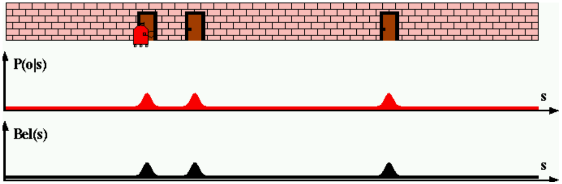
. . .

Probabilistic Localization

I:Initial Belief



II:Sensor update

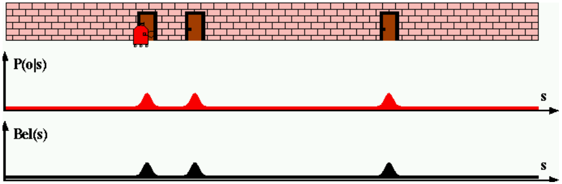


. . .

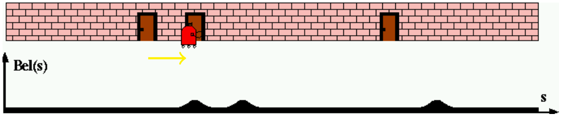
Probabilistic Localization

. . .

II:Sensor update



III:Movement

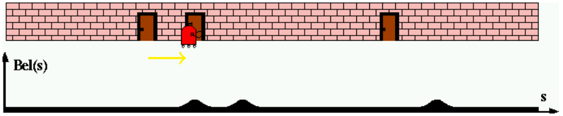


. . .

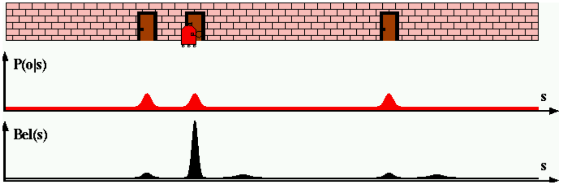
Probabilistic Localization

. . .

III:Movement



IV:Sensor Update



Self Localization - the Math

Idea: Estimate the prob. that the robot is in location $\langle x, y \rangle$ with orientation θ for a given map m.

$$P(l_{x,y,\theta}|m).$$

1. Initialize P for given m.
2. Update relative to the robot's movements and new sensor data

Initialization:

$$P(l_{x,y,\theta}|m) := \frac{1 - P(occ_{x,y})}{\sum_{x',y'}(1 - P(occ_{x',y'}))}$$

(P is very low when $\langle x, y \rangle$ is occupied, somewhat higher otherwise.)

Integrating movements

Assumption: we have information about

- t: the time since the last update
- τ: trajectory measured by wheel encoders ($\approx$ direction $+$ speed)
- Error model of the wheel encoders

Idea:

$$P(l) := \alpha \times \sum_{l'} P(l') \times P(l \mid l', \tau, t)$$

$P(l \mid l', \tau, t)$ is the prob. that the robot will be in location l when it moves with trajectory τ for t seconds (typically 0.25 sec) starting in l'.
(These values are precomputed and stored in a table.)

Sensor Integration

In analogy to map building we have:

$$P(l \mid s_T, s_{T-1}, \ldots, s_1, m) = P(l \mid s_{T-1}, \ldots, s_1, m) \times P(s_T \mid l, m)$$

[Assumption: sensor readings are independent of each other (as before)!]

Update-loop:

```
$P(l) := P(l \mid m)$ (see initialization)
for all $l, s$ do
    $P(l) := P(l) \times P(s \mid l, m)$
```

Main problem: Estimation of $P(s \mid l, m)$.

Estimation of $P(s \mid l, m)$

Let $P(r_{l'} \mid l, m)$ be the prob. that l' reflects a sensor beam when the robot is at location l in map m.

Let $P(R_i \mid l, m)$ be the prob. that at least one l' in region R_i reflects a sensor beam.

$$P(R_i \mid l, m) = 1 - \prod_{l' \in R_i} (1 - P(r_{l'} \mid l, m))$$

[Assumption: reflections of the l' are independent of each other.]

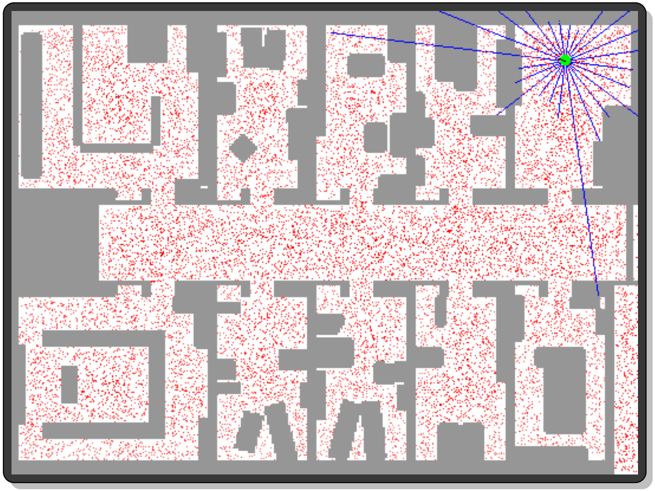
Note: if R_i is hit then the R_j $(1 \leq j < 1)$ are not hit. Let s be a reading which is inside R_i. Then

$$P(s \mid l, m) = P(R_i \mid l, m) \times \prod_{j=1}^{i-1} [1 - P(R_j \mid l, m)]$$

$\Rightarrow$ After setting the initial position (e.g. graphically by setting the robot in the map), a robot can reliably follow its positions with this method.

Monte-Carlo Lokalisierung (MCL)

[Thrun, Burgard, Fox, Dellaert]



`http://robots.stanford.edu/movies/sca80a0.avi`

`http://www.cs.cmu.edu/~mercator/global.gif`

The Markov Assumption

The world changes only as a result of the actions of the robot.

The next state of the system (robot and environment) depends only on the current state and the next action.

+ Uncertainties about the effects of actions are allowed.
– ignores exogenous effects, i.e. changes in the world which are not due to the robot's actions (e.g. visitors in a museum).

Beyond the Markov Assumption

“The world changes only as a result of the actions of the robot.”

- Slight violations of the Markov are tolerable, i.e. the system will still work ok (e.g. few dynamic objects in the environment)
- If there are many dynamic objects (e.g. visitors in a museum!) the “bad” sensor values must be filtered out.

    If the robot knows roughly where it is (with respect to its internal map), one approach would be to compare the actual sensor data with the expected ones and remove those that are wildly off.