Introduction

To achieve a reasonable degree of autonomy of robots, these have to be able to percept their operating environment and to create their own idea (internal model) about the environment. Learning a map of the working environment is thus one of the fundamental problems in mobile robotics. Maps allow robots to efficiently carry out their tasks, allow localization, efficient motion planning without collision with obstacles, and activity planning. The key questions of robotic mapping are:

- How to extract relevant information from raw sensor data?
- How to integrate gathered data over time?
- How to represent the integrated information?

Answers to these questions are found in the lecture. More precisely, we will discuss one of the mostly used approach to environment representation – occupancy grid maps introduced by Moravec and Elfes in 1985. The main idea consists in the fact that no single sensor modality alone can adequately capture all relevant features of a real environment. To overcome this problem, it is necessary to combine data from different sensors and from multiple points of view (*sensor fusion*). Occupancy grid is a two-dimensional random field that maintains stochastic estimates of the occupancy state of each cell (typically of size 3-5 cm) in a spatial lattice. The cell estimates are obtained by interpreting sensor range data using probabilistic models that capture the uncertainty in the spatial information provided by sensors. Readings taken from multiple points of view are combined by the Bayesian approach to allow the incremental updating of the occupancy grid.

Fig.1.: *An example of an occupancy grid. The lighter the cell the higher probability that it is occupied by an obstacle.*

Laser data

Although occupancy grids can be built from data produced by various sensors like sonars, laser range-finders, and cameras, we will focus only on lasers in the next text. Lasers (an acronym for *light amplification by stimulated emission of radiation*) emit a very narrow and focused laser beam. The beam is reflected by a nearest obstacle in its direction and thus returns back to the sensor, where it is detected. More precisely, phase-shift of a wave of the beam (or time-of-flight alternatively) is determined, from which a distance to the obstacle is computed. Nowadays laser range-finders do not measure a distance in one direction only, they use a rotating mirror instead, which reflects the beam into many directions. The output of such lasers is a vector of n distances $z = z^1, z^2, \ldots, z^n$, which can be interpreted (given sensor position (s_x, s_y, s_ϕ)) as a vector of points $p^i = (x^i, y^i)$ (see also Fig. 2) :

$$\begin{bmatrix} x^i \\ y^i \end{bmatrix} = \begin{bmatrix} s_x + z^i \cos(s_\phi + \phi^i) \\ y_i + z^i \sin(s_\phi + \phi^i) \end{bmatrix},$$

where ϕ^i is a direction of the i-th beam, i.e.:

$$\phi^i = \phi_{min} + i\phi_{res}.$$

ϕ_{res} is angular resolution of the scanner:

$$\phi_{res} = \frac{\phi_{max} - \phi_{min}}{n}$$

and ϕ_{min} ϕ_{max} are minimal and maximal directions of a laser beam.

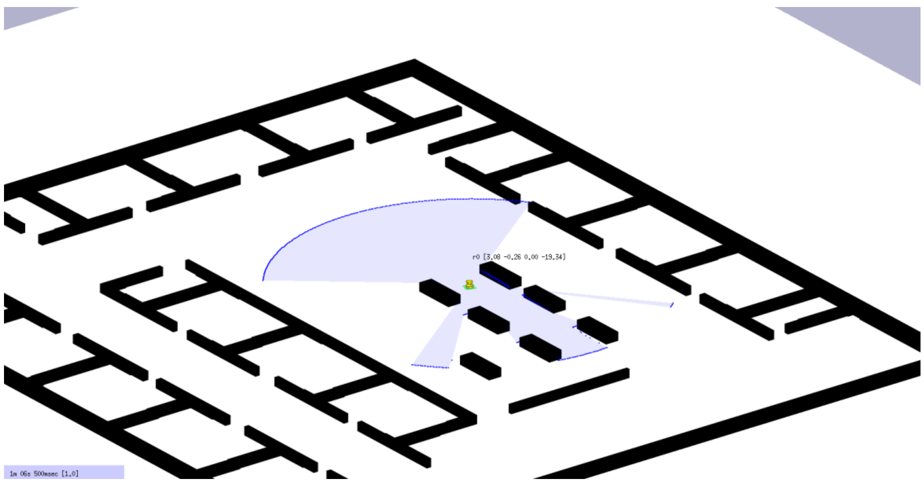

Fig.2.: *Laser scan.*

Occupancy grid update

As mentioned above, each cell of the grid holds information about occupancy of the corresponding space in the form of probability. The question is, how to determine these probabilities given a set of laser measurements (each as a vector or distances), i.e. what is our believe of a state of the map m_t in time t:

$$Bel(m_t) = p(m_t|z^1, z^2 \ldots, z^t)$$

Computing this equation is computationally intractable, therefore we assume that individual cells $m[xy]$ are independent (although this is not true in general):

$$Bel(m_t) = \prod_{x,y} Bel(m_t^{[xy]})$$

In other words, we can update each cell individually. To do this Bayes rule is employed:

$$Bel(m_t^{[xy]}) = \eta p(z_t|m_t^{[xy]})Bel(m_{t-1}^{[xy]}),$$

where η is a normalization constant ensuring that probabilities of all possible states of $m_t^{[xy]}$ sum to 1, $p(z_t|m_t^{[xy]})$ is a sensor model, which will be explained in the next section, and $Bel(m_{t-1}^{[xy]})$ is the current believe in the state of $m_{t-1}^{[xy]}$ determined in the previous step. We have typically no apriory information about the environment, therefore $Bel(m_0^{[xy]})$ is set to $\frac{1}{2}$ for all cells.

A practical approach

Note that probability has values from an interval $[0, 1]$. Moreover, it goes fast to its limiting values when Bayes rule is applied. To overcome this, pure probabilities are not store in individual grid cells, so call *log odds* are used instead. Odd of probability $p(A)$ is defined as a ratio of it and its complement:

$$odds(A) = \frac{p(A)}{p(\neg A)} = \frac{p(A)}{1 - p(A)}$$

and has a range $[0, +\infty)$ and a range of a logarithm of $odd(A)$ is $(-\infty, +\infty)$.

If Bayes rule is applied on $p(A|B)$, we get

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

and likewisely

$$p(\neg A|B) = \frac{p(B|\neg A)p(\neg A)}{p(B)}$$

So, we can derive

$$odds(A|B) = \frac{p(A|B)}{p(\neg A|B)} = \frac{p(B|A)p(A)}{p(B|\neg A)p(\neg A)} = \lambda(B|A)o(A),$$

where

$$\lambda(B|A) = \frac{p(B|A)}{p(B|\neg A)}$$

. Finally, after application of logarithm, we get addition instead of multiplication:

$$\log odds(A|B) = \log \lambda(B|A) + \log odds(A)$$

Sensor model for laser data

A sensor model aims to determine $p(z_t|m_t)$, i.e. probability of receiving a sensor measurement z_t given a map m_t and assuming that robot position is known. Keep on mind that if a sensor measures an obstacle in some direction at a distance r, not only a cell corresponding this distance is influenced, but also all cells closer to the sensor than r should be updated (and also few with a distance higher that r). Intuitively, if an obstacle is measured at a distance 10 m in front of a robot, then it is highly probable that there is no obstacle at a distance 1 m in this direction. But, may be, an obstacle is with some probability (assuming sensor noise) at 10.1 or (with lower probability) at 10.2. Therefore, given a single sensor measurement in some direction, the model should specify, how cells in this direction are updated (we relax angular uncertainty).

Various models can be specified, one of these computes a probability density function of z_t as an average of two functions, the first one (model_{O}) corresponds to the probability of occupancy, while the second one is a complement of "emptiness" model_{V}:

$$p(z_t|m_t^{[xy]}) = \frac{1 + model_O^{z_t}(r) - model_V^{z_t}(r)}{2},$$

where (α, r) are polar coordinates of the cell $m_t^{[xy]}$ in sensor coordinate system and z_t is measured distance.

$$\begin{aligned} model_v^r(\delta) &= \begin{cases} 1 - \left(\frac{\delta}{r-\epsilon}\right)^2, & \text{for } \delta \in < 0, r - \epsilon > \\ 0 & \text{otherwise} \end{cases} \\ model_o^r(\delta) &= \begin{cases} 1 - \left(\frac{\delta - r}{\epsilon}\right)^2, & \text{for } r < X \wedge \delta \in < r - \epsilon, r + \epsilon > \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

The models are shown in Fig. 3. You can see the blue function representing $model_v^r$ is highest in the vicinity of the robot and then decreases slowly. This corresponds to the fact, that measuring an obstacle at some distance gives us high probability that there is no obstacle near the robot and this probability decreases with distance.

The red function, in the opposite, expresses probability of the obstacle. This is highest, as expected, at the distance the sensor measured an obstacle (i.e. at r) and it quadratically decreases with distance. Moreover, the function has non zero values in the small area around r specified by a constant ϵ

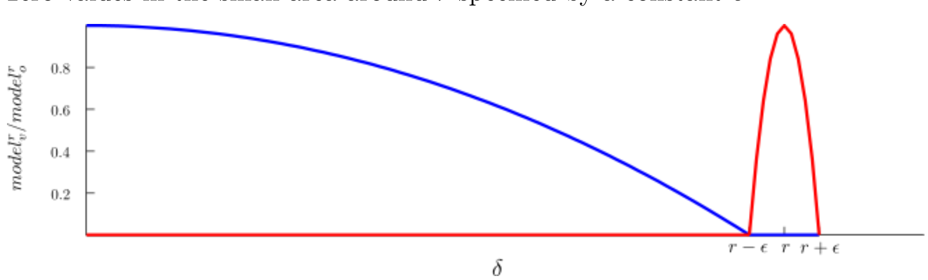

Fig.3.: *A laser range-finder model.*

A practical approach

Contemporary laser range-finders are really accurate, their precision is typically in order of centimetres, which is lower than or equal to the size of a grid cell. Therefore, a simpler sensor model can be used: the grid cell C corresponding to the sensor measurement is set to 1 (note that this still should be fused with the old value of the cell using Bayes rule), while cells lying of an abscissa CS, where S is a position of the robot are set to 0, see Fig. 4left). For consecutive filling of individual cells, Bresenham's algorithm as presented in the previous lesson can be successfully employed.

Another (and more effective) alternative is to update cells influenced by a whole scan (i.e. scan beams from all directions) at once. We can imagine that cells corresponding individual measurements in the scan form a border of some region. While a border of this region can is filled with ones, its interior is filled with zeros, see Fig. 4right). Again, we can employ an algorithm from computer graphics to fill the interior. Polygon filling, e.g. flood fill [1] or line scan rendering [2] is the appropriate option in this case,

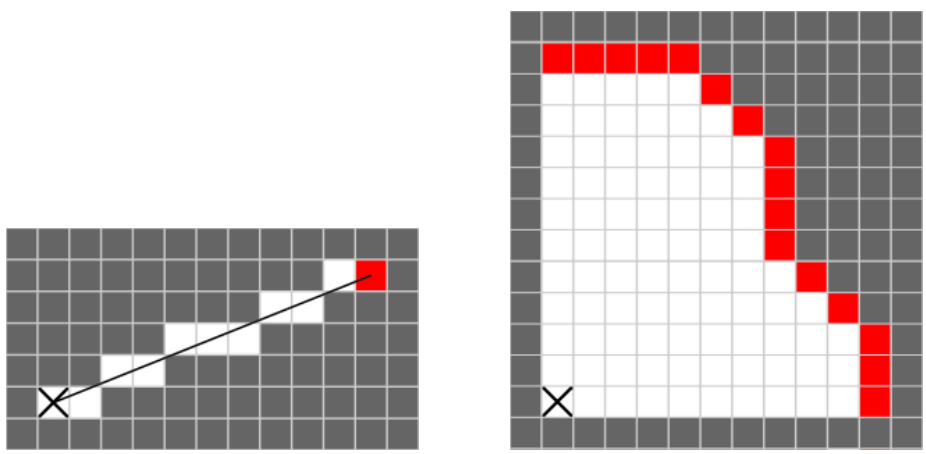

Fig.4.: *A simplified laser range-finder model.*

References

[1] Flood fill https://en.wikipedia.org/wiki/Flood_fill [2] Line scan rendering https://www.cs.uic.edu/~jbell/CourseNotes/ComputerGraphics/PolygonFilling.html