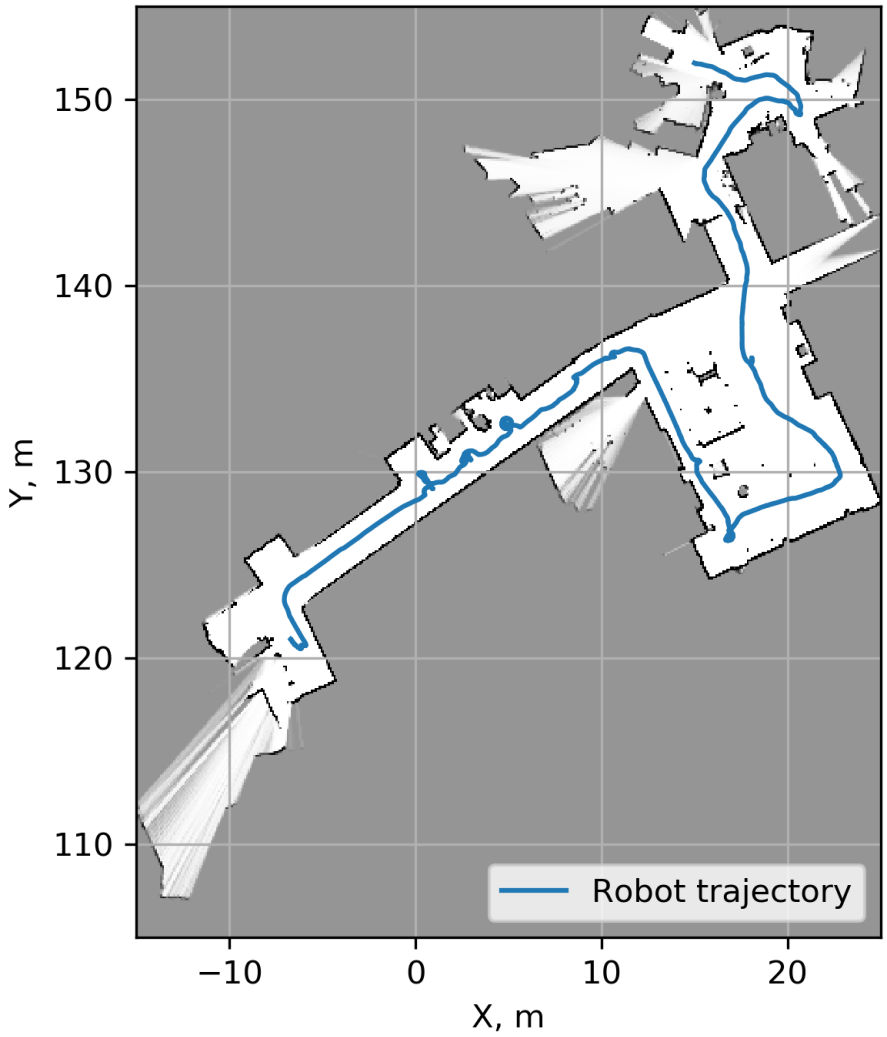
Occupancy Grid Mapping

Artyom Pavlov, 2019

Mapping problem

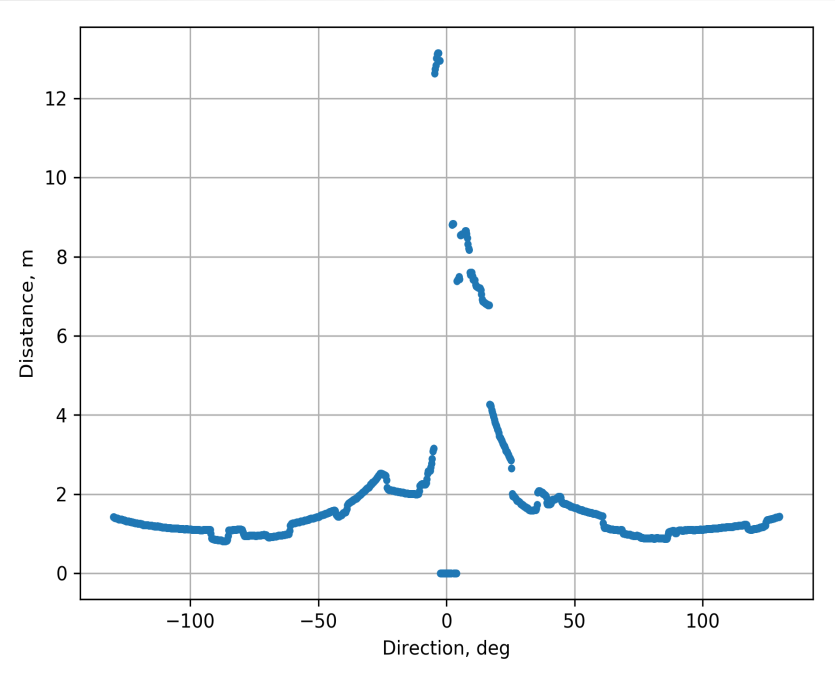
- By having sensor readings (e.g. lidar, sonar) and known robot position we want to determine state of the environment in which robot operates
- We assume environment is static
- One of the most common approaches to this problem called “occupancy grid”
- It’s a probabilistic approach in which we model world as a discrete map. Each pixel of the map is independent and can either be empty or occupied.
- Because we don’t have exact information we use *probabilities*. 1 denotes occupied pixel, 0 an empty one, and 0.5 means it can be either one (e.g. because we don’t have any information to judge the pixel state)

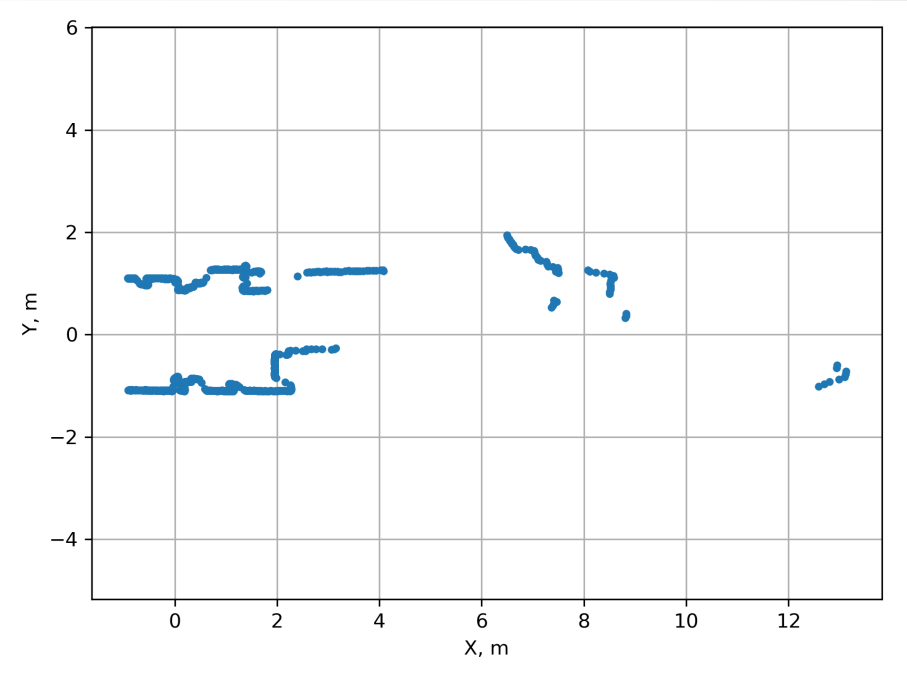
Occupancy grid example



Sensor model

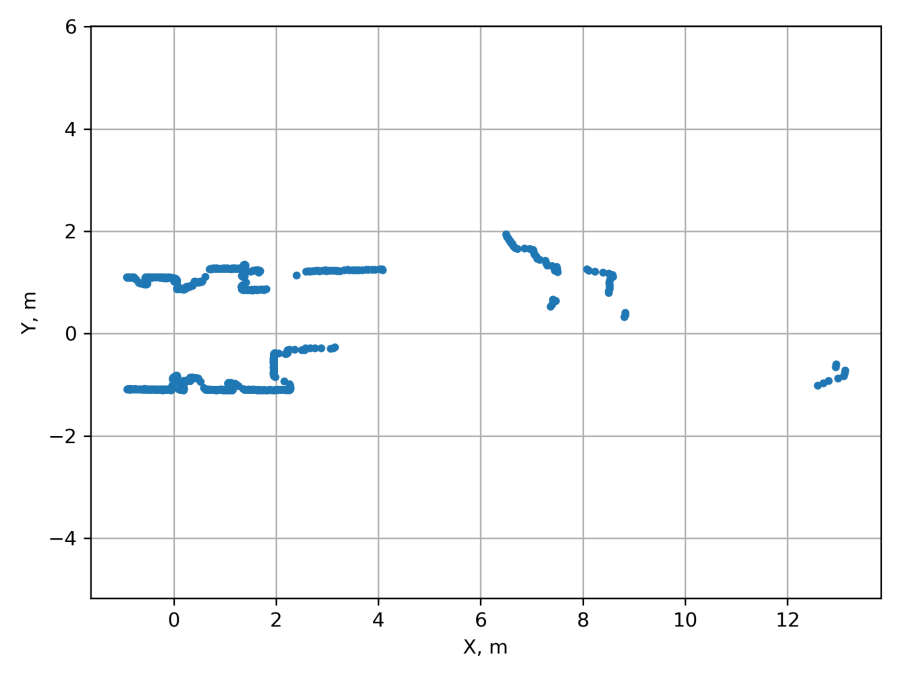
- Our robot equipped with LiDAR which produces 1040 measurements in 260 degrees Field of View (FOV)
- First we need to convert sensor readings from polar coordinates to Cartesian

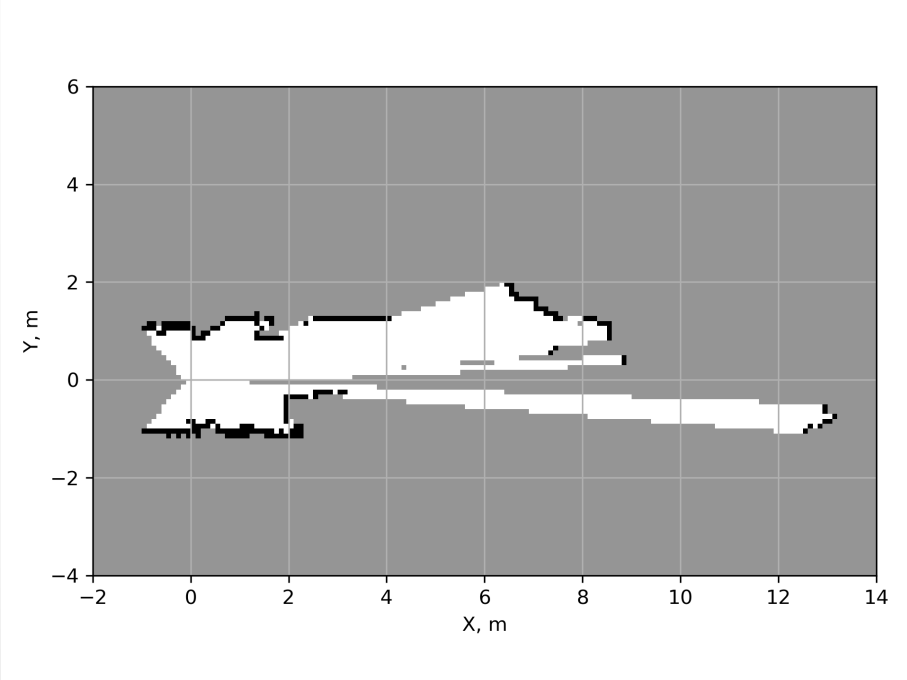




Sensor model

- We assume that space between robot and measured point is empty and space which corresponds to the measured point is occupied



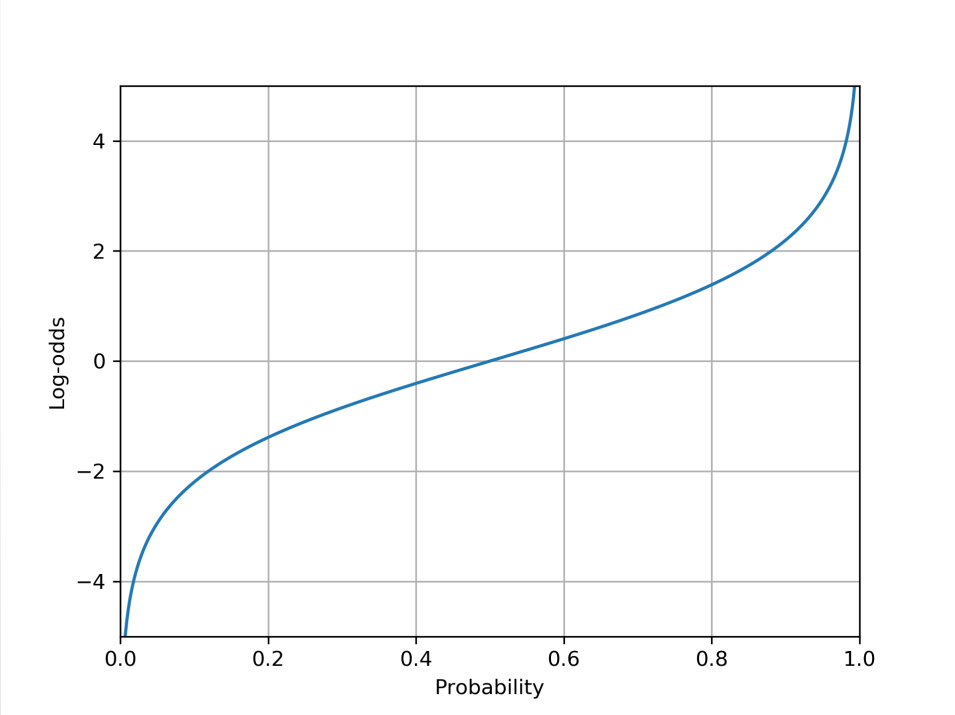


Accumulation

- We need to accumulate a sequence of potentially inconsistent measurements into a single map
- We can’t fully trust each measurement, thus if the current measurement indicates that cell is occupied we interpret it as “with probability 0.5 + p it’s occupied”, and if it’s empty as “with probability 0.5 - p it’s occupied”
- Here p is factor of how much we trust a single measurement
- By applying Bayes’ theorem we can calculate the final probability values for each cell of the map
- The easiest way to do it is to use log-odds representation

Log-odds representation

- $l = \log(p/(1-p))$
- $p = 1/(1+\exp(-l))$
- $l_{i+1} = \log(p_i/(1-p_i)) + l_i$, here p_i – is probability for the cell given by i-th measurement



Task

- Using given scans and poses build plot with the occupancy grid and robot trajectory (see slide 3)
- scans.npy contains 5000x1040 2D array of lidar measurments
- poses.npy contains 5000x3 2D array of robot poses in the form (x, y, phi)
- tools.py contains 3 helper functions
- Process negative coordinates correctly!
- Additional task: generate a video with the animation which will demonstrate how occupancy grid changes when robot moves

Task: algorithm

- Create grid filled with zeros
- For each measurement and pose:
  - Convert points to Cartesian coordinates in the global reference frame
  - Create probabilities occupancy grid for the measurement
  - Convert probabilities to log-odds representation and add them to the grid
- Convert grid from log-odds representation to probabilities
- Plot occupancy grid and robot trajectory

Python, numpy, and pyplot tips

- Using broadcasting you can shift points in the array as: `points + np.array([2, 3])`
- You can apply numpy operations to whole array:

  `np.log(arr)` or `np.exp(arr)`
- `arr1 + arr2` will perform element-wise addition of array elements if array shapes are equal, same goes for other arithmetic operations
- To create an array filled with zeros use `np.zeros(shape)`
- To plot 2D array as an image you can use one of the following functions: `plt.imshow(..)`, `plt.matshow(..)` or `plt.spy(..)`

Python, numpy, and pyplot tips

- To change color pallet of the plotted image use `cmap` argument of the plotting function.
- By default image plotting function will use pixel number for axes, to change this behaviour use `extent` argument of the plotting function
- To save figure use `plt.savefig(filename)`. To change DPI of the saved image use `dpi` argument.
- To transpose array you can write `arr.T`
- To reverse order of array or slice you can write `arr[::-1]`
- To generate video you can either use `matlpotlib.animation` module or generate and save each frame as a separate image and convert then into video using ffmpeg