Fusion information

Autonomous Agent

Lorenzo Vannini

Università degli studi di Firenze

June 29, 2023

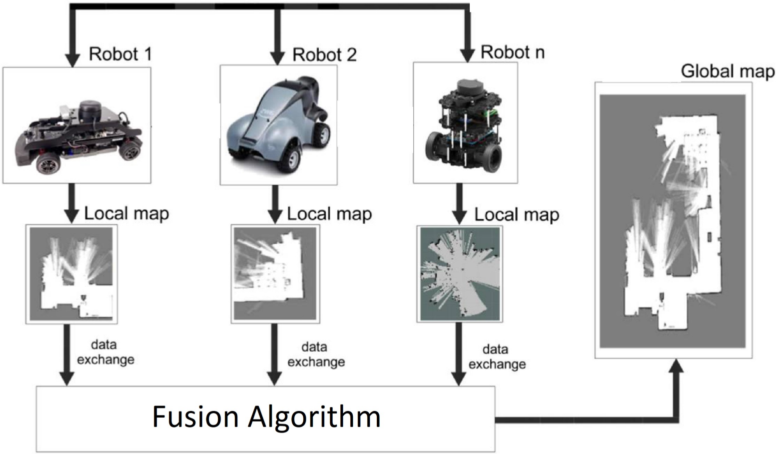
Overview

1. Introduction
   - Mathematical Framework
   - Information fusion
2. Test Results

Introduction

This project concentrates on the study on the map fusion problem in the context of a multi-robot map building approach.
Each robot builds its own local map using its observations.
As a result, there will be a set of local maps that can be **fused** into a global one.

Introduction



Introduction

Suppose that each robot is equipped with a sensing device having a limited field of view depending on the robot position.
Suppose that we want to sense/monitor a given surveillance area $\mathcal{X}$

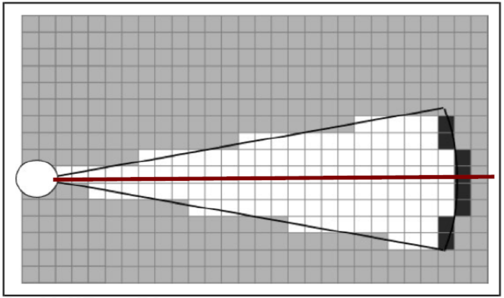


Figure 2: Robot Scanning an environment

Occupancy grid

A widely adopted environment representation is the occupancy grid map, where the vehicle's surrounding is divided into cells.
Every cell is associated with a physical location and contains the occupancy probability, i.e. the probability to be occupied by an object.
Given a cell in the occupancy grid

$$\theta_k \qquad \forall k = 1, \ldots, M$$

The cell occupancy are modelled as independent binary random variables and for **every** cell and for **every** robot we have a probability

$$P_i(\theta_k) \in [0, 1] \qquad \forall i = 1, \ldots, N$$

that the cell is occupied given the observations collected during its trajectories

Occupancy grid

Given the probabilities $P_i(\theta_k)$ and a threshold $\varepsilon \in [0, 0.5]$, each cell is classified by the robot i as follows:

$$P_i(\theta_k) = \begin{cases} \text{free if } \; P_i(\theta_k) < \varepsilon \\ \text{occupied if } \; P_i(\theta_k) > 1 - \varepsilon \\ \text{uncertain otherwise} \end{cases}$$

If no prior information is available on cell θ_k, we set

$$P_i(\theta_k) = \frac{1}{2}$$

Kullback-Leibler average

The (weighted) Kullback-Leibler average (KLA) of the densities $p_1(x), \cdots, p_N(x)$ is defined as

$$\bar{p} = \arg\min_p \sum_{i=1}^{N} \pi_i \int p(x) \log \frac{p(x)}{p_i(x)} dx$$

can be re-written as

$$\bar{p} = \frac{\prod_{i=1}^{N} p_i(x)^{\pi_i}}{\int \prod_{i=1}^{N} p_i(x)^{\pi_i} dx}$$

Information fusion

In a discrete environment, by setting $\pi_i = \frac{1}{N}\ \forall i$ the previous equation becomes

$$\bar{p}(\theta_i) = \frac{\prod_{j=1}^{N} p_j(\theta_i)^{\pi_j}}{\prod_{j=1}^{N} p_j(\theta_i)^{\pi_i} + \prod_{j=1}^{N}(1 - p_j(\theta_i))^{\pi_j}}$$

Overview

1. Introduction
2. Test Results
    - Initial simple experiments
    - More Realistic Experiments
    - Travelled distance algorithm
    - Local efficiency second algorithm

Initial simple experiments

By creating a differential-drive kinematic motion model, a pure pursuit controller and a sensor model from a given path

$$\mathcal{P} = \{(x_1, y_1), \cdots, (x_K, y_K)\}$$

we can obtain the following results

Initial simple experiments

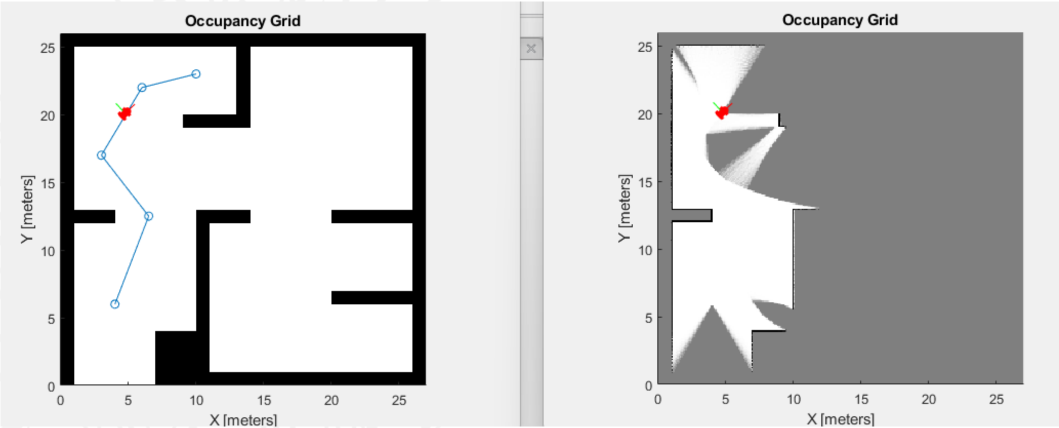


Figure 3: First Robot scanning the environment

Initial simple experiments

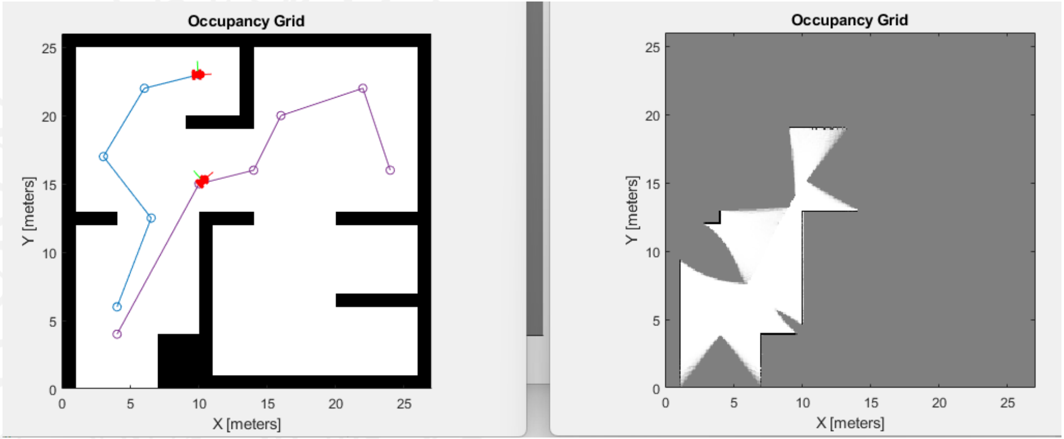


Figure 4: Second Robot scanning the environment

Initial simple experiments

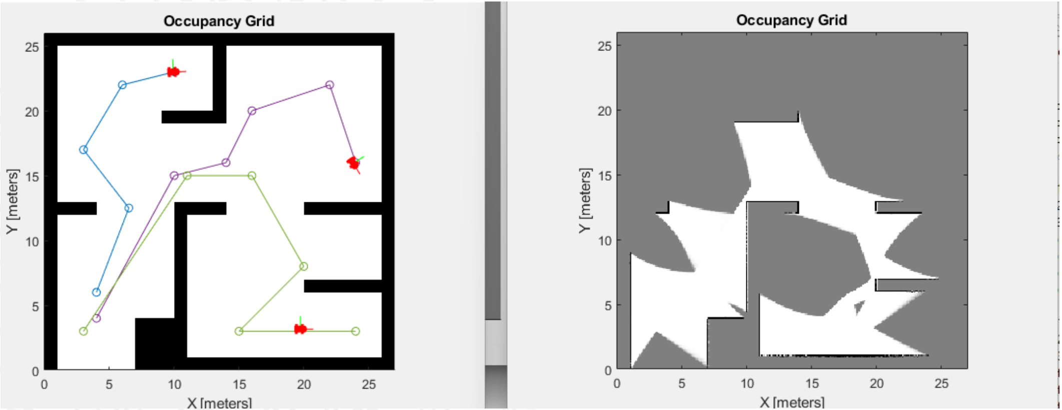


Figure 5: Third Robot scanning the environment

Initial simple experiments

Figure 6: Final Map with range $r = 6$

Initial simple experiments

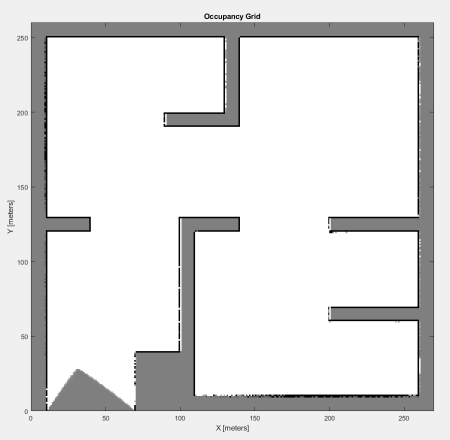


Figure 7: Final Map with range $r = 9$

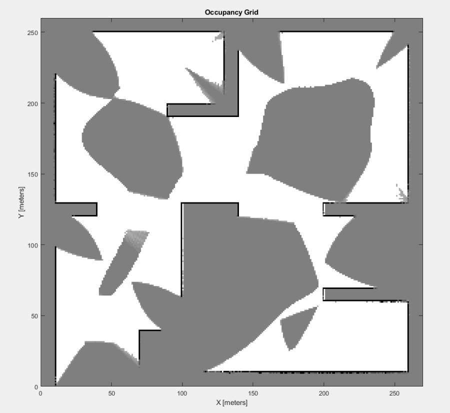


Figure 8: Final Map with range $r = 4$

More Realistic Experiments

In order to represent as good as possible reality we might add a Gaussian noise to the sensor model

$$\xi \sim \mathcal{N}(0, \sigma^2)$$

or a uniform noise

$$\xi \sim \sigma^2 \mathcal{U}(0, 1) \;\; \sigma \in \mathbb{R}$$

More Realistic Experiments

At this point we might also add a coefficient $\eta_i \in [0, 1]$ representing the efficiency of the i_{th} robot.
It is fair to state that the longer the path that the robot travels the more likely is for the measurements to be corrupted by external factor.
Given a path

$$\mathcal{P}_i = \{(x_1, y_1), \cdots, (x_K, y_K)\}$$

we can calculate the distance travelled by the robot i with

$$d_i = \sum_{r=1}^{K-1} \sqrt{(x_{r+1} - x_r)^2 + (y_{r+1} - y_r)^2}$$

More Realistic Experiments

With the previous equation

$$d_i = \sum_{r=1}^{K-1} \sqrt{(x_{r+1} - x_r)^2 + (y_{r+1} - y_r)^2}$$

we can estimate the efficiency using a decreasing function such that $\eta(d = 0) = 1$ and $\eta(\infty) = 0$

$$\eta(d) = e^{-\alpha d} \quad \alpha \in \mathrm{R}^+$$

in our case $\alpha = 0.01$

More Realistic Experiments

The information fusion algorithm becomes

$$\bar{p}(\theta_i) = \frac{\prod_{j=1}^{N}[\frac{1}{2} + (p_j(\theta_i) - \frac{1}{2})\eta_j]^{\pi_j}}{\prod_{j=1}^{N}[\frac{1}{2} + (p_j(\theta_i) - \frac{1}{2})\eta_j]^{\pi_j} + \prod_{j=1}^{N}(1 - [\frac{1}{2} + (p_j(\theta_i) - \frac{1}{2})\eta_j])^{\pi_j}}$$

More Realistic Experiments

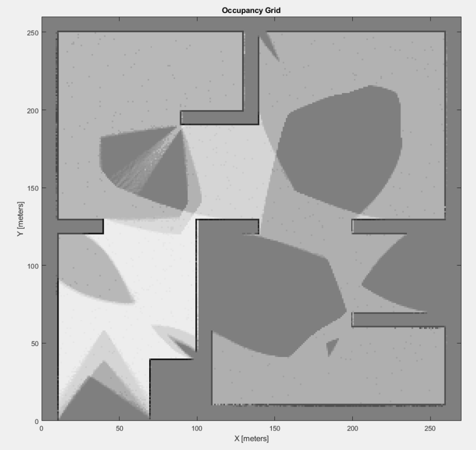


Figure 9: Final Map with η and $\xi \sim 0.1\,\mathcal{U}(0,1)$

More Realistic Experiments

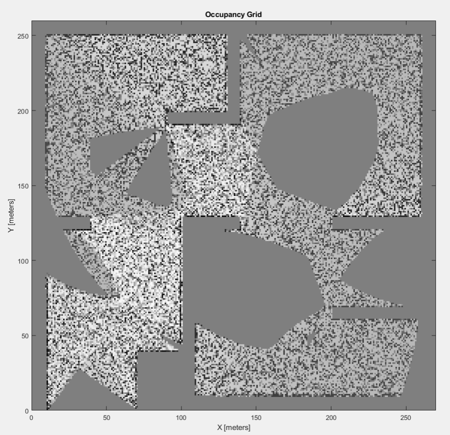


Figure 10: Final Map with η and $\xi \sim \mathcal{N}(0, 0.9)$.

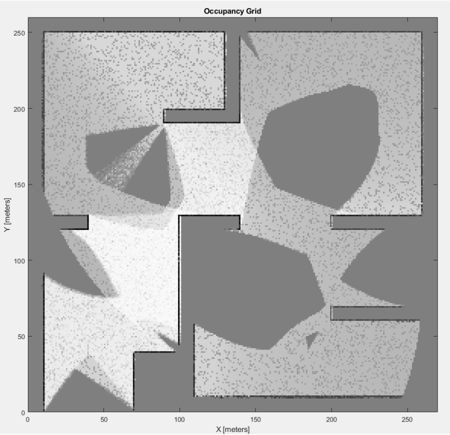


Figure 11: Final Map with η and $\xi \sim \mathcal{N}(0, 0.1)$.

Travelled distance algorithm

Main idea, we should also consider the distance travelled as a measure of uncertainty.
For every point $\mathbf{p}_i = [x_i, y_i] \in \mathbb{R}^2 \quad \forall i$ we can measure the **distance travelled form the origin of the i-th path and create a local efficiency** $\eta^r_{ij} \, \forall r = 1, \cdots, N$

Travelled distance algorithm

Algorithm 1 Travelled Distance algorithm

Require: $\mathcal{P}_i = \{(x_1^{(i)}, y_1^{(i)}), \cdots, (x_K^{(i)}, y_K^{(i)})\}\ \forall i = 1, \cdots, N,$
$2D$ $(m_1 \times m_2)$ map
a distance function d
$\beta \geq 0$
for $i = 1, \cdots, m_1$ **do**
  for $j = 1, \cdots, m_2$ **do**
    $\mathbf{p} = [i, j]$ ▷ Define the point in 2D space
    for $r = 1, \cdots, N$ **do**
      $\mu_{ij}^r = d(\mathbf{p}, (x_1^{(r)}, y_1^{(r)}))$ ▷ Define the distance from $\mathbf{p}$ to the start
      $\eta_{ij}^r = e^{-\beta \mu_{ij}^r}$

$$\bar{p}(\theta_{ij}) = \frac{\prod_{r=1}^{N}[\frac{1}{2} + (p_r(\theta_{ij}) - \frac{1}{2})\eta_{ij}^r]^{\pi_r}}{\prod_{r=1}^{N}[\frac{1}{2} + (p_r(\theta_{ij}) - \frac{1}{2})\eta_{ij}^r]^{\pi_r} + \prod_{r=1}^{N}(1 - [\frac{1}{2} + (p_r(\theta) - \frac{1}{2})\eta_{ij}^r])^{\pi_r}}$$

Figure 12: First Algorithm

Travelled distance algorithm

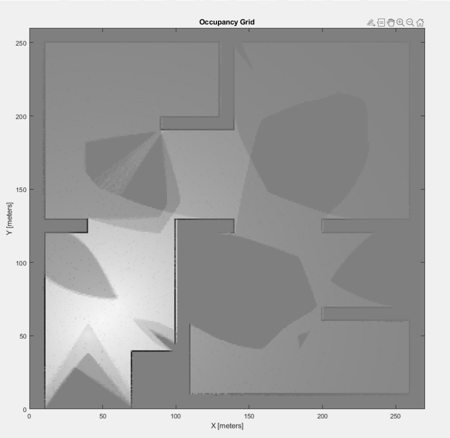


Figure 13: Final Map with $\beta = 0.01$

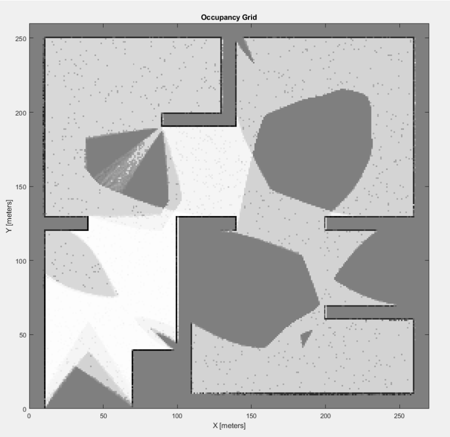


Figure 14: Final Map with $\beta = 0.0001$.

Local efficiency second algorithm

Main idea, we should also consider the distance **from fixed centers** as a measure of confidence.
Let's define $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_N\}$ where $\mathbf{c}_i \in \mathbb{R}^2$ and N is the number of robots in the system.
For every point $\mathbf{p}_i = [x_i, y_i] \in \mathbb{R}^2 \quad \forall i$ we can measure the **closest center** to a generic point and create a **local efficiency** $\eta^r_{ij}\ \forall r = 1, \cdots, N$.
Let's also define a $d_{\max} > 0$ the maximum sensing distance.

Local efficiency second algorithm

Require: $\mathcal{P}_i = \{(x_1^{(i)}, y_1^{(i)}), \cdots, (x_K^{(i)}, y_K^{(i)})\}\ \forall i = 1, \cdots, N,$
$2D$ $(m_1 \times m_2)$ map
$\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_N\}$ where $\mathbf{c}_i \in$ map
$d_{\max} \geq 0$
a distance function d

for $i = 1, \cdots, m_1$ **do**
  for $j = 1, \cdots, m_2$ **do**
    $\mathbf{p} = [i, j]$ ▷ Define the point in 2D space
    find $k^* \in \{1, \cdots, N\} \mid d(\mathbf{c}_{k^*}, \mathbf{p}) \leq d(\mathbf{c}_l, \mathbf{p})\ \forall k^* \neq l$ ▷ closest center
    $d^* = d(\mathbf{c}_{k^*}, \mathbf{p})$
    for $r = 1, \cdots, N$ **do**
      if $d(\mathbf{c}_r, \mathbf{p}) \leq \mathrm{d}_{\max}$ **then**
        $\eta_{ij}^r = 1 + \frac{d^* - d(\mathbf{c}_r, \mathbf{p})}{\mathrm{d}_{\max} - d^*}$
      else
        $\eta_{ij}^r = 0$

$$\bar{p}(\theta_{ij}) = \frac{\prod_{r=1}^{N}[\frac{1}{2} + (p_r(\theta_{ij}) - \frac{1}{2})\eta_{ij}^r]^{\pi_r}}{\prod_{r=1}^{N}[\frac{1}{2} + (p_r(\theta_{ij}) - \frac{1}{2})\eta_{ij}^r]^{\pi_r} + \prod_{r=1}^{N}(1 - [\frac{1}{2} + (p_r(\theta) - \frac{1}{2})\eta_{ij}^r])^{\pi_r}}$$

Figure 15: Second Algorithm

Local efficiency second algorithm

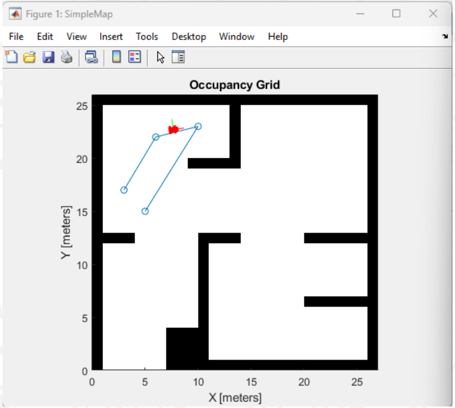


Figure 16: Robot 1 path

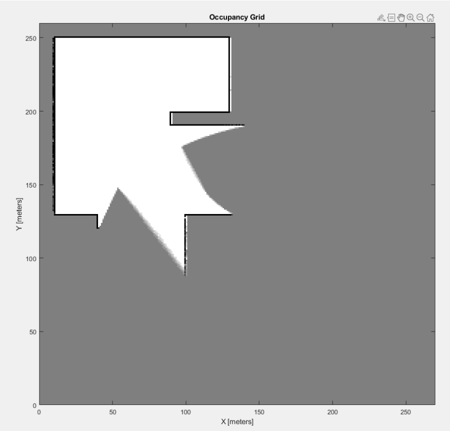


Figure 17: Robot 1 Map

Local efficiency second algorithm

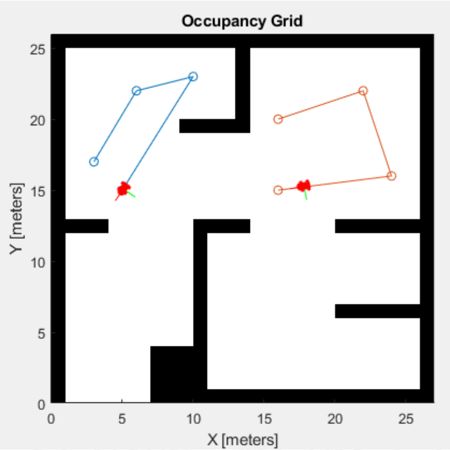


Figure 18: Robot 2 path

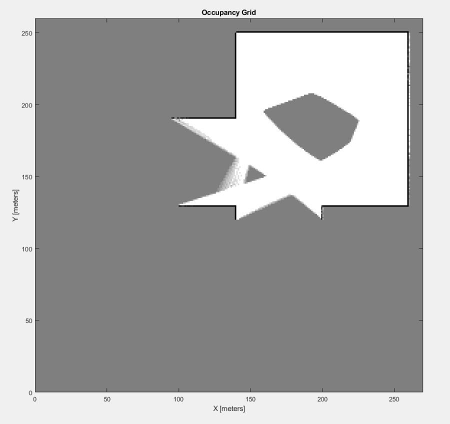


Figure 19: Robot 2 map

Local efficiency second algorithm

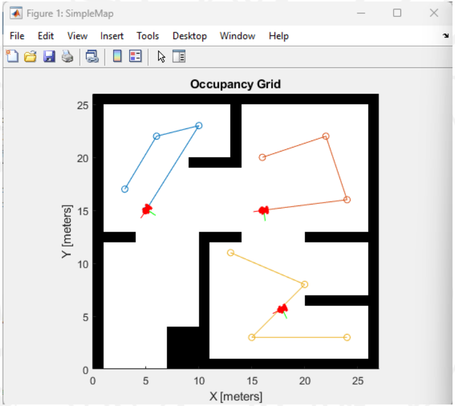


Figure 20: Robot 3 path

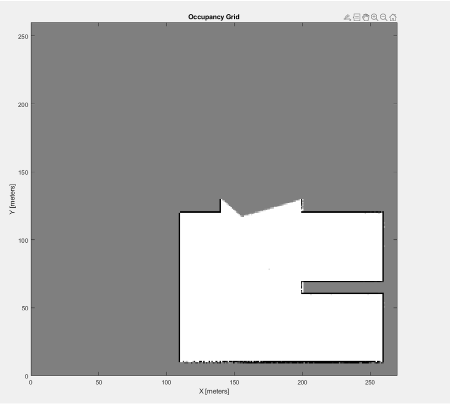


Figure 21: Robot 3 map

Local efficiency second algorithm

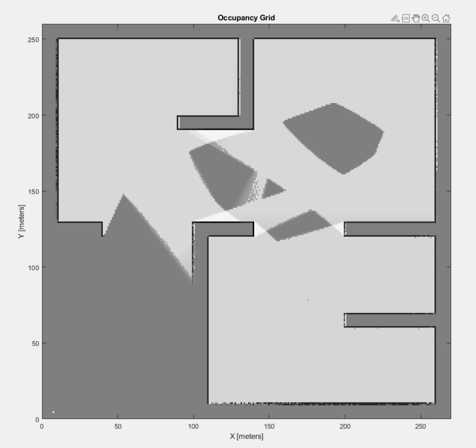


Figure 22: Final Map with no algorithm

Local efficiency second algorithm

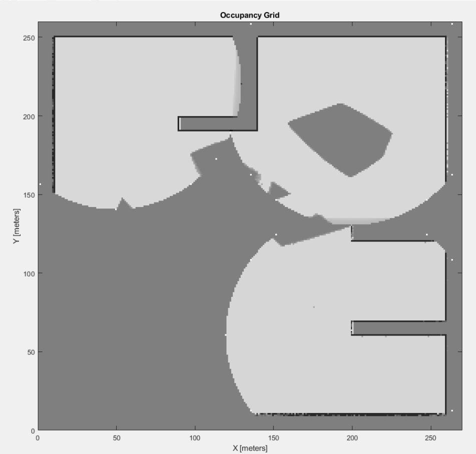


Figure 23: Final Map with $d_{max} \gg 0$

Local efficiency second algorithm

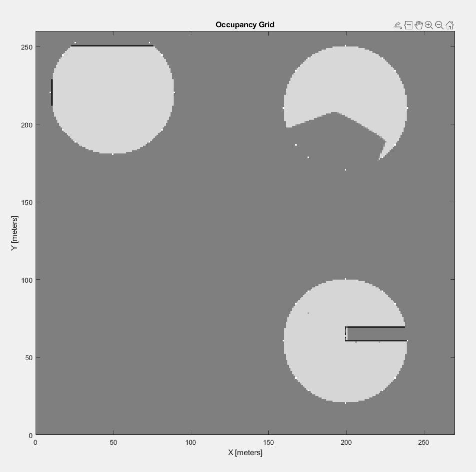


Figure 24: Final Map with $d_{\max} > 0$