Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

BACHELOR THESIS

Locating the source of a WLAN signal using multiple robots and Dec-POMDP planning

vorgelegt von

Tobias Krüger

MIN-Fakultät

Fachbereich Informatik

Studiengang: Software System Entwicklung

Matrikelnummer: 6919298

Abgabedatum: 17.08.2020

Erstgutachter: Dr. Mikko Lauri

Zweitgutachter: Michael Görner

Abstract

Gathering information with multiple robots in a decentralized manner is an interesting alternative to the use of human resources. However, to solve information gathering tasks without communication and under uncertainty, we require a reasonable approach.
Decentralized partially observable Markov decision processes (dec-POMDPs) can be used to model decentralized decision-making problems like an information gathering task even under uncertainty.
This bachelor thesis applies dec-POMDP to a signal source localization task to demonstrate the possibility of solving a continuous-state information gathering problem with a non-linear policy graph improvement (NPGI) algorithm. For the task we propose, multiple robots have to gather potentially noisy signal strength measurements in different locations, so that we can estimate the location of a WLAN signal source.
To solve this task, we introduce a software system that can be deployed on two TurtleBot2 robots and a central server. The central server utilizes an NPGI algorithm to plan and distribute policies to the two robots. Afterward, both robots gather signal strength measurements according to their policy. Finally, the central server aggregates all taken measurements to estimate a Wi-Fi router's location that serves as the WLAN signal source.
With this system, we conduct physical experiments to demonstrate that it can be applied to a real-world scenario. Additionally, we also simulate a higher number of experiments with the same system to compare its efficiency to a random movement baseline. Both approaches yield similarly good estimations of the WLAN signal source location for the experimental environment we use.

Zusammenfassung

Das Sammeln von Informationen mit mehreren dezentralisiert arbeitenden Robotern stellt eine interessante Alternative zur menschlichen Arbeitskraft dar. Um jedoch eine solche Aufgabe der Informationserfassung ohne Kommunikation und mit potenziellen Ungenauigkeiten zu lösen, benötigen wir einen angemessenen Ansatz.
Dezentralisierte teilweise beobachtbare Markov Entscheidungsprozesse (dec-POMDPs) können genutzt werden, um dezentralisierte Entscheidungsfindungsprobleme wie zum Beispiel Aufgaben zur Informationserfassung zu modellieren. Dies ist sogar dann möglich, wenn bei der Erfassung von Informationen Unsicherheiten vorliegen.
Im Rahmen dieser Bachelorarbeit wenden wir dec-POMDP zur Lokalisierung von Signalquellen an, um zu demonstrieren, dass es möglich ist eine Aufgabe zur Informationserfassung mit einem kontinuierlichen Zustandsraum durch einen NPGI-Algorithmus (non-linear Policy Graph Improvement) zu lösen. Die vorgeschlagene Aufgabe sieht vor, dass mehrere Roboter an unter-

schiedlichen Stellen Messungen von Signalstärken aufnehmen, die eventuell Ungenauigkeiten aufweisen können. Mit den aufgenommen Messungen ermitteln wir dann die ungefähre Position einer WLAN-Signalquelle.
Um diese Aufgabe zu lösen, stellen wir ein Software-System vor, welches auf zwei TurtleBot2 Robotern und einem zentralen Server eingesetzt werden kann. Der zentrale Server benutzt den NPGI-Algorithmus, um Befehlssätze für die beiden Roboter zu planen. Anschließend sammeln die beiden Roboter Messungen der Signalstärke entsprechend ihres jeweiligen Befehlssatzes. Die aufgenommenen Messungen werden abschließend zur Auswertung an den zentralen Server gesendet, um die ungefähre Position des Wi-Fi Routers, welcher als Quelle des WLAN Signals dient, zu bestimmen.
Mithilfe dieses Systems führen wir mehrere Experimente durch um zu demonstrieren, dass es auf ein Szenario in der echten Welt angewendet werden kann. Darüber hinaus führen wir auch eine größere Menge an Experimenten in Simulation durch, sodass ein Vergleich zwischen unserem und einem auf zufälligen Bewegungen basierenden Ansatzes möglich ist. Wir stellen fest, dass beide Ansätze ähnlich gute Annäherungen an die echte Position der WLAN-Signalquelle für die gewählte Experiment-Umgebung erzeugen.

Contents

1 Introduction

1.1 Background and Motivation

With billions being invested into space exploration [OECD: Organisation for Economic Co-operation and Development 2014] and more than 80 percent of the world's oceans unmapped, unobserved and unexplored [National Oceanic and Atmospheric Administration 2018], there are a lot of open research topics for which the gathering of information through human exploration is merely unreasonable. It requires a lot of effort and investment to keep humans safe in such hostile environments like outer space or the deep sea. Therefore, it is often easier and more sensible to use robots developed for information gathering tasks in those areas, to minimize human exposure to danger.
Robots can be specifically constructed to withstand specific environments and are thus ideally suited to complete information gathering tasks in outer space, underwater and in areas with extreme temperatures or biological hazards.

Furthermore, there are a few more basic examples where information gathering through robots could be useful, even though not strictly necessary. For instance, multiple robots could patrol a perimeter and gather information about their surroundings to detect suspicious activities, as suggested in 'Robot Plans Execution for Information Gathering Tasks with Resources Constraints' [Wang, Dearden, and Hawes 2015]. Another interesting example could be the application of drones and robots to gather information on public Wi-Fi networks and their access points. This information could then be used to find the nearest access point of a particular network or advise users which public network has the strongest reception in their location. Previous work in this area has applied crowdsourcing techniques to localize the source of a Wi-Fi signal [Wu and Luo 2015]. However, this strategy does require an incentive to motivate people to help and collect the necessary information [Wu and Luo 2014]. Information gathering through robots could potentially be used in this area to fill in missing data or replace the crowdsourcing approach entirely.

But since robots other than humans cannot rely on intuition and instead require an explicit instruction set to complete a task, the question arises what the best instruction set for a specific task might be.

There are many ways to give a robot a set of instructions. First of all, one could program the robot with a strictly defined behavior that would generate instructions based on its observations. It is also possible that this behavior is not directly defined by a programmer, but instead through a machine learning algorithm or artificial intelligence. Thirdly, and for the given proposal more importantly, a planning algorithm could generate a set of instructions and send it to the robot. It would specifically design the plan to solve the task ahead so that the robot is not required to make complex decisions while executing it. Instead, the robot repeats what the planning algorithm determined to be the best course of action step by step. This approach would also make it possible to coordinate multiple robots ahead of their deployment, without relying on them to communicate during the execution of their information gathering task.

One way to achieve this preplanning for multiple robots is by using the dec-POMDP framework. Dec-POMDP stands for **dec**entralized **P**artially **O**bservable **M**arkov **D**ecision **P**rocess and is a mathematical framework. It is generally used to model cooperative tasks with multiple agents working together to achieve a common goal, while also accounting for a partially unknown environment and missing communication between those agents [Oliehoek and Amato 2016]. A planning algorithm for dec-POMDPs can create a joint policy considering all the possible actions and states and derive local policies for all the used robots from it. These local policies are instruction sets executed by the robots to accomplish a certain part of the goal. Because of the planning stage, each robot's local policy is designed to complement the others, without requiring direct cooperation.

This approach can be used to solve continuous-state information gathering tasks in simulation [Lauri, Pajarinen, and Peters 2020] and could potentially be applied to a few of the examples and research topics mentioned above. This thesis demonstrates that dec-POMDP can also successfully be applied to model a continuous-state information gathering task in a real-world environment. As an example, we implement a signal source localization task.

1.2 The Scope of this Thesis

This thesis aims to demonstrate that we can use dec-POMDP to model a signal source localization task with a continuous state space and that a planning algorithm can be applied to find an efficient solution to the modeled problem. For this purpose, we implement, test and evaluate a small example with two robots as agents.

The task itself is to locate the source of a Wi-Fi signal, and the agents we use are two TurtleBot2s [1], which can take signal strength measurements. To get an estimate of the Wi-Fi router's location, we use a particle filter to evaluate the taken measurements. The evaluation process as well as the dec-POMDP algorithm are running on a central computer that can communicate with the robots before and after a complete run.

In contrast to other related work outlining the dec-POMDP framework's application to cooperation tasks with a clearly defined goal like the grid meeting problem described in [Eker et al. 2011], this task does not have a clear goal state. Instead, it only defines the maximum

[1]TurtleBot2 is a Robot with an open hardware design by the Open Source Robotics Foundation. More information available at: `https://www.turtlebot.com/turtlebot2/`, Accessed: 16.08.2020

number of execution steps. Each of the execution steps consists of a movement action and taking a signal strength measurement. After all robots have reached this execution step limit, a run is considered to be complete, and the evaluation is triggered.

The robots used for this bachelor thesis have been supplied with additional hardware by the TAMS[2] research group. However, the original robot TurtleBot2 is equipped with everything necessary to recreate the experiment. The additional hardware equipment added to the robot contains a laser-scanner and a laptop with more computational power than the original netbook. This upgrade helps the robot with its basic movement capabilities and path-finding skills. Therefore, using otherwise equipped TurtleBot2s may result in different execution times.

The proposed information gathering task environment is the robotic lab at the TAMS research group. Each of the robots is equipped with a predefined map of the area to enable path-finding. A smartphone is placed in the environment ahead of time to function as the Wi-Fi router that is supposed to be located.

The implementation of a planning algorithm for dec-POMDPs is not a part of this thesis. Instead the algorithm provided by Mikko Lauri et al. in "Multi-agent active information gathering in discrete and continuous-state decentralized POMDPs by policy graph improvement" [Lauri, Pajarinen, and Peters 2020] is used and adjusted to fit this particular example. First of all, that means defining a discrete action and observation space. Secondly, it means implementing a way for the robots to execute the policies generated by the algorithm.

This thesis's main contribution is implementing a system to run multiple experiments for the above-described example task in a real-world environment and evaluating the findings. In particular, it aims to compare the benefits and drawbacks of using a dec-POMDP algorithm to complete an information gathering task. To achieve this, we set a baseline by implementing a random movement strategy. Instead of following a policy, each robot chooses its next location at random when this strategy is used. Experiments with the random movement approach are using the same environmental variables as the planning algorithm so that we can directly compare the results of both approaches to one and other. To gain a sufficiently large set of data for the comparison between the dec-POMDP approach and the random movement baseline, we also conduct several experiments in a simulated environment.

[2]Technical Aspects of Multimodal Systems (TAMS) is a research group at the University of Hamburg

2 Background Knowledge and Related Work

The following chapter gives an insight into the technologies used in this thesis by describing how they work and why they are important to this thesis. It also discusses other works related to dec-POMDP and signal source localization to better place the proposed information gathering task.

2.1 Particle Filter

A particle filter is a way to represent a probability distribution through a set X_t of M so-called particles at the time t [Thrun, Burgard, and Fox 2005]. These particles each represent a sample from the distribution. Therefore they always consist out of a value $x_t^{[m]}$ describing their position and a weight value $w_t^{[m]}$ that describes the probability at the particles position. The weight of a particle defines its importance and increases with higher probability. A lower probability results in lower weight and, therefore, lower importance. The set of all particles and in particular their weights represent the belief of the particle filter.

For this thesis, a particle filter is used to estimate a Wi-Fi router's location in the real world. That means each particle represents a possible location of the router and the probability of it being in exactly that location. The belief of the particle filter could, for example, be represented using a map containing color-coded pixels to make the result more easily understandable for humans.

To get a belief that represents a good estimation of the true probability distribution, the particle set has to be recursively constructed by a particle filter algorithm. Throughout this thesis, we use the sequential importance resampling (SIR) algorithm with the adaptive resampling method [Särkkä 2013, p. 124-126]. To use this algorithm, first the starting set of M particles has to be initialized by sampling at random from the state space. The starting weight for each particle is $1/M$.
During each iteration we sample M particles from the importance distribution $p(x_t|u_t, x_{t-1})$, which is based on the previous state of the particle x_{t-1} and the control value u_t at the current time step t. The control value in our case for example is a signal strength measurement [Thrun, Burgard, and Fox 2005].
Using the new particle and the control value taken at the current time step, the weight $w_t^{[m]}$

for the given particle can be calculated using Equation 2.1 [Särkkä 2013].

$$w_t^{[m]} \propto w_{t-1}^{[m]} \frac{p(u_t|x_t^{[m]})p(x_t^{[m]}|x_{t-1}^{[m]})}{p(x_t^{[m]}|u_t, x_{t-1}^{[m]})} \tag{2.1}$$

The next step of the algorithm is resampling. However since we use adaptive resampling the algorithm first checks the effective number of particles which we calculate as shown in equation 2.2 [Särkkä 2013].

$$n_{eff} \approx \frac{1}{\sum_{i=1}^{M}(w_t^{(i)})^2} \tag{2.2}$$

Only when this number drops below a custom-defined threshold, the algorithm needs to perform the resampling procedure. It draws M particles from the previous particle sets in a way where particles with high weight are potentially drawn multiple times, while particles with a low weight are potentially not drawn at all. The probability of drawing a specific particle from the previous set is equivalent to the particle's weight. Afterward, the algorithm sets the weight of each particle in the new set to $1/M$. This resampling procedure ensures that we do not have only a few particles with a high weight, while most particles have a weight close to zero. Instead, we get a distribution that is more focused on a relevant part of the state space.

The resulting set of particles then represents the belief of the particle filter. With multiple repetitions, this result increases its accuracy.

2.2 Robot Navigation

During the policy execution, each of the agents needs to move to given positions. To do so, the robots need to locate themselves and plan collision-free paths to said position. For this thesis, we are using the Adaptive Monte Carlo Localization (AMCL) supplied by the amcl [1] package, which is based on the algorithm introduced in 'KLD-Sampling: Adaptive Particle Filters' [Fox 2001]. This package allows a robot to locate itself in an environment based on laser scans. Furthermore, we are using the move_base package [2], which allows for high-level interaction with the navigation stack of a robot. It provides an interface where a goal location can be supplied and tries to plan and execute a path for the robot to move to a given destination. In combination with a given map of the environment, the move_base package and the amcl package enable a robot to determine its location on the map and, based on that, move towards its destination.

AMCL is a localization approach for a two dimensional known map that uses an adaptive particle filter to approximate the current location of a robot [Fox 2001]. This concept is

[1]A package by the Open Source Robotics Foundation, Inc Available at: `http://wiki.ros.org/amcl`, Accessed 06.08.2020

[2]A package by the Open Source Robotics Foundation, Inc Available at: `http://wiki.ros.org/move_base`, Accessed 06.08.2020

described in much more detail by Thrun et al. [Thrun, Burgard, and Fox 2005] and Fox [Fox 2001]. But since it is a tool the robots used in this thesis need for self-localization, this section summarises of how AMCL works.

Similar to how we are using particle filters in this thesis to approximate a signal source's location, a basic Monte Carlo Localization or MCL for short uses the belief of a particle filter to portray the potential locations of a robot. Each particle represents a location where the robot could be, and the weight of the particle represents the likelihood of the robot being in precisely that location [Thrun, Burgard, and Fox 2005]. The adaptive version of the Monte Carlo Localization adjusts the sample size for each resampling step while constructing the particle filter. If the uncertainty of the robot's location is high, a larger number of samples is used. If the uncertainty is lower, a lower number of particles is used. This process increases the particle filter algorithm's efficiency compared to an algorithm with a fixed sample size [Fox 2001].

2.3 dec-POMDP: Basic Principals

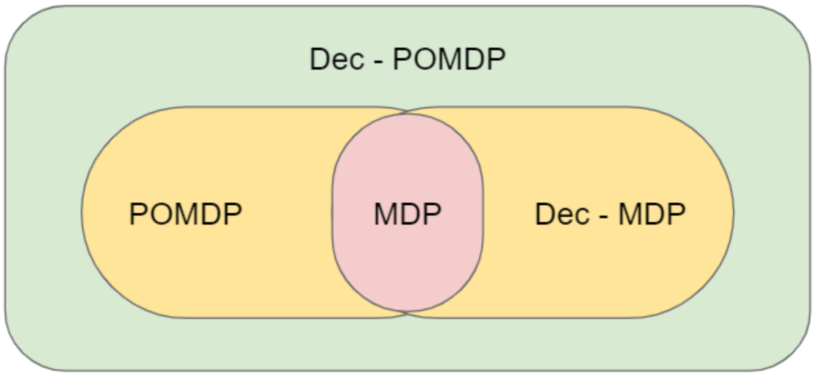
An essential part of this thesis is the concept of decentralized Partially Observable Markov Decision Processes (dec-POMDP). According to Lauri, Pajarinen and Peters "the Dec-POMDP is a general model for sequential co-operative decision-making under uncertainty" [Lauri, Pajarinen, and Peters 2020, pp. 2].

That means it is a framework that allows us to model a task using multiple robots as a set of agents working together towards a common goal, while not requiring any communication between them. Dec-POMDP can even deal with possible uncertainties during the task execution, like noise in observations or inaccuracy in actions and the system's current state. This section discusses different versions of Markov Decision Processes and explains the basic principles of the dec-POMDP. It also introduces the heuristic algorithm for information gathering dec-POMDPs that is used throughout this thesis and discusses alternative application areas for dec-POMDPs.

To better understand what exactly dec-POMDP is, it is helpful to look at Partially Observable Markov Decision Processes (POMDP), decentralized Markov Decision Processes (dec-MDP), and Markov Decision Processes (MDP). All these different frameworks are compared to one another in detail in 'The Complexity of Decentralized Control of Markov Decision Processes' [Bernstein et al. 2002] and are related to Dec-POMDP, as shown in Figure 2.1. Further information on the different versions of Markov Decision Processes can also be found in [Thrun, Burgard, and Fox 2005] or [Oliehoek and Amato 2016].

According to [Thrun, Burgard, and Fox 2005] a Markov Decision Process assumes a fully observable environment at all times, but allows for a possible non-deterministic action model. A planning algorithm, which is supposed to implement MDP, generates a plan enabling an agent to execute actions based on observations instead of sequentially going through a list of

Figure 2.1 The relationship among different versions of Markov Decision Process models



actions. A plan like this is called a policy. Using the basic MDP model we could implement a simple task using a single agent in a fully observable environment.
An example of a task that could be modeled as a Markov Decision Process, is a robot that has to move to a specified destination while avoiding an obstacle as described in [Thrun, Burgard, and Fox 2005]. This task's policy would consist of actions that make the robot move a certain direction and distance towards this destination. These movements do not have to be accurate, but after each movement, the observation taken by the robot has to result in an accurate estimation of the current state of the system. In this case, the current state of the system is the actual location of the robot.

The problem of finding a policy to solve an MDP task that is at least as efficient as a certain threshold is P-complete [Bernstein et al. 2002]. To determine a policy's efficiency, we can calculate its value depending on a reward function that determines the reward for taking a specific action in a specific state. An implementation example for MDPs can be found in [Bernstein et al. 2002].

If the state is not fully observable, we need to use a Partially Observable Markov Decision Process (POMDP). Unlike the basic MDP model, this allows for noise in the observations that the robot takes while executing a policy [Thrun, Burgard, and Fox 2005]. That means that the state of the system may be unclear even after the observation. To compensate for this, a policy generation algorithm has to use a belief state in order to determine the value of a policy. According to [Bernstein et al. 2002] finding a policy for a POMDP, which is at least as efficient as a certain threshold, is PSPACE-complete.

When the state is fully observable, but we want to use multiple agents, we can apply the concept of a decentralized Markov Decision Process (dec-MDP). A task modeled through dec-MDP makes use of multiple agents that work together to achieve a common goal without requiring any communication with each other. That means a single agent does not know the overall state of the system. Instead, the state is jointly observable [Bernstein et al. 2002] and can only be determined by combining the observations of all agents. Likewise, the state

transition does not depend on a single action, but instead on the combination of actions taken by all of the involved agents. To ensure that the goal state will be reached, a planning algorithm for dec-MDP has to construct local policies for each agent. All of the local policies together make up a joint policy. The task of finding a joint policy for two or more agents that solves a given problem optimally is NEXP-hard for two or more robots [Bernstein et al. 2002].
An example of the application of dec-MDP can be found in "Applications of DEC-MDPs in Multi-Robot Systems" [Beynier and Mouaddib 2011]. They have applied this concept to a scenario where two robots have to explore eight places of interest, similar to a Mars rover mission.

Finally, a decentralized Partially Observable Markov Decision Process (dec-POMDP) combines both previously introduced concepts. It makes use of multiple agents like the dec-MPD, and it allows for inaccuracy in the observations like the POMDP. A formal definition for a dec-POMDP is the tuple $(T, I, S, A_i, Z_i, P^s, P^z, b^0, p_t)$ [Lauri, Pajarinen, and Peters 2020, pp. 5-6] where the individual variables are defined as follows:

- $T \in \mathbb{N}$ is the maximum number of time steps.
- $I = \{1, ..., n\}$ is a set of n agents.
- S is a finite set of hidden states where s^t denotes the state at time step $t \in T$. The set of states depends on the initial joint belief b^0.
- A_i is the collection of all action sets. Each agent $i \in I$ has one finite set of local actions. Furthermore, we define A as the joint action space where each joint action at a certain time step consists out of all the local actions taken at that time step.
- Z_i is the collection of all observation sets. Each agent $i \in I$ has one finite set of local observations. Furthermore, we define Z as the joint observation space where each joint observation at a certain time step consists out of all the local observations taken at that time step.
- P^s is the state transition probability. It can be used to determine the probability of a certain state at the next time step s^{t+1} based on the current state s^t and the combined action $a^t \in A$.
- P^z is the observation probability. It can be used to determine the probability of a certain joint observation z^{t+1} based on the state s^{t+1} and the joint action $a^t \in A$.
- $b^0 \in \Delta(S)$ where $\Delta(S)$ is the space of probability mass functions over S, defines the joint belief at the initial time step.
- p_t is the set of all reward functions which determine the reward at different time steps $t \in T$. We also define p_T to be the function for the final reward at time step T.

This definition includes the set of agents I, the set of joint actions A, and the set of joint observations Z a dec-MDP definition contains [Beynier and Mouaddib 2011]. It also includes a state transition probability P^s and an observation probability P^z similar to the ones a POMDP definition contains [Bernstein et al. 2002]. In contrast to a regular POMDP, the definition for a dec-POMDP uses the probability of a combined action or observation instead of the probability of a single action or observation.

Using this definition, we can model an information gathering task like the source localization proposed in this thesis. The problem of optimally solving a dec-POMDP is NEXP-hard for a number of robots n where $n \geq 2$ [Bernstein et al. 2002].

2.3.1 The Planning Algorithm for Information Gathering dec-POMDPs

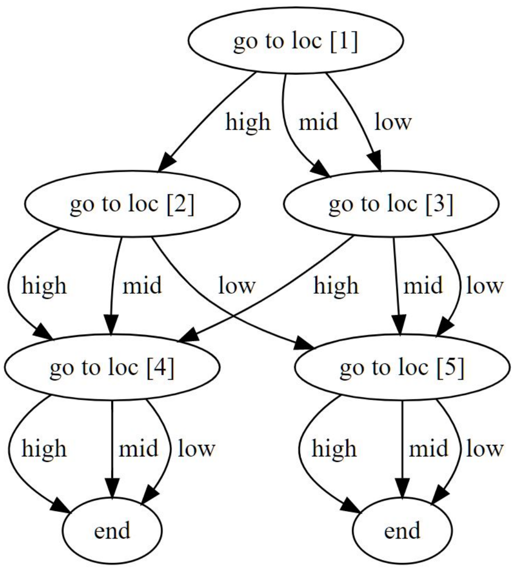
This section's content is based on the work by Lauri, Pajarinen, and Peters [Lauri, Pajarinen, and Peters 2020]. It gives an overview of the non-linear Policy Graph Improvement algorithm (NPGI)[3] algorithm they introduce to solve dec-POMDPs that model information gathering tasks by generating policy graphs for each of the employed agents. These policies are custom-tailored to the possible actions and observations that the particular agent can execute or perceive. The individual policies for each agent are called local policies, while the combined collection of all local policies is called a joint policy.

Local policies are defined as $\pi_i = (Q_i, q_i, \gamma_i, \lambda_i)$ for each agent i. Each policy π_i is a representation of a decision tree with Q_i as a set of nodes, q_i as the starting node, γ_i as a collection of actions, and λ_i as a node transition function. An example of how a policy can be visualized is given in Figure 2.2. A node always corresponds to an action in γ_i. It is also possible that multiple nodes correspond to the same action. In the given example, the actions are movement instructions that require the robot to move to a certain location. The node transition function γ_i defines outgoing edges for each node. The decision on which edge gets activated next, is always dependent on the observation made in the current node. In the example shown in Figure 2.2 three discrete measurement values are uses as the possible observations

To generate efficient policies for a dec-POMDP problem, it is necessary to define reward functions p_t for the dec-POMDP that determine how useful it is to execute an action in a certain state. In the given case, these reward functions are convex functions depending on the agents' joint belief. To optimize the joint policy during the planning stage, the expected sum of rewards obtained when each local policy is executed accordingly should be maximized. To achieve this, we use the NPGI algorithm.

[3]The exact implementation we are using can be found under `https://github.com/laurimi/npgi` Accessed: 14.08.2020

Figure 2.2 An example of how a policy graph might look like. The actions, in this case, are movement instructions and the observation that determine along which edges the execution progresses are discrete measurement values.



Algorithm 1 NPGI Algorithm according to [Lauri, Pajarinen, and Peters 2020, p. 16]

```
Input: Policy $\pi = (Q, q_0, \gamma, \lambda)$, initial belief $b^0$
Output: Improved policy $\pi$
1: while not converged and time limit not exceeded do
2:     $B \leftarrow FORWARDPASS(\pi, b^0)$
3:     $\pi^+ \leftarrow BACKWARDPASS(\pi, B)$
4:     if $V_0^{\pi^+}(b^0, q_0) \geq V_0^{\pi}(b^0, q_0)$ then $\pi \leftarrow \pi^+$
5:     end if
6: end while
7: return$\pi$
```

The algorithm takes in a randomly generated joint policy π and improves it over a set amount of time or until the expected sum of rewards has converged. This requires two steps during each iteration. First the forward pass and secondly the backward pass. Both of these are described in more detail in Lauri, Pajarinen and Peters work [Lauri, Pajarinen, and Peters 2020, pp. 16-19]. This section only offers a short introduction to convey a better understanding of the NPGI Algorithm.

The forward pass iterates over all possible paths in the joint policy graph $\pi = (Q, q_0, \gamma, \lambda)$ and determines the expected belief b_q for every joint policy graph node $q \in Q$. It then returns the set $B = \{b_q | q \in Q\}$ of all expected beliefs. We can calculate those beliefs for continuous state problems using a particle filter algorithm.

The backward pass in contrast iterates over all nodes $q_i^t \in Q_i^t$ at time step t in each agents local policy $\pi_i = (Q_i, q_{i,0}, \gamma_i, \lambda_i)$ and tries to improve them by adjusting the action for the specific node or the node transition function λ_i. For each node, the backward pass maximizes the nodes values lower bound by finding the optimal action. To always find the optimal action the algorithm starts at time step $t = T - 1$ and goes backwards from there. This way the algorithm can directly choose the optimal actions at each iteration, without planning ahead for future time steps to determine the values of nodes.

After these two steps have been completed, the NPGI algorithm checks whether the new policy has a higher value then the old one. If it does, the old joint policy gets replaced. If not, the old one gets used again for the next improvement iteration. To ensure the joint policy does not get stuck in a local maximum this way, we use some randomization during the backward pass.

2.3.2 Other Applications of dec-POMDP

In this thesis, we use dec-POMDP to model a signal source localization task, but it is also interesting to consider other examples of tasks that can be modeled as dec-POMDPs. This section introduces other applications of the dec-POMDP framework which were implemented in related works and compares them to the task proposed for this thesis.

One example of how dec-POMDP can be applied, can be found in 'A finite horizon DEC-POMDP approach to multi-robot task learning' [Eker et al. 2011]. Eker et al. describe multiple cooperative tasks for two robots. The two robots each get supplied with a local policy by a dec-POMDP planning algorithm, which gets trained in a simulated environment to reach optimal policies for the given problem. This strategy was tested in three examples. The first task is for the two robots to find each other in a 3x3 grid regardless of their starting position. The second example added another factor to the first task, by adding a two cells wide obstacle in the middle of the grid. This results in a U shaped grid, which is more complicated to handle for the robots. The last example mentioned is using two robots to push a box towards a predefined destination. For this task, the robots had to work together to achieve a common goal without communicating. [Eker et al. 2011]

The above-described example uses a very different approach to what this thesis is proposing. Instead of trying to achieve a certain goal state, the example we are proposing is an information gathering task. An information gathering task in itself has no goal state. Instead, it is limited to a predefined time period, after which the gathered information is evaluated to gain knowledge. Furthermore, Eker et al. suggest using evolution strategies to generate effective policies in contrast to the NPGI algorithm that we are using [Eker et al. 2011].

Akin et al. [Aşık and Akın 2013] propose another more complex example usage of dec-POMDP. They apply the dec-POMDP framework on a robot soccer game in a simulated environment to develop team strategies. To find the best policy for a soccer team to win, they use a genetic algorithm. The most interesting thing about this approach is that they applied it to an example with five agents, which increases the complexity of the problem compared to the two robots that we use. However, the task they propose has a clear goal state, unlike the information gathering task, this thesis introduces.

2.4 Signal Source Localization

Another very interesting area of related work to this thesis are different methods for signal source localization. This chapter introduces different techniques that can be used to locate a signal source or a signal receiver. It also discusses different approaches to signal source localization and indoor localization from related works.

'A Survey of Indoor Localization Systems and Technologies' by F. Zafari, A. Gkelias, and K. Leung [Zafari, Gkelias, and Leung 2017] aims to introduce different indoor localization techniques and compare localization systems that are using these techniques. Unlike this bachelor thesis, they only consider using these techniques to enable a mobile device to localize itself in an indoor environment. Therefore, their work is not directly comparable to the work accomplished throughout this thesis. However, the techniques introduced and discussed in [Zafari, Gkelias, and Leung 2017] are relevant for the signal source localization task proposed in this thesis.

All of the introduced techniques could also be applied to a signal source localization problem, where instead of multiple known router locations, the mobile device's location is known at each point in time. Therefore, it makes sense to consider all of them when deciding how to implement the signal source localization. This section gives a short overview of all techniques introduced in [Zafari, Gkelias, and Leung 2017]. It also introduces a different approach to signal source localization taken by Nikolay Atanasov et al. in their paper 'Distributed Algorithms for Stochastic Source Seeking With Mobile Robot Networks' [Atanasov, Le Ny, and Pappas 2014].

Angle of Arrival (AoA) is a localization technique that typically uses an array of antennae to estimate the angle of an incoming signal by calculating the arrival time difference between the different antennae [Zafari, Gkelias, and Leung 2017]. If we calculate this angle in two positions, we could estimate our position relative to the signal source or the signal source location if we know our location. However, this requires an antennae array, which the robots we are using do not have. Therefore, this technique can not be used.

Time of Flight (ToF) or Return Time of Flight (RToF) are both techniques that depend on measuring the time it takes a packet to travel between the transmitter and the receiver. If the value is multiplied by the speed of light, we get the distance between receiver and transmitter. If these measurements are taken for multiple transmitters, they could be used for self-localization, as suggested in [Zafari, Gkelias, and Leung 2017]. But if instead multiple measurements are taken in different locations for one transmitter, they could be used to localize the transmitter. However, this technique requires either strict time synchronization between receiver and transmitter for ToF, or at least well defined communication between the transmitter and the receiver. For this thesis however, we do not aim to implement any additional communication abilities to the transmitter because the information gathering experiment should show results regardless of the Wi-Fi router used. The implementation this thesis is proposing does not require the receiver to be connected to the Wi-Fi network.

Received Signal Strength (RSS) is one of the most straightforward approaches for indoor localization and widely used according to [Zafari, Gkelias, and Leung 2017]. It is the actual strength measured in decibel-milliwatts (dBm) at the receiver and can be used to estimate the distance between transmitter and receiver [Zafari, Gkelias, and Leung 2017]. With one measurement and multiple transmitters, a mobile device's location can be estimated relative to the transmitters using this approach and basic geometry. But more importantly, this approach can also be used by utilizing multiple measurements in different locations to localize one transmitter.
The operating system, combined with the Wi-Fi chip the robots are already equipped with for our proposed experiment, enables us to measure the received signal strength of nearby Wi-Fi networks. Unlike the Angle of Arrival approach, this technique does not require any additional hardware or additional implementations on the transmitter side, like in the ToF or RToF approach. Therefore, this is the approach we are using throughout this thesis.

According to [Zafari, Gkelias, and Leung 2017] there is a tool called RADAR, which uses relative measurements of the received signal strength called RSS indicator values to estimate the location of a user. This tool achieves a median accuracy of 2.94 meters or even 2.5 meters if a Kalman filter is used to improve the result.

The journal paper 'Distributed Algorithms for Stochastic Source Seeking With Mobile Robot Networks' [Atanasov, Le Ny, and Pappas 2014] introduces multiple approaches to source seeking using received signal strength measurements. The most interesting one to this thesis is a distributed model-based algorithm, which can be compared to the approach that we are taking.
A network of ten distributed robots is used as sensors to localize a wireless radio signal source. They introduce a model for the received signal strength for this specific case and use it to determine the distance between each of the sensors and the signal source. Using this distance, a particle filter is constructed to estimate the location of the signal source. N. Atanasov et al. managed to estimate the transmitter's position within 2.96 meters in a simulated obstacle-free environment. In contrast to the goal of this thesis however, their proposed solution does not use a planning algorithm. Instead, the robots try to maximize the mutual information gradient between their expected measurements and the signal source location during each step of the experiment.

Another approach to signal source localization is presented by Twigg et al. [Twigg et al. 2012]. They only use a single robot to locate a radio signal source through gradient assisted exploration. Unlike the solution we are suggesting, this approach does not require prior knowledge of the environment except for information about the fading [Twigg et al. 2012]. Also, their approach does not require any planning ahead of time. However, the presented algorithm is only designed for a single robot [Twigg et al. 2012], while our approach can utilize multiple robots.

Instead of using robots, it is also possible to rely on humans to collect the necessary information to approximate a Wi-Fi signal source's location. Through crowdsourcing techniques, it is possible to utilize users' mobile devices to collect signal strength measurements [Wu and Luo 2015] [Wu and Luo 2014]. Similar to the technique we are using to approximate the signal source location, this approach also relies on multiple measurements in different locations to calculate the probability of the signal source location being in a specific location. But while we record each measurement with the same hardware, their approach has to deal with measurements taken by various mobile devices. Also, employing robots instead of humans to solve a signal source localization task eliminates the necessity of incentivizing human participation, which Wu et al. describe as a big challenge of crowdsensing techniques [Wu and Luo 2014].

3 Methodology

This chapter describes the methods used for the practical part of this thesis. In particular, it explains how we gather Wi-Fi signal strength measurements and how we apply a dec-POMDP planning algorithm to the signal source localization problem. Furthermore, it introduces the system we develop for the proposed signal source localization task. This system incorporates the NPGI algorithm to generate policies, executes the generated policies on two robots, and evaluates the gathered signal strength measurements to estimate the signal source location.

3.1 Wi-Fi Signal Strength modeling

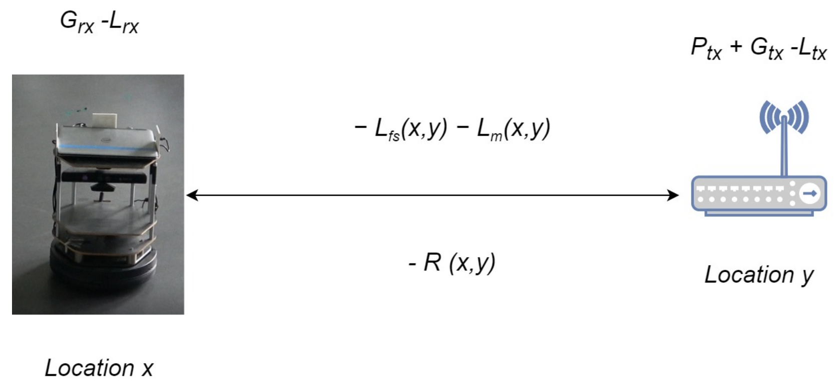
An essential part of this thesis is providing the robots and the server with methods to handle measured Wi-Fi signal strengths. To execute a given policy, a Wi-Fi signal strength measurement needs to be taken and converted to a discrete value after each executed action. This conversion process requires a well-defined method to interpret some signal strength value. Also, after finishing a policy, the gathered measurements have to be evaluated to approximate the location of the router.

This section describes how exactly signal strength measurements are taken and evaluated. We introduce an equation to model the received signal strength, enabling us to construct a particle filter from signal strength measurements. The model also allows us to empirically choose intervals for discrete signal strength values, based on the size of the room in which the experiment is conducted. Furthermore, this section discusses the impact of chance and orientation on a single measurement and proposes a combination of multiple measurements and constant rotation during the measuring process to mitigate that impact.

Using the command-line tool iwlist[1] with the scanning parameter, the Wi-Fi Adapter installed on the robots laptop can be instructed to scan all available Wi-Fi networks. The result of this scan also contains the received Wi-Fi signal strength of each access point in dBm. To interpret the measured value, it is necessary to convert the dBm value to one of the discrete signal strength values predefined for the dec-POMDP algorithm.
This can easily be achieved by comparing the measured value with the predefined interval for each discrete value and converting it to the matching one. To make sure the mentioned discrete values and their intervals can be used as expected, it is necessary to fit the intervals to the experimental setup.

[1]iwlist is a ubuntu command-line tool. Additional information can be found under `http://manpages.ubuntu.com/manpages/xenial/en/man8/iwlist.8.html`, Accessed 10.08.2020

Figure 3.1 The RSS model as originally introduced in [Atanasov, Le Ny, and Pappas 2014]



G_{rx}:	receiver gain	L_{rx}:	receiver loss
G_{tx}:	transmitter gain	L_{tx}:	transmitter loss
P_{tx}:	transmitter output power	$R(x, y)$:	noise
$L_{fs}(x, y)$:	free space loss	$L_m(x, y)$:	multi path loss
$Location\ x$:	receiver	$Location\ y$:	transmitter

Furthermore, the particle filter evaluation of the acquired results depends on a method to interpret the signal strength as well. In this case, we do not have to convert the measurements into discrete values. Instead, we compare each measured signal strength value to the expected value for the distance between a particle and the location where this measurement has been taken. The comparison between the real and the expected value can be used to adjust a particle's weight. Through this method, we can apply the sequential importance resampling described in section 2.1 to construct a particle filter.

To achieve both, fitting the intervals for the discrete values used in the policies and generating the expected signal strength for a given distance, it is essential to create a model first. This model should represent the expected received signal strength value for a certain distance. It also enables us to empirically choose intervals for the discrete values mentioned earlier based on the distances we expect to see during an experiment.

To model the received signal strength, a variation of the model suggested in [Atanasov, Le Ny, and Pappas 2014] is used for this thesis. The original model proposed by Nikolay Atanasov et al. is shown in equation (3.1). Figure 3.1 also shows a more detailed explanation of the original model.

$$P_{rx}(x, y) = P_{tx} + G_{tx} - L_{tx} + G_{rx} - L_{rx} - L_{fs}(x, y) - L_m(x, y) - R(x, y) \tag{3.1}$$

$P_{rx}(x, y)$ is the received signal strength value with x being the Wi-Fi receiver's location and y the location of the Wi-Fi transmitter. The equation to calculate this value consists out of four terms.

First of all, there are five experiment specific constants. P_{tx} is the transmitter output power and gets added to G_{tx} the transmitter gain and G_{rx} the receiver gain. The transmitter loss L_{tx} and the receiver loss L_{rx} are constants as well and get subtracted from the other constants.

The next step is to subtract the Free Space loss $L_{fs}(x, y)$ which is defined as shown in equation (3.2) [Atanasov, Le Ny, and Pappas 2014]. Where again x is the location of the Wi-Fi receiver and y the location of the Wi-Fi transmitter. d is the distance between receiver and transmitter and f is the transmitted frequency in hertz. For the proposed experiment this frequency is $f = 2.4Ghz = 2.4 \cdot 10^9 Hz$.

$$L_{fs}(x, y) = -27.55 + 20 \log 10(f) + 20 \log 10(d) \tag{3.2}$$

Subtracting the multi-path loss $L_m(x, y)$ is the third step. However, the proposed experiment only takes place in a single room. Since the multi-path loss is only relevant if there are solid obstacles like walls between receiver and transmitter, it can be ignored for this calculation.

The fourth and last step is subtracting $R(x, y)$, which represents the noise. According to [Atanasov, Le Ny, and Pappas 2014], the noise can be modeled either using a Rician distribution or a Rayleigh distribution. The choice on which one to use depends on whether the measurement has been taken in line of sight or not. But again, the proposed experiment only takes place in a single room. Therefore only the Rician distribution defined as $R(x, y) = Rician(b, \sigma, loc)$ is used. b, σ and loc are all constants which we determine by fitting the curve to values measured in a test scenario.

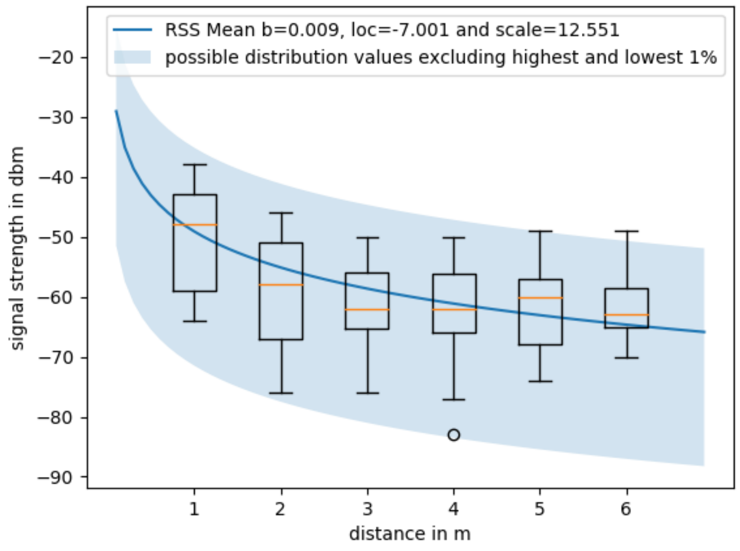
For the experiment proposed in this thesis, we use a mobile phone as the transmitter and the Wi-Fi chip of a laptop as the receiver. Both are lacking specifications for gain and loss, and the transmitter output power is unknown as well. Therefore the first portion of the equation 3.1 $P_{tx} + G_{tx} - L_{tx} + G_{rx} - L_{rx}$ consists out of only unknown variables. To compensate for this circumstance, we set them to 0 and model their impact on the equation through the Rician function as well. Another potentially viable way to cope with the missing values would have been to determine them using model fitting. But since the five values get added and subtracted from each other, the results would differ significantly depending on the used algorithm and chance. Furthermore, there would be no guarantee that the generated values would represent the real values at all since a high loss, for instance, could compensate for a high gain. That means it would be an additional effort without any information gain.

With the variables $P_{tx}, G_{tx}, L_{tx}, G_{rx}, L_{rx}$ and $L_m(x, y)$ all being set to zero, the remaining function can be reduced to the term shown in Equation (3.3). To use this function as a model, it has to be fitted to realistic data. In order to achieve this some real world data has to be recorded and evaluated.

$$P_{rx}(x,y) = -L_{fs}(x,y) - Rician(b,\sigma,loc) \tag{3.3}$$

To gather the real-world data for the model fitting process, we conduct a small experiment. A robot is placed at distances of one to six meters, and 120 signal strength measurements are taken at each distance. To determine the impact of the robots orientation on the taken measurement, these 120 measurements are split up into four different directions with 40 measurements for each orientation.
The hereby gathered data can then be used combined with the fit function supplied by the module scipy.stats [Virtanen et al. 2020] for Rician distributions, to fit the model for this thesis. This fitting process determines values for the still missing free variables. Therefore the rest of this thesis uses the parameters $b = 0.009$, $\sigma = 12.551$ and $loc = -7.001$ to model the signal strength. Figure 3.2 displays the complete model in combination with all experimental measurements. The measurements are displayed as box plots for each distance, and the blue line represents the mean signal strength value for a given distance determined by the model. The light blue background represents the range of the models' distribution, excluding the lowest and highest one percent to improve the display.

Figure 3.2 Test measurements and the resulting model

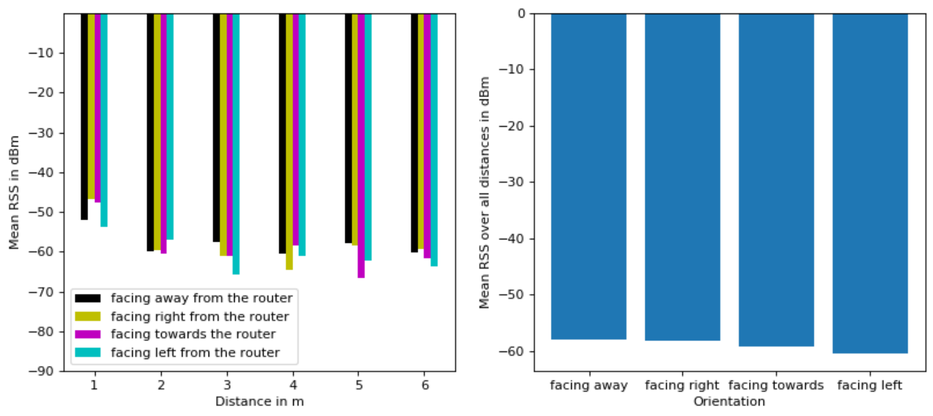


As one can see from the broad distribution of measurements displayed in Figure 3.2, the received signal strength can vary significantly from measurement to measurement. In fact, a low measurement at a one-meter distance could be worse than a high measurement at a six-meter distance. Therefore a single signal strength value is not enough to make a reasonable estimation of the distance between receiver and transmitter. To mitigate the error resulting from the chance involved in taking a single measurement, the robots always take multiple measurements for each action.

Furthermore, as suggested before, it is also important to consider the impact of the robots' rotation on the taken measurement. If the Wi-Fi receiver was placed precisely in the middle of the robot with equally obstructing parts of the robot around it, the orientation should have no impact. But since the Wi-Fi chip's exact location is unknown, the received signal strength might be weakened by obstructing robot parts in some directions.

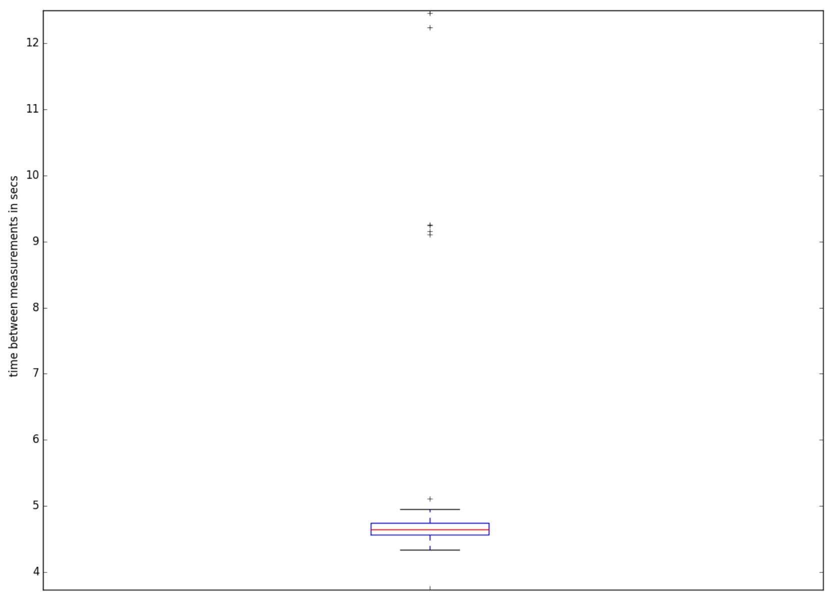
The left graphic in Figure 3.3 shows what the mean received signal strength for each orientation and distance is in the data set used to fit the model. The four considered directions are the robot facing away from the router (black), the robot facing right (yellow), or left (cyan) when view from the router or facing towards the router (magenta). The right chart displays the mean of all measurements taken for each orientation, disregarding the distance.

Figure 3.3 Mean values from test measurements by orientation



Both bar charts show that the orientation does make a difference for the taken measurement. If we consider only the right bar chart, it is possible to reason that facing away from the Wi-Fi router tends to yield the lowest signal strength values. However, if the left chart is taken into account as well, we can see that facing away from the router does not lead to the lowest average signal strength for all distances. The difference between orientations in the recorded data set might be pure chance since no clear trend is visible in the left bar chart in Figure 3.3. However, further investigation would lead beyond the scope of this thesis. Therefore it is important to take measurements independent of the orientation. To achieve this, the robot not only takes multiple measurements with each action but also rotates slowly while doing so. This constant rotation ensures that each measurement is taken with a slightly different orientation, therefore mitigating the orientation's impact on the overall result.

Figure 3.4 Average time to take a measurement based on 100 measurements



To find out how fast a robot has to rotate, we look at the average time it takes to get a measurement. Figure 3.4 displays the duration between measurements of an experiment where one robot took 100 measurements. From this experiment, we can conclude that a robot needs between four and five seconds to take a measurement on average.

The few outliers where it took longer than nine seconds to take a measurement are not taken into consideration when determining the rotation. Since, in these cases, it takes double or triple the duration to get a measurement, these outliers are likely caused by an internal retry mechanism of the network interface. Therefore, the actual result probably took only the regular four to five seconds to measure, while the left over time was spent on attempts without any result. However, this is only a theory where further investigation is not necessary within this thesis since the impact of seven outliers on the turning speed decision in a sample size of 100 is negligible.

For the rotation speed, we choose the arbitrary value $0.8\ rad$, which means that the robot turns by $0.8\ rad \approx 45.84°$ per second. This way, a full rotation takes the robot approximately eight seconds, and each measurement is taken during a little more than a half rotation. Furthermore, each of them starts and ends at orientations different from the previous measurement, so that multiple measurements are balancing each other out, therefore mitigating the impact of the orientation. The main reason behind not going for a full rotation during the measurement process is that we do not know how exactly the wireless network interface scan works on different devices. Since only one network out of all the scanned ones is interesting for us, some of the scanning time might not be relevant for our result. Therefore a full rotation during each measuring process might result in a "blind spot" on the rotation spectrum.

Moreover, the higher the speed, the higher the risk of the robot tipping over during a rotation. The rotation speed we chose does not put the robot at risk of tipping over.

3.2 Source localization using dec-POMDP

After explaining the basic principles of dec-POMDP and the NPGI algorithm introduced by Lauri, Pajarinen, and Peters [Lauri, Pajarinen, and Peters 2020] in the background knowledge and related work section 2.3, this section gives an overview of how the planning algorithm is used for this thesis. Furthermore, it outlines how we apply the dec-POMDP framework to model a signal source localization task.

The algorithm itself is already explicitly designed to solve information gathering tasks and has been shown to work on a continuous-state source seeking problem in simulation by Lauri et al. [Lauri, Pajarinen, and Peters 2020, pp. 26 - 29]. Therefore, we do not make any changes or additions to the algorithm itself. However, we do need to specify multiple input parameters in order to fit the algorithm to our real-world example.
First and most important, the planing algorithm depends on a signal strength model that fits the real world. For this, we use the received signal strength model introduced in section 3.1, since it is already fitted to the equipment used in our real-world experiment.
Using this model, we can also empirically choose discrete abstractions for the possible signal strength values. This is necessary since dec-POMDP requires a finite observation space. The number of discrete values chosen is crucial to the problem's complexity and will increase the planning time. To keep the planning time low, we use three different discrete values:

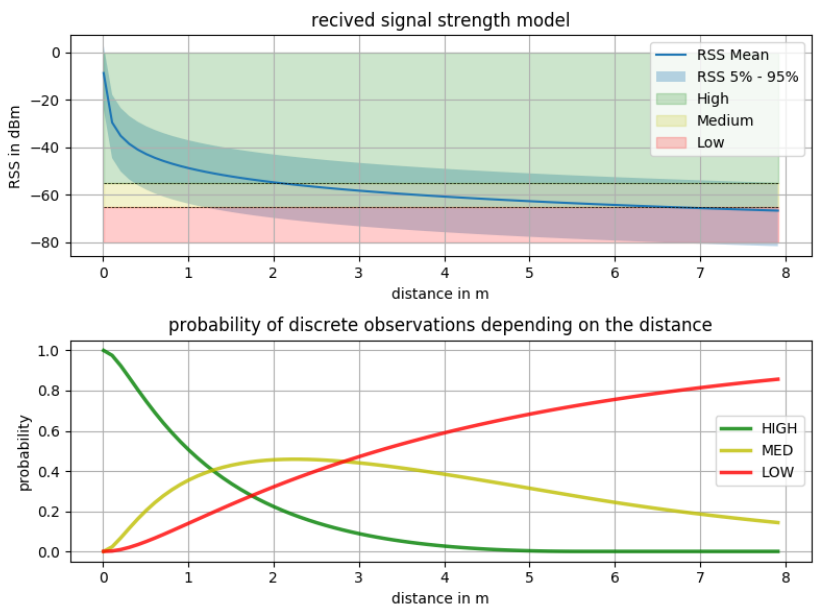
- High $\rightarrow$ signal strength $\geq -55dBm$
- Medium $\rightarrow$ $-55dBm \geq$ signal strength $\geq -65dBm$
- Low $\rightarrow$ $-65dBm \geq$ signal strength

These thresholds result in the likelihoods of observing a specific discrete value shown in Figure 3.5.

Furthermore, we need to define a collection of actions for the dec-POMDP model ahead of time. This collection of actions is based on a predefined movement graph, where each node represents a real-world location. For each connection between locations on this graph, we define two actions. One that makes a robot move from, for example, location one to location two and one that makes a robot move from location two to location one. These actions can then be used by the NPGI algorithm while generating policies to decide which node $q \in Q$ can correspond to which action depending on its predecessor. This way we end up with a policy that moves a robot along the predefined movement graph. We define this movement graph separately for each experiment.

After defining the parameters we need to model our signal source localization task using

Figure 3.5 Probability of a discrete value occurring based on the model introduced in Section 4.1



dec-POMDP, we also need to define the parameters that are important for the NPGI algorithm. They have various impacts on the performance of the algorithm and the generated policies. They are defined separately for each experiment as well and can be explained as follows [Lauri, Pajarinen, and Peters 2020]:

- event horizon
  The specified event horizon determines how many time steps each policy takes. Since an information gathering task has no clear goal state, we need to specify a maximum number of actions an agent may take. The defined event horizon plus one directly corresponds to the maximum height of a policy graph. For example, if we run the planning algorithm with an event horizon of three, it generates a directed graph with a height of four where all nodes for the first three layers define an action, and all nodes in the fourth layer are simply marked with an end tag.

- policy width
  The policy width specifies the maximum width for all policy graphs. For example, a width of two would result in policies with a maximum of two reachable nodes at each time step.

- improvement steps
  The specified number of improvement steps determines through how many iterations the policies should be improved. If the joint policy value converges before this limit is reached, the planning process completes sooner. A higher number leads to more optimal policies, but also linearly increases the planning time.

- number of particles
  A high number of particles used during the forward pass of the planning algorithm increases the accuracy of the policy value calculation process. However, it also increases the planning time.

- number of rollouts
  The number of rollouts determines how often the reward of a local policy has to be calculated during each backward pass to determine the expected sum of rewards for each agent i that starts at node q_i^t where t is the current time step.

- number of particles per rollout
  The number of particles that are used during each rollout for the backward pass. A higher number increases the accuracy of the reward function but also increases planning time.

- number of agents
  Currently, the number of agents is limited to precisely two agents due to the implementation of the NPGI algorithm we use.

For the simulated source localization experiment Lauri, Pajarinen and Peters used $T = 3, 4, .., 7$ as planning horizon, width 2 or 3, 10000 particles, 100 rollouts, 50 improvement steps and two agents [Lauri, Pajarinen, and Peters 2020].

3.3 Software Architecture

This section explains the general software architecture used in the practical part of this thesis. It describes the fundamentals of the Robot Operating System (ROS) we use to implement our system and introduces the different components we develop to solve the signal source localization task. Furthermore, this section explains how the various components communicate and how a user can interact with the proposed system.

We implement the software required for this thesis using the Robot Operating System (ROS). ROS is a framework specifically designed for the development of robotics software. It can be used to create applications for a single robot and peer to peer networks of hosts [Quigley et al. 2009]. To discuss an application developed using ROS, it is essential to first introduce the terminology of the framework as defined in 'ROS: an open-source Robot Operating System' [Quigley et al. 2009].

Nodes are processes within a ROS application. They are supposed to logically encase a part of the system that could theoretically be run independently of all other nodes. In our case, one node, for example, could be responsible for the execution of policies.

Topics are a way for those nodes to communicate with each other. A node may subscribe to a topic to receive all the information other nodes publish to it. Publishers and subscribers, however, are unaware of each other. A node that publishes information to a topic does not know if any other node receives this information, and a node that subscribes to a topic does not know whether any other node is publishing information there.

Messages are the data structures that nodes can use to send information over topics. They can be custom-defined to contain standard primitive types, arrays, or other message types.

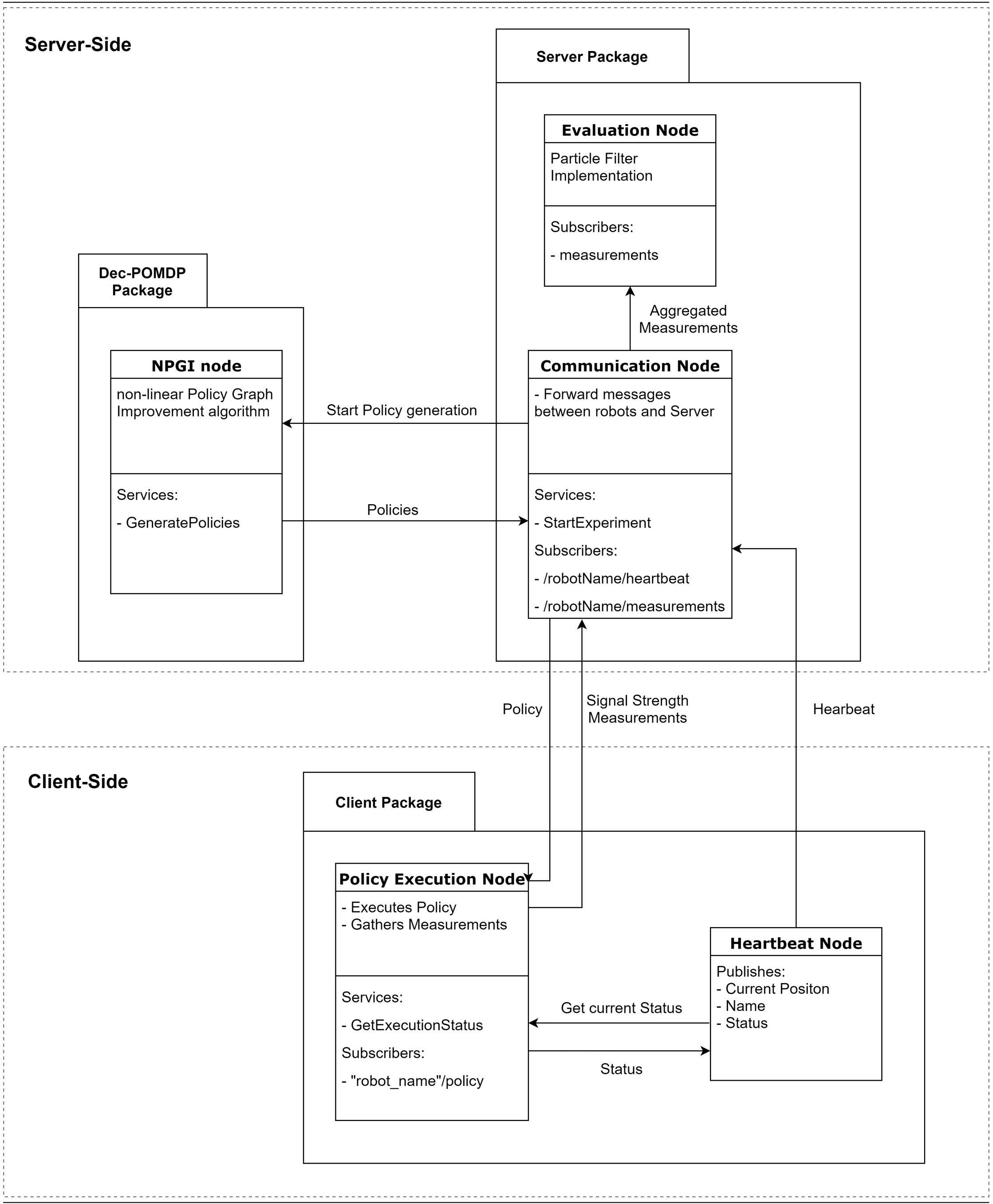
Services , in contrast to topics, can be used for synchronous transactions between nodes. A node that offers a service can be called by another node with a request and returns an appropriate response. This behavior allows for direct communication between two nodes.

To solve the proposed signal source localization task using dec-POMDP, we design and implement a system including three main components. First of all, a client-side module that enables the robots to execute a local policy. Second, a module that includes the NPGI algorithm, as introduced by Lauri et al. [Lauri, Pajarinen, and Peters 2020]. Third, a central module that evaluates taken measurements and establishes the communication between the client-side nodes and server-side nodes. Furthermore, we introduce an alternative way to start the client-side module, which, instead of executing a local policy, randomly moves the robot along a given movement graph.

We implement each of the three main components as a ROS package, which is just a directory containing a file with a description of the package and its dependencies. For each of the packages, we also define one or more launch files. A launch file defines instructions to instantiate one or multiple nodes with a predefined configuration. We can use the roslaunch tool [Quigley et al. 2009] to execute a launch file.

The final software architecture we end up with for the proposed system is shown in Figure 3.6. It displays all of the packages and shows how they communicate with each other.

Figure 3.6 Software architecture of the introduced system



3.3.1 Dec-POMDP Package

The dec-POMDP package generates Policies for the robots. It utilizes an implementation[2] of the NPGI algorithm introduced by Lauri et al. [Lauri, Pajarinen, and Peters 2020]. To use this algorithm in our ROS environment, we encapsulate it in a ROS node. The introduced node offers a service that can be called by other nodes to trigger the policy generation. A request to this service contains all the planning algorithm parameters, as defined in section 3.2. The response of this service then contains the two generated policies.

To enable the algorithm to work, we also need to define the actions each robot can take. Since an action always consists of moving from the current location to a neighboring location and taking a measurement, we need to define a movement graph. We specify a set of locations and a set of allowed moves, where the set of allowed moves defines all neighboring locations for each location. The configuration file defining both of these sets can be adjusted for each experiment individually.

3.3.2 Client Package

The system's client-side package is the part we deploy on each TurtleBot2 that we want to use for an experiment. It consists of three main parts. The first part of the client package is its dependencies. To provide navigation and localization capabilities, we utilize the move_base and amcl packages introduced in section 2.2. Additionally, we use the `tams_turtlebot`[3] package developed by the TAMS-Group to start the nodes that are necessary to interact with the TurtleBot2 hardware.

Second, each client has a node that enables it to execute a given policy. We design this node specifically for the source localization task. The execution process begins with the policy graph's starting node, performs the defined actions, and progresses according to the perceived observations. For each action, the execution node performs, the robot has to move to the specified location. As soon as the robot has reached the action's goal location, the policy execution node takes a specified number of signal strength measurements. Afterward, it calculates the mean value of the taken measurements and converts it to a discrete value, as described in section 3.2. The policy execution node then progresses to the next node on the policy graph based on this discrete value. We define the specific amount of measurements taken in each location through a configuration file similar to the NPGI algorithm parameters. Once this ROS node executed the policy, it sends all taken measurements to the server for evaluation.

Finally, each robot has a separate 'heartbeat' node that publishes messages to a topic at

[2] The implementation we use was developed by M.Lauri and is available under `https://github.com/laurimi/npgi` Accessed: 13.08.2020

[3] A ROS package specifically designed to start the TurtleBot2 with the hardware configuration employed by the TAMS-Group. Additional information can be found under `https://github.com/TAMS-Group/tams_turtlebot` Accessed: 13.08.2020

a regular interval. Each of these messages contains the robot's name, position, and status. The central server subscribes to these topics published by all robots that are part of the experiment to keep track of them. It uses the name to identify a specific robot and the position as a starting location for the planning process. The status part of the message specifies the current status of the robot's policy execution. The four possible status are listed below:

- `MOVING`
  The robot is currently moving in response to an action issued during policy execution.
- `MEASURING`
  The robot is currently taking signal strength measurements.
- `IDLE`
  The robot is currently waiting for a new policy to execute.
- `ERROR`
  The robot has encountered an ERROR during policy execution. This status could appear if the policy execution node is, for some reason, unreachable or if either the measuring or the moving process has failed.

The 'heartbeat' node has to communicate directly with the policy execution node over a service to get this status. Therefore, one might suggest combining these two functionalities into a single node, but we want the 'heartbeat' node to keep sending messages to the server even when the policy execution node is for some reason unavailable or in a faulty state.
The heartbeat messages technically do not have to be sent regularly for the experiment. The client-side system can complete an entire policy without any communication to the server during the execution. The central server only uses these messages to determine which Robots are available and where they are to start the planning process. Nevertheless, the information where each robot is at each point in time and what their current status is, is very useful to us as an introspection method.

The Random Movement Alternative to the regular policy execution utilizes the same essential dependencies and the 'heartbeat' node. The only difference is that this alternative does not offer a node that can execute a policy. Instead, it has a node that randomly chooses goal positions from a given movement graph and collects measurements at each location the robot visits. This alternative client setup can be launched through a different launch file and offers the ability to start an experiment with the random movement enabled.

3.3.3 Server Package

The server package consists of two major parts, an evaluation node, and a communication node. The evaluation node is responsible for evaluating the signal strengths that were measured by the deployed robots. The communication node handles all communication between the robots and all nodes running on the server, like the dec-POMDP planning node and the evaluation node. This section introduces both of these nodes in more detail.

Evaluation Node

The evaluation part of the system is a ROS node running on the central server, just like the communication node and the dec-POMDP planning node. Its purpose is to evaluate the aggregated measurements from all robots and give an estimate of the searched Wi-Fi access points' true location. To achieve this, the node makes use of the particle filter, as introduced in section 2.1. The final belief of this particle filter is the overall result of the information gathering task and to construct it this node iteratively evaluates all taken measurements at the end of the experiment. To get the probability of the signal source location beeing in the exact location of a particle based on a signal strength measurement this node utilizes the RSS model introduced in 3.1.

The number of particles the particle filter uses is set through an external configuration file.

Furthermore, the module also visualizes the result using the RVIZ [4] tool. We represent the particle filters belief using color-coded squares, where a red-colored square represents a particle with the lowest weight of all particles. For all other particles, the green value of the color increases, and the red value decreases according to the particles' weight, so that the particle with the highest weight is completely green. This way, we get a human interpretable visualization of the belief state.

Communication Node

To better organize communication between the client-side nodes and all server-side nodes, we introduce a communication node. Its main purpose is to keep track of all available robots and forward communication between server and client nodes.
Since the dec-POMDP planning algorithm always needs to know how many robots are available for the planning process, a single cannot ask for a new policy by itself. Therefore, the architecture implemented for this thesis is designed with this communication node as a central part. It subscribes to each robot's heartbeat topic to receive their status information and keep track of all usable agents. When we want to conduct an experiment, we can instruct this central node to start the process over a service it provides. Calling this service triggers the node to start the policy generation using all agents that are currently available. As soon as the dec-POMDP module returns the policies, they get distributed to the robots accordingly.

When all of the robots have finished their policy execution, they send all taken measurements to the communication node, which aggregates the gathered measurements into a single ordered data set and sends this set to the evaluation node. The measurements in the data set are in the same order in which they were taken by the robots.

[4] RVIZ is a visualization tool for ROS. More information can be found under `http://wiki.ros.org/rviz` Accessed: 13.08.2020

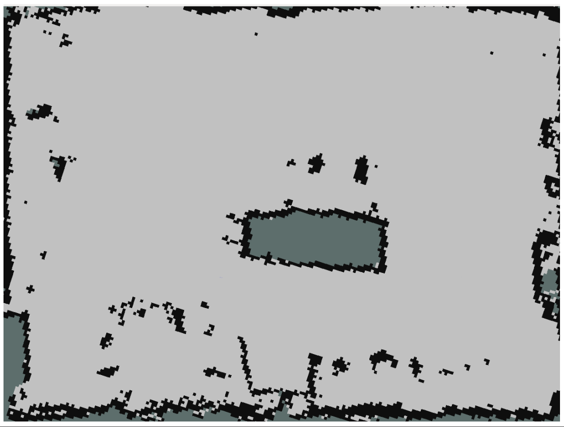
4 Experimental Setup

This chapter introduces the environment and the hardware we use for all experiments presented in this thesis. It presents a map of the room in which we conduct all our experiments and specifies the exact hardware both robots are equipped with. Furthermore we also introduce the WiFi router that we want to localize.

4.1 The Map

As explained in the section 2.2 Robot Navigation, each robot needs a predefined map of its environment to enable navigation. Since all experiments are conducted at the laboratory of the Technical Aspects of Multimodal Systems (TAMS) work group, we are using a map of their laboratory. While the original map [1] displays the entire floor of the building, our experiments are only conducted in the robotics laboratory shown in Figure 4.1. The room in which we conduct all our experiments is about $10m$ long and $7m$ wide meters. The obstacles that are marked within the room are tables, chairs and shelves near the walls and a couch in the middle.

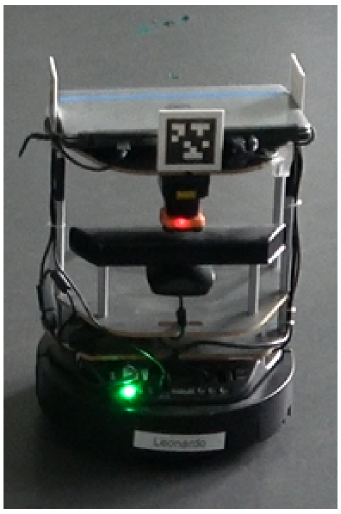
Figure 4.1 The TAMS work group robotic laboratory (source: `https://github.com/TAMS-Group/tams_turtlebot/blob/master/tams_turtlebot_navigation/maps/tams_navigation.pgm`)



[1]Full Map supplied by the TAMS Group and available under `https://github.com/TAMS-Group/tams_turtlebot/blob/master/tams_turtlebot_navigation/maps/tams_navigation.pgm`

4.2 The Robots

Figure 4.2 The backside of one the TurtleBot2s that are used to conduct experiments for this thesis



The robots that are used for all of the experiments conducted throughout this thesis are two TurtleBot2s. One of the two robots used can be seen in Figure 4.2. Each of the robots is equipped with a Kobuki Base[2] that enables the robot to move around. Mounted on the mobile base is a front-facing Kinect[3] camera and a back-facing laser range finder. The Kinect camera has a built in depth sensor that can be used to estimate the distance to obstacles in a field of view with a horizontal angle of $57°$, and a vertical angle of $43°$ [Andersen et al. 2012]. The laser range finder mounted on the robot displayed in Figure 4.2 is a Hokuyo UTM-30LX 2D laser range finder[4]. This laser rangefinder has a detection range of $270°$ and can detect obstacles at distances between $0.1m$ and $30m$ or with lower accuracy $60m$ [HOKUYO AUTOMATIC CO. 2012]. The other robot we use is equipped with a different model, the Hokuyo URG-04LX laser range finder [5]. This device has a smaller detection range of only $240°$ and only detects obstacles reliably at distances of up to four meters [HOKUYO AUTOMATIC CO. 2009]. However, the difference between the two laser range finders does not have any significant impact on the experiments we conduct since even the one with the low detection range can detect at least one wall at all times on the map we are using for the proposed experiments. The robot uses both the Kinect camera and the laser range finder for self-localization and obstacle avoidance. Furthermore, a laptop is mounted on the robot and

[2]Kobuki is a Product of YUJIN ROBOT Co, Ltd more information can be found under `http://kobuki.yujinrobot.com/` Accessed: 06.08.

[3]A Camera originally developed for the Xbox 360 game console by Microsoft `https://www.microsoft.com/` Accessed: 06.08.2020

[4]The Hokuyo UTM-30LX is a scanning laser range finder developed for robots by HOKUYO AUTOMATIC CO., LTD `https://www.hokuyo-aut.jp/search/single.php?serial=169` Accessed: 06.08.2020

[5]The Hokuyo URG-04LX is a scanning laser range finder developed for robots by HOKUYO AUTOMATIC CO., LTD `https://www.hokuyo-aut.jp/search/single.php?serial=165` Accessed: 14.08.2020

connected to the mobile base, the laser range finder, and the Kinect camera. It functions as the control unit of the robot and is equipped with a Wi-Fi chip.

4.3 The WLAN Signal Source

Throughout this thesis we use a Wi-Fi Router as the WLAN signal source. The exact hardware we are employing as a Wi-Fi router is a 'Huawei P20 Lite' smartphone. It is configured as a mobile Wi-Fi hotspot on a $2.4\ Ghz$ frequency. There are two main reasons behind choosing a smartphone as the signal source. One of the reasons is that a smartphone can be placed in and moved to arbitrary locations without any additional effort since it does not require an external power source like a regular Wi-Fi router might. Another reason why we are using a smartphone for our experiment is that it is possible to see significant differences in received signal strength inside our experimental environment, as we have shown in section 3.1. The difference in received signal strength between one and six meters is advantageous to us since we need an accurate approximation of the distance between the signal source location and the location where a measurement was taken. If the differences between measurements at distances of one to six meters were unrecognizable, it would be impossible to approximate the signal source location within the robotic lab where we conduct all experiments for this thesis.

5 Results and Discussion

For this thesis, we conduct several experiments to show that dec-POMDP can successfully be applied to a signal source localization task. This chapter gives an overview of the conducted experiments and compares them to each other. We also introduce a baseline for the proposed source localization task, by conducting experiments with the robots moving randomly between possible locations. Additionally to the physical runs, we conduct a more significant number of simulated experiments using both the random movement and the dec-POMDP approach. This enables us to compare the average results of both approaches, and we can show that the dec-POMDP approach yields similar or slightly better results than the random movement strategy. To put this thesis into better perspective, this chapter also discusses challenges that we encountered during the experimental runs.

5.1 Real-world Experiments

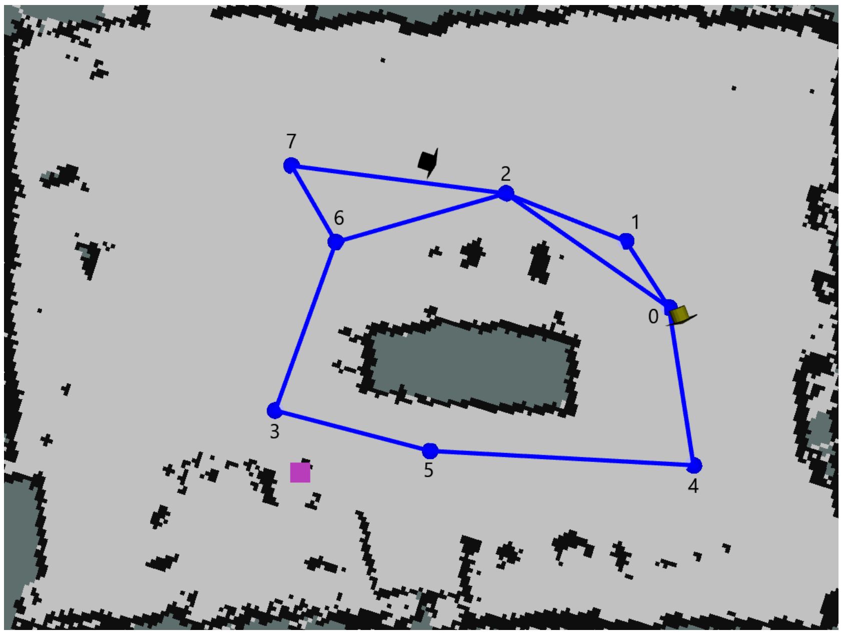
Before discussing our implementation's efficiency, we demonstrate that our approach is working as intended and can be used to solve the proposed signal source localization task. To achieve this, we present one full experimental run and discuss its results in this section. Furthermore, we present the results of other physical executions to demonstrate the capabilities of the introduced system further.

The physical experiment we use to demonstrate our approach's exact functionality is conducted as follows:

First of all, the two robots and the mobile phone, which acts as the Wi-Fi router, are placed at arbitrary locations in the room. The two robots both have to be localized manually to enable their navigation. For evaluation purposes, the Wi-Fi router location is documented as well. Furthermore, the movement graph for the dec-POMDP planning algorithm has to be defined ahead of time as well. For this specific experiment, eight predefined locations are used to form the movement graph shown in Figure 5.1.

Each of the blue dots in Figure 5.1 represents a location a robot could visit and the blue lines indicate which locations the robot could move to from its current position. The real world location of the Wi-Fi router is represented using a purple square and the real world locations of the two robots are depicted as short arrows, where the orientation of the arrow indicates the orientation of the corresponding robot. Figure 5.1 shows both robots in their starting locations right after self localization but before executing any actions to solve the source localization task. For the planning algorithm the movement graph location nearest to

Figure 5.1 The movement graph, the starting locations of both robots and the location of the Wi-Fi router for the experiment described in Section 5.1



a robot is considered its starting location. However, that does not necessarily mean the first action is to visit that location. Instead, the best action according to the planning algorithm might be moving to an adjacent location.

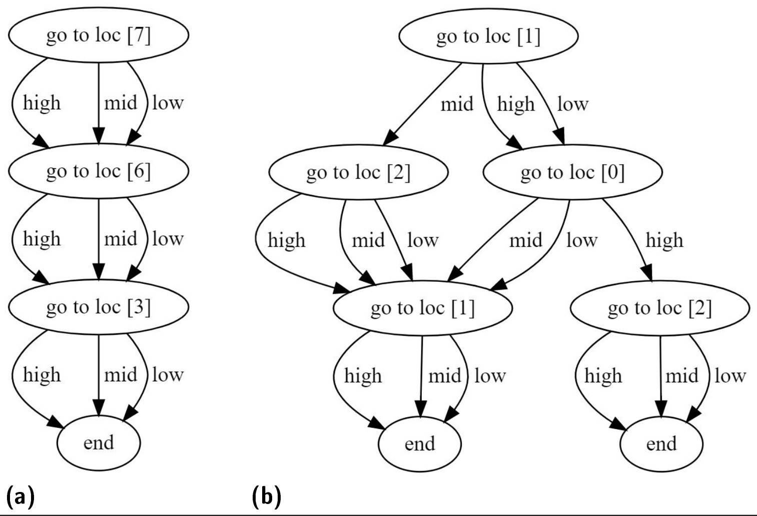
After completing this preparation work the NPGI algorithm is run to construct optimized local policies for the two robots. For the planning process itself we use the actual parameters shown in Table 5.1. We introduced the parameters for the planning algorithm in section 3.2 and the client and evaluation parameters in section 3.3.

Table 5.1 Experiment parameters for the experiment described in Section 5.1

Planning algorithm parameters			
Event Horizon	3	Policy Width	2
Improvement Steps	4	Rollouts	100
Particles for Planning	1000	Particles for each Rollout	100
Client and Evaluation parameters			
Measurements per action	5	Particles for Evaluation	1500

The planning algorithm generates local policies for the two used robots based on these parameters. Figure 5.2 displays the two policies that were generated for this specific experiment. Each node in the displayed graph represents one state the robot could be in and the outgoing edges of a node define how a robot should proceed after executing the action defined by the current node.

Figure 5.2 The two local policies that were used for the experiment presented in section 5.1. **(a)** Policy for Agent 0 / donny (Turtlebot) **(b)** Policy for Agent 1 / Leo (Turtlebot)

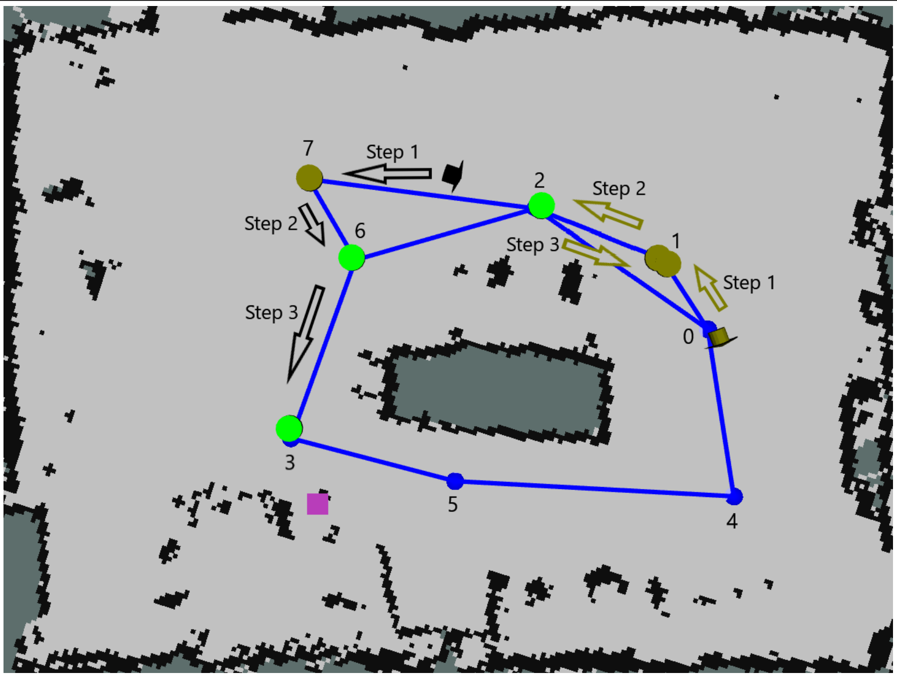


An action consists of moving to a given location and taking a measurement. It is always noted as 'go to loc' and the goal locations number. The action 'go to loc [7]' for example means move to location 7 and take a measurement there. The location numbers all correspond to a specific location on the movement graph shown in Figure 5.1. The policy execution always starts at the starting node specified by the planning algorithm. In this case the starting nodes are the only top level nodes of both graphs respectively. The next node is picked based on the observation that was made during the previous action.

The observations for this experiment are generated from the mean of 5 signal strength measurements. Which discrete observation this mean value is translated to is defined through the intervals introduced in section 3.2. Every signal strength value higher than $-55dBm$ translates to $high$ and every value below $-65dBm$ gets interpreted as a low value. Everything in between is viewed as mid. The discrete values that were observed throughout this experiment are displayed in Figure 5.3. The colored dots on top of the blue dots, which mark the locations, depict the discrete values that were used to decide on the next location to travel to. A green circle marks a 'high', a dark orange circle marks a 'mid' and a red circle marks a 'low' observation.

Figure 5.3 also depicts the robots as short filled arrows in their starting locations and the movements they conducted throughout the experiment as empty arrows in the same color. Each of the empty arrows represents the action the robot took at the specified time step.

Figure 5.3 Discrete signal strength values observed and actions taken by both robots at the given time steps



As an additional explanation to Figure 5.3 we also summarize the process of this experiment as follows:

- Actions of Agent 0 at the given time step, represented by black arrows in Figure 5.3
  1. Action: Move to Location 7; Observation 'mid'
  2. Action: Move to Location 6; Observation 'high'
  3. Action: Move to Location 3; Observation 'high'
- Actions of Agent 1 at the given time step, represented by yellow arrows in Figure 5.3
  1. Action: Move to Location 1; Observation 'mid'
  2. Action: Move to Location 2; Observation 'high'
  3. Action: Move to Location 1; Observation 'mid'

This successful execution of the given policies demonstrates that the client-side module introduced in section 3.3.2 does work as intended. It can read a given policy, execute the defined actions, and proceed through the policy graph according to the perceived observations.

During the policy execution process all 5 measurements collected in one location are also directly transmitted to the central evaluation module. Typically, we would wait for both of the robots to complete the policy execution and then evaluate the gathered measurements to estimate the Wi-Fi routers location. But to show how exactly the system is working, we send the gathered measurements to the central evaluation unit after each action. This way it is easier to understand what impact each action has and we can directly verify that the policy is executed correctly.

Figure 5.4a shows the initial belief of the evaluation module. Each of the small squares represents one particle of the particle filter we use for evaluation. A brighter green value in the particles color represents a higher weight, while a brighter red value represents a lower weight. Since the particle filter displayed in this graphic was just initialized, the weight of every particle is the same. In order to demonstrate that the system introduced throughout this thesis can work, the belief of this particle filter should represent a good estimation of the Wi-Fi routers real world location after the experiment was conducted.

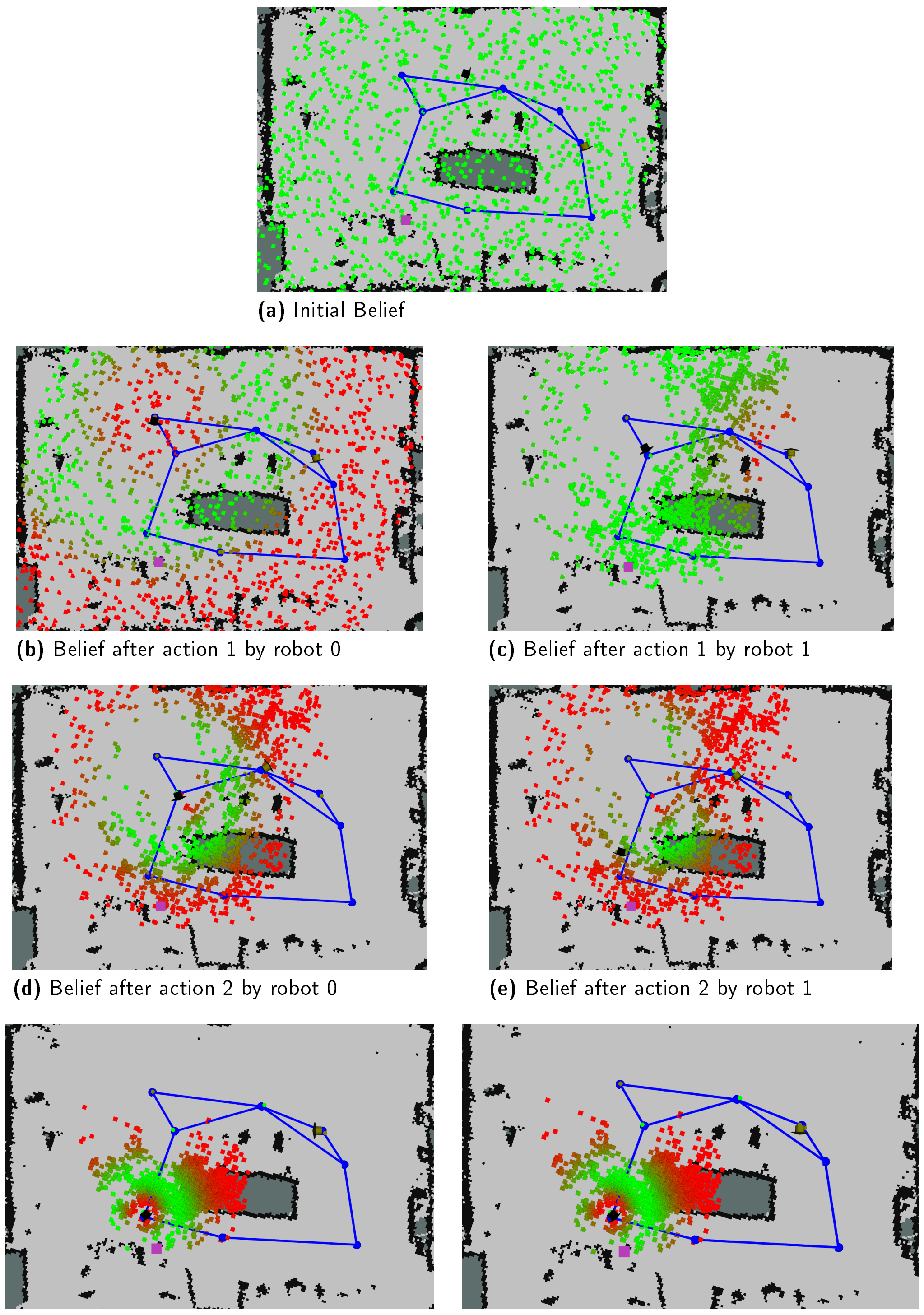
The six Figures 5.4b - 5.4g present the beliefs of the particle filter after each of the three actions per robot. The five measurements we take with every action are fed to the particle filter algorithm one by one, but for simplicity, we only show the belief after all five have been evaluated.

Finally, after all measurements were collected and fed to the particle filter we end up with a belief that represents an estimation of the signal source location. The final belief of the particle filter can be seen in Figure 5.4g.
The shown particle filter does represent a close estimation of the real signal source location here displayed as a purple square. To generate a metric that enables us to compare this result with other experiments we are using a weighted root mean squared error function (RMSE) displayed in equation 5.1. The function takes the real world location and compares it to all particles to calculate the error in meters, while also taking into account the weight of each particle.

$$Weighted\_RMSE = \sqrt{\frac{\Sigma_{i=1}^{n} w_i d(loc_{real}, loc_i)^2}{\Sigma_{i=1}^{n} w_i}} \tag{5.1}$$

We can apply this function to our resulting particle filter to get a quantitative score for its accuracy. The final weighted RMSE for this specific experiment is $1.41m$. To put this value into perspective we can also look at the weighted RMSE of the particle filter after each evaluated measurement. The resulting values are depicted in Figure 5.5.

Figure 5.4 The particle filter progression for the experiment described in section 5.1



(a) Initial Belief

(b) Belief after action 1 by robot 0

(c) Belief after action 1 by robot 1

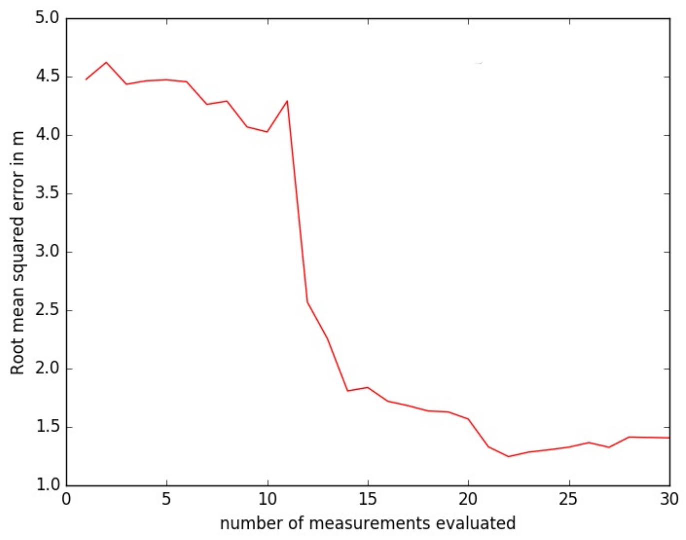
(d) Belief after action 2 by robot 0

(e) Belief after action 2 by robot 1

(f) Belief after action 3 by robot 0

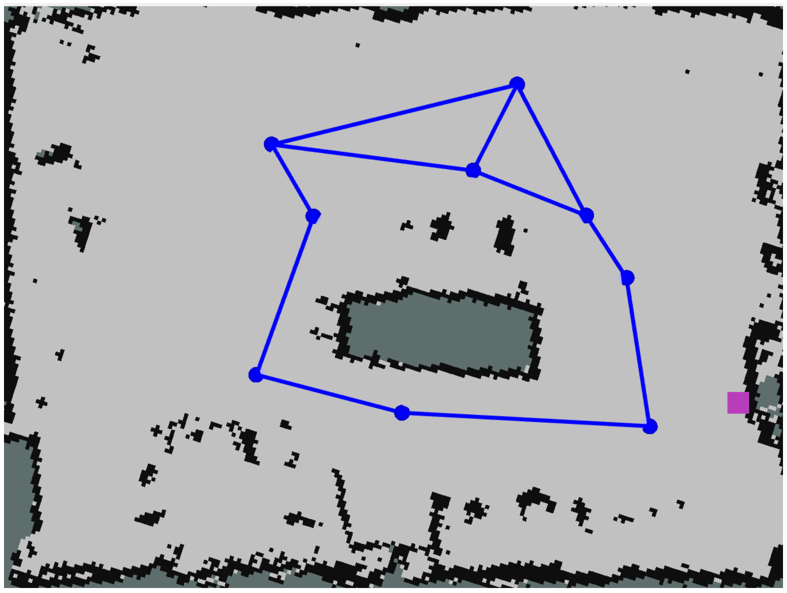
(g) Belief after action 3 by robot 1 / Final Belief

Figure 5.5 The weighted root mean squared error in m after each evaluation step during the described experiment



In addition to the above described experiment we can also consider other physical experiments. Figure 5.6 presents another movement graph that is slightly different to the first experiment we described. The Wi-Fi router is also placed at a different location.

Figure 5.6 The movement graph used for all experiments recorded to compare the dec-POMDP to the random movement approach



Based on this new configuration we conduct three different physical experiments with the parameters as shown in Table 5.2. The three experiments differ only in the number of measurements taken per action. The first experiment is conducted with 5 measurements per action, the second one with 10 and the last one with 15.

Table 5.2 Experiment parameters for the experiment described in Section 5.1

Planning algorithm parameters			
Event Horizon	4	Policy Width	3
Improvement Steps	8	Rollouts	100
Particles for Planning	1000	Particles for each Rollout	100
Client and Evaluation parameters			
Measurements per action	5 / 10 / 15	Particles for Evaluation	1500

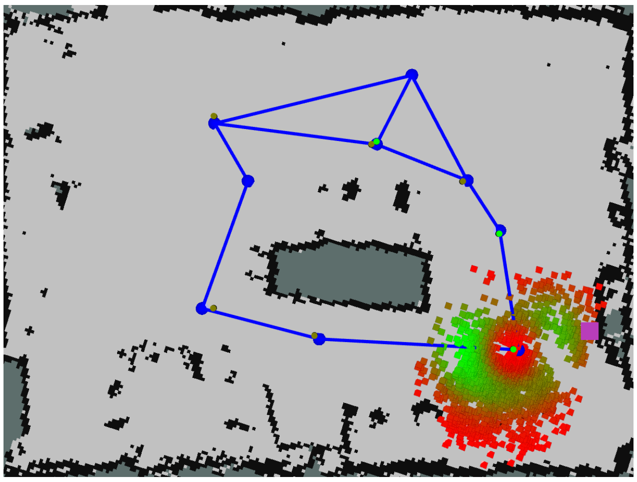
The comparison between these three experiments allows us to highlight the impact of a location change in contrast to multiple measurements taken in a single location. It also helps us to further demonstrate the capabilities of the introduced system.

Figure 5.7 presents the resulting particle filter beliefs for the three conducted experiments. All three of the depicted beliefs indicate the general area of where the Wi-Fi router was placed, which demonstrates that the system is working as intended. For this group of experiments, however, it is more interesting to consider the direct comparison based on the weighted root mean squared error metric introduced earlier.

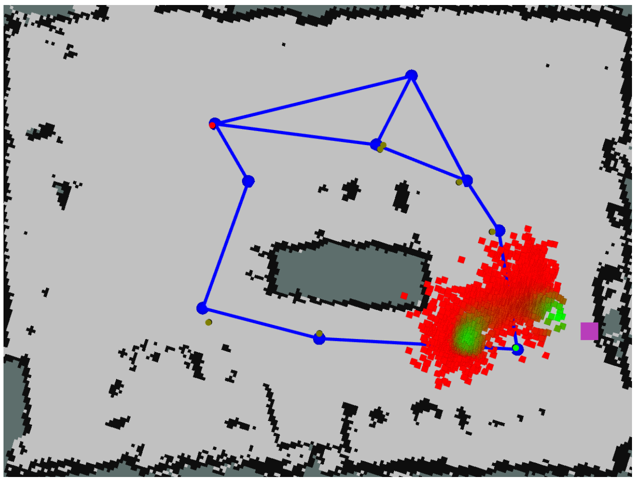
If we plot the weighted RMSE over all evaluated measurements for each of the experiments, we get the graph displayed in Figure 5.8. From this quantitative comparison we can see that the resulting beliefs are indeed all similarly accurate despite the difference in evaluated measurements. The weighted RMSE after the final evaluation step is $\sim 1.627m$ for the experiment with 15 measurements per action, ~ 1.640 for the experiment with 10 and ~ 1.644 for the experiment with 5.

Another interesting aspect of the presented results is that the error value converges slower depending on the amount of measurements taken every action. For example the evaluation error for the experiment with only 5 measurements per action is already down to ~ 1.657 after only evaluating 26 measurements. Meanwhile, the error of the experiment with 15 measurements per action first reaches a value under $2m$ after evaluating 60 measurements. This difference is to be expected since taking a lot of measurements in only one or two locations does not necessarily allow us to determine the direction from where the signal originates.

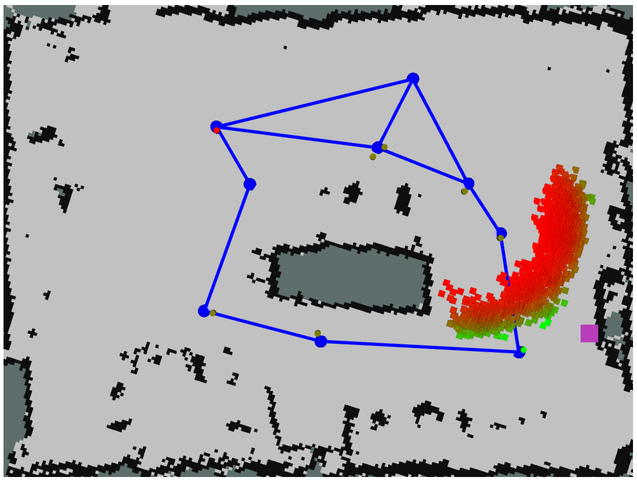
Figure 5.7 The resulting particle filter beliefs for three physical experiments.



(a) 5 measurements per action

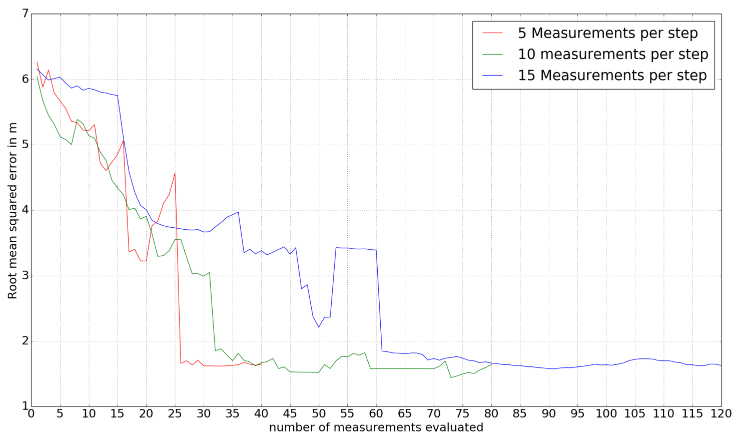


(b) 10 measurements per action



(c) 15 measurements per action

Figure 5.8 The weighted RMSE for a group of three experiments that were conducted with the same conditions and parameters except for the number of measurements taken per action



5.2 Challenges

While conducting experiments, we encounter a few challenges that are rather important to this thesis even though not being critical to its results. This section discusses these challenges to review the results of this thesis from a different perspective. It reviews the impact of each challenge and considers the limitations they impose on solving an information gathering task using the dec-POMDP planning algorithm introduced by Lauri et al. [Lauri, Pajarinen, and Peters 2020].

5.2.1 Collision Avoidance

The first challenge that arises from conducting an experiment is avoiding any collisions between the robots.
Since the robots can not rely on communication between each other during policy execution, they cannot avoid collisions by transmitting their current positions to other robots. Furthermore, each action an agent executes to complete a policy may take an undefined amount of time. Therefore, not every agent will be at the same time step at the same real-time, which makes avoiding collisions by modifying the policy generation rather difficult. If that were not the case, it would have been easy to introduce a new rule to the planning algorithm that requires the generated policies to never include the same action for different agents at the same time step.

For a solution to this challenge we need to consider the two possible causes of collisions between multiple robots.
The first possible cause is two robots crossing paths. When two robots are moving at the same time it is possible that they collide with each other, because the path planned by the

navigation module does not account for moving obstacles. To completely solve this problem it would be possible to implement a concept called 'Reciprocal Velocity Obstacles' [van den Berg, Ming Lin, and Manocha 2008]. This concept works by monitoring the velocity of moving obstacles to avoid them while assuming that other robots employ the same behavior [van den Berg, Ming Lin, and Manocha 2008]. However, implementing this concept would go beyond the scope of this thesis. Instead, we can prevent such collisions by stopping the movement of one of the two robots manually. If one of the robots is stationary, the navigation module of the other robot can generate a path around. Once the possibility of a collision is averted the other robot can continue moving to its goal as well. This might increase the duration it takes to conduct an experiment and it means that an experiment requires constant human supervision. But since signal strength measurements are only taken whenever a robot has reached its destination, manual intervention while both robots are moving does not obstruct the measurement process. Therefore the result stays the same with or without intervention. For all of the physical experiments presented in this thesis, intervention never was necessary.

A far more interesting possible cause of collision is if one robot tries to reach a location already occupied by another one. In this case, the move_base package we are using for navigation already provides the necessary collision avoidance, since the other robot is stationary while taking measurements. However, if the robot occupies the location for a more extended amount of time to collect the specified number of signal strength values, the waiting robot would eventually consider its goal unreachable. To prevent this, we can resend the current goal to the move_base package whenever it is aborted.

This approach does lead to longer execution times. But it does not change the overall result of an experiment, as long as we still evaluate the taken measurements in the order in which they were meant to be taken by the planning algorithm.

5.2.2 Time Consumption of Physical Experiments

Another interesting challenge to this thesis is the time it takes to conduct a physical experiment properly.
To come to a meaningful conclusion on the efficiency of using dec-POMDP on an information gathering task, it is necessary to conduct a reasonable number of experiments. The high variance in the received signal strength obviously does have an impact on the final experiment result. In one experiment we could have a lot of measurements that fit the RSS-model we use to construct the particle filter perfectly. The next experiment could have a lot of deviating measurements. These two experiments would result in very different estimations of the signal source location even if none of the other parameters changed. Therefore, we need to conduct a large number of experiments, to compare the introduced system to a random movement baseline.

The necessity for a large amount of recorded experiments, however, leads us to a time problem. First, each of the used robots needs to localize itself. This does require some time, but is only necessary for the first experiment after each charging break. Second, the planning algorithm requires some time that is very dependent on the number of improvement steps and the other configuration parameters. This might also change slightly depending on the server the planning algorithm is running on.

The next time consuming factor of each experiment is the time a robot needs to execute a policy. This obviously depends on the event horizon, which defines the number of actions each robot has to take. The time it takes to execute a single action depends on the travel time to its next location and the number of measurements that the robot has to take. The time consumed by taking measurements can be approximated as 4-5 secs multiplied by the number of measurements that have to be taken, as we have shown in section 2.4. For example it would take approximately 1 min to 1 min and 15 secs to take 15 measurements. The time it takes to travel to the next location on the other hand, can not be quantified as easily. Collision avoidance between each robot and their environment and between the robots themselves adds an unknown factor to the time it would take for a robot to travel the line of sight distance. If, for example, two robots were to try and visit the same location at the same time, one of the robots would have to wait for the other to record its measurements and move away to the next location before taking its place.

Because of these various time consuming factors, it is only possible to conduct eight to ten physical experiments within a full day of work. To collect larger amounts of data we can run experiments in simulation.

An experiment run in simulation does not require constant human supervision and also requires less time. In simulation the robots do not have to be manually localized and also do not require any time to move to a given location. Furthermore, we can directly sample from the RSS model introduced in Section 3.1 and ,therefore, save the time it would normally take to gather measurements.

5.3 Simulated experiments

In order to show that it is possible to use a dec-POMDP planning algorithm to solve a signal source localization task, this section compares the results of multiple simulated experiments to a baseline. The baseline for the introduced problem is drawn using a random movement approach. Instead of following a preplanned policy, robots choose the next location to move to at random in experiments using the random movement approach.

To ensure that this comparison does represent the general case, it is essential to record and evaluate multiple experiments using both the dec-POMDP and the random movement approach. Consequently, we need to make use of simulation because physical experiments consume too much time.

Recording multiple simulated experiments in a row without user input, however, requires slight modifications to the overall system. These modifications do not have any influence on the result of one single experiment. They only define the behavior of the robots between experiments where they would usually be controlled by the user and a way to simulate the process of taking signal strength measurements

The main addition to the System is a redistribution of the robots after each experiment. While previously, each experiment was prepared independently by the user, now multiple are executed consecutively. Therefore, the preparation part where the user decides on an arbitrary starting position for the two robots is dropped. But since the starting locations do impact the planning algorithm and the random movement approach, it is necessary to clearly define how the starting locations for each of the experiments are chosen if not by a user. The three available options are:

1. The robots start the next experiment in the exact locations where they ended up in the previous experiment. This approach would not require any additional implementations, but the manually chosen starting positions of the first experiment do have an impact on all ensuing experiments. However, that would be unwanted behavior and is the reason we are not using this approach.

2. A specific starting position could be defined for each robot. After each experiment, the robots would return to their individual starting locations before starting the next experiment. With this approach, the chosen starting locations have a significant impact on all the results. Therefore, this behavior is not a reasonable approach either.

3. Each of the robots moves to a randomly chosen starting location before an experiment is conducted. That means the starting locations for each experiment are independent and identically distributed random variables. This ensures that the impact of the starting location on a single experiment does not influence our overall average results when evaluating a bigger set of recorded experiments. As a result, this is the option we are implementing.

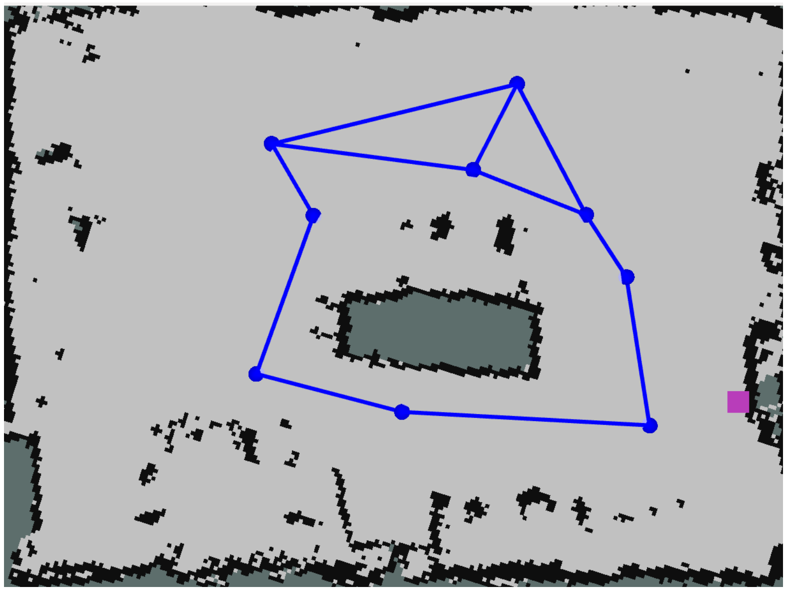
Furthermore, we make a slight change to the evaluation process of the taken measurements. Previously, for the physical experiments introduced earlier, the taken measurements were sent to the central evaluation module after each action. Now the measurements are all gathered on the robot first and are only sent to the central evaluation module after the policy execution has finished. This change is made since the whole reason for modeling this signal source localization task as a dec-POMDP problem is that communication between the central planning and evaluation unit and the agents is impossible during policy execution.

At which point in time the measurements are evaluated does not have an impact on the resulting particle filter, as long as they are evaluated in the same order. Therefore, we can make this change and still get the same results by evaluating the measurements in the same order in which they were recorded. Meaning, we first supply all measurements taken in time step one by both robots to the particle filter algorithm before moving on to the measurements taken at time step two and so on.

The last modification we make to the system itself is the introduction of a new module. Its purpose is to offer an interface which enables the user to trigger a series of experiments using either the dec-POMDP planning approach or the random movement approach. When called it triggers an experiment, waits for it to finish and triggers the next one until the specified number of experiments is reached.

By applying these new modifications to our system, we can record for example 100 experiments in a row. However, as shown in the previous section, it is rather impractical to record enough experiments in the real world to give an accurate assessment of the performance both approaches yield. Therefore, we use simulation to record more experiment data.
As it stands, the system already offers the ability to run multiple client nodes on a single host system as long as a specific name is defined for each client node. The measurement taking process can also be simulated by sampling directly from the received signal strength model we introduced in section 2.4. Since this RSS model was fitted to a data set of example measurements recorded explicitly with equipment used throughout this thesis, the simulated measurements are very similar to the measurements a robot would perceive in a physical experiment.

Figure 5.9 The movement graph used for all experiments recorded to compare the dec-POMDP to the random movement approach



To show the efficiency of using a dec-POMDP planning algorithm on our signal source localization we record 200 experiments using the dec-POMDP approach and the same amount using the random movement approach. All of these experiments are conducted using the same movement graph which is displayed in Figure 5.9. The blue dots represent the locations a robot could move to while the blue lines indicate where a robot is allowed to move to next.

The black arrow marks the location and orientation of both robots before the first experiment. The Wi-Fi router location is marked by a purple square on the right hand side.

For the planning algorithm and the experiments themselves we use the parameters shown in Table 5.3. Most of the parameters stay the same for all experiments conducted in this section, but only half of the experiments are conducted with an event horizon of four and five improvement steps. The other half is conducted with an event horizon of three and four improvement steps. This allows us to investigate the impact of the event horizon and number of improvement steps on the overall result of an experiment.

Table 5.3 Experiment parameters for the experiments described in Section 5.3

Planning algorithm parameters			
Event Horizon	4 / 3	Policy Width	3
Improvement Steps	5 / 4	Rollouts	100
Particles for Planning	1000	Particles for each Rollout	100
Client and Evaluation parameters			
Measurements per action	15	Particles for Evaluation	1500

To compare multiple particle filter results to one another, we use the weighted root mean squared error (RMSE) function introduced in section 5.1. By applying the weighted RMSE function, we can determine a quantitative value for the accuracy of a particle filter belief. Using this value we can compare the results of multiple experiments to determine what approach yields the best results.

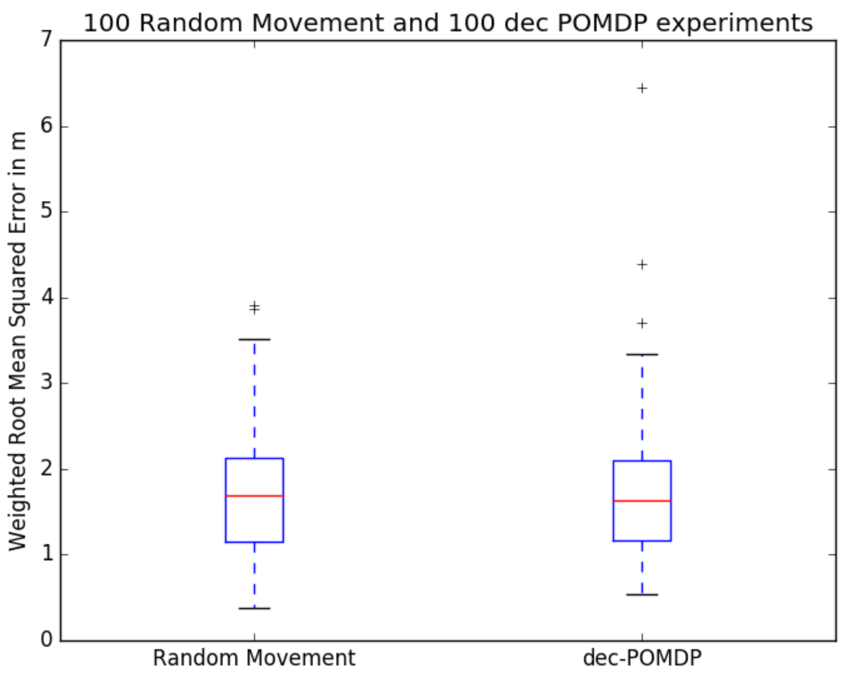
The first set of experiments we consider is conducted with an event horizon of four and five steps during the planning process. We look at 100 experiments using the dec-POMDP approach and compare them to 100 experiments using the random movement strategy. For each of these experiments we calculate the weighted RMSE for the resulting particle filter and display it in Figure 5.10. The displayed box plot indicates that both approaches yield very similar results. Even when we compare the mean and median weighted RMSE of this data set directly as shown in Table 5.4, the difference between the two approaches is insignificantly small.

Table 5.4 Results for 200 experiments in simulation with Event Horizon: 4 and Improvement Steps: 5

	100 experiments using the dec-POMDP approach	100 experiments using the random movement approach
Average weighted RMSE	1.77 m	1.69 m
Median weighted RMSE	1.63 m	1.70 m

Because the difference between the two approaches is negligible, this experimental setup

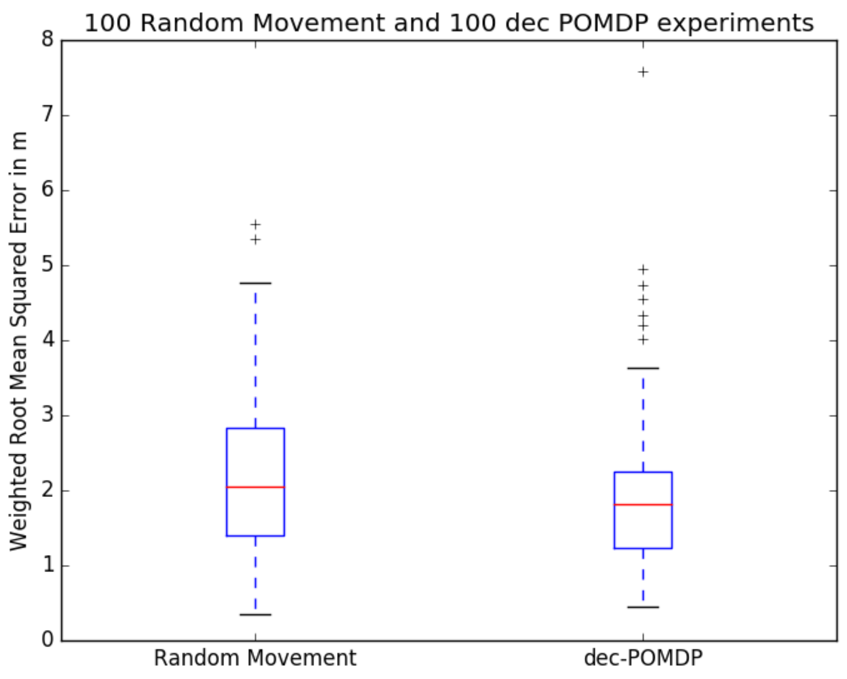
Figure 5.10 Comparison between 100 experiments using the random movement strategy and 100 experiments using dec-POMDP planning with an event horizon of 4



cannot be used to demonstrate that dec-POMDP planning has an advantage over a random movement strategy. However, this finding still is important to this thesis. It shows that for the proposed source localization problem, both strategies can yield equally good results for a specific parameterization. The main reason behind the similarity is the event horizon in comparison to the number of locations. Since each of the robots can take four actions, and there are only nine locations both robots could visit, it is not unlikely that a random movement approach could collect measurements for a significant portion of the overall available locations. While the planning algorithm tries to use the four actions per robot it has as efficiently as possible, the random movement approach reaches similarly good results since, with four actions per robot, it is likely to visit enough locations for a reasonable estimation of the signal source even without proper planning. To underline this point, we demonstrate the impact of lowering the event horizon with more simulated experiments.

The Figure 5.11 shows the results of 100 experiments for each approach with an event horizon of only 3. Also the dec-POMDP planning algorithm was applied to this problem with only 4 improvement steps instead of the 5 that were used in the previous data set. This change was made to reduce the planning time.
As one can see from the results presented in Figure 5.11 and the exact mean and median of the weighted RMSE for both approaches displayed in Table 5.5, the dec-POMDP approach

Figure 5.11 Comparison between 100 experiments using the random movement strategy and 100 experiments using dec-POMDP planning with an event horizon of 3



does have a small advantage over the random movement approach. But this advantage of only $21cm$ is still an insignificant difference between the two approaches.

Table 5.5 Results for 200 experiments in simulation with Event Horizon: 3 and Improvement Steps: 4

	100 experiments using the dec-POMDP approach	100 experiments using the random movement approach
Average weighted RMSE	1.99 m	2.20 m
Median weighted RMSE	1.82 m	2.04 m

What we can conclude from the presented results however, is that the dec-POMDP approach does show similarly good results as the random movement strategy does. Also, we have demonstrated that our system is capable of simulating experiments to gather larger sets of data in simulation.

6 Conclusion

To conclude this Bachelor thesis on the topic 'Locating the source of a WLAN signal using multiple robots and Dec-POMDP planning', this section sums up the findings and achievements of this thesis. It looks back on the original lead question, whether it is possible to apply dec-POMDP to a signal source localization problem with a continuous state space, and conduct physical experiments successfully, by employing a policy graph improvement algorithm.

The first and most important achievement of this thesis, is the implementation of a system that can successfully call a policy graph improvement algorithm and apply the generated policies to solve a signal source localization task. During the course of this thesis, we included the non-linear policy graph improvement algorithm for dec-POMDPs with a continuous state space, introduced by M. Lauri et al. [Lauri, Pajarinen, and Peters 2020], into a larger system. We have developed a client-side package that can be deployed on a TurtleBot2, which allows the robot to execute a given policy autonomously. Furthermore, we introduced a central evaluation node that can successfully approximate a Wi-Fi router's location by applying a particle filter to the signal strength measurements collected during policy execution. Finally, we also developed a central communication node that handles all communication between the used robots and server-side nodes for planning and evaluation.

While the original paper, which introduced the planning algorithm, only tested the algorithm on a simulated signal source localization task, the system introduced in this thesis can be applied to a real world scenario. As we have shown in section 5.1 the system is able to plan two policies, distribute them to two robots and evaluate the measurements both robots gathered during policy execution in a physical experiment.

To enable the measurement evaluation, we use a Wi-Fi router as our WLAN signal source and fit the received signal strength model originally introduced in [Atanasov, Le Ny, and Pappas 2014] to our experimental setup in section 3.1. To achieve this, we recorded a set of example measurements using the TurtleBot2 robot and a Huawei P20 Lite as the Wi-Fi router and fitted the RSS model to it. This model allows us to approximate the distance between a location and the signal source from a measurement taken in that location.

Furthermore, we introduced a random movement strategy to solve the signal source localization task. This alternative approach to solve the proposed problem can be used as a baseline. In section 5.3 we compared this baseline to the results of the dec-POMDP approach. We conducted 400 experiments and deduced from their results that for certain experimental setups the dec-POMDP approach shows similar results to the random movement baseline.
All of these experiments had to be conducted in simulation to not require constant supervision

and an unreasonable amount of time, but since we only made a few minor changes, we can expect experiments in the real world to show similar results.
From the evaluated data sets, we concluded that the dec-POMDP and random movement approach have a similar accuracy for experimental setups where the combined event horizon of both robots is near the total number of locations. We have also shown that experiments with a lower event horizon on the same movement graph result in a small but still insignificant advantage in accuracy for the dec-POMDP approach. Based on this information, we presume that the similarity in accuracy for the two approaches depends on the defined experimental setup. The difference in accuracy is likely to increase if the gap between locations and event horizon widens. As we already discussed in section 5.3, the random movement approach's accuracy depends on the chance to visit as many different locations as possible. If both approaches can take measurements in every possible location, because the combined number of actions the agents are allowed to take is close to the number of locations, there is a high chance that both approaches generate equally accurate results. However, if there is a significant difference between the combined number of actions and the number of locations, the impact of which locations are chosen to take measurements increases. Therefore, the chance that the dec-POMDP approach outperforms the random movement approach could potentially increase for more complex movement graphs.

More importantly, for this thesis, we can conclude from these experiments that it is indeed possible to successfully apply the NPGI algorithm for dec-POMDPs to a signal source localization problem. We have not only shown that the planned policies can successfully be applied to a real-world experiment to get an approximation of a WLAN signal source, but also demonstrated that our approach shows similarly good results as a random movement strategy.
Based on this knowledge, we can also conclude that dec-POMDP as a framework can successfully be applied to information gathering tasks with a continuous state space.

Bibliography

Andersen, M.R. et al. (Sept. 2012). "Kinect Depth Sensor Evaluation for Computer Vision Applications". In: *Technical Report Electronics and Computer Engineering* 1.6. Available at: https://tidsskrift.dk/ece/article/view/21221 Accessed: 06.08.2020.

Aşık, Okan and H. Levent Akın (2013). "Solving Multi-agent Decision Problems Modeled as Dec-POMDP: A Robot Soccer Case Study". In: *RoboCup 2012: Robot Soccer World Cup XVI. Lecture Notes in Computer Science, vol 7500.* Ed. by Xiaoping Chen et al. Springer, pp. 130–140. DOI: 10.1007/978-3-642-39250-4_13.

Atanasov, Nikolay A., Jerome Le Ny, and George J. Pappas (Oct. 2014). "Distributed Algorithms for Stochastic Source Seeking With Mobile Robot Networks". In: *Journal of Dynamic Systems, Measurement, and Control* 137.3. 031004. Available at https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/137/3/031004/6118584/ds_137_03_031004.pdf Accessed: 14.08.2020. ISSN: 0022-0434. DOI: 10.1115/1.4027892.

Bernstein, Daniel et al. (Dec. 2002). "The Complexity of Decentralized Control of Markov Decision Processes". In: *Mathematics of Operations Research* 27. DOI: 10.1287/moor.27.4.819.297.

Beynier, Aurélie and Abdel-Illah Mouaddib (Oct. 2011). "Applications of DEC-MDPs in Multi-Robot Systems". In: *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*. Ed. by Enrique Sucar, Eduardo Morales, and Jesse Hoey. IGI Global. Chap. 16, pp. 361–384. DOI: 10.4018/978-1-60960-165-2.ch016.

Eker, Baris et al. (2011). "A finite horizon DEC-POMDP approach to multi-robot task learning". In: *2011 5th International Conference on Application of Information and Communication Technologies (AICT)*. Available at: https://ieeexplore.ieee.org/abstract/document/6111001, Accessed: 16.11.2019. DOI: 10.1109/icaict.2011.6111001. URL: https://ieeexplore.ieee.org/abstract/document/6111001.

Fox, Dieter (2001). "KLD-Sampling: Adaptive Particle Filters". In: *Advances in Neural Information Processing Systems 14*, pp. 713–720.

HOKUYO AUTOMATIC CO., LTD (2009). *Scanning Laser Range Finder URG-04LX-UG01 Specification*. Avalable at: https://www.hokuyo-aut.jp/ Accessed: 14.08.2020.

— (2012). *Scanning Laser Range Finder UTM-30LX/LN Specification*. Avalable at: https://www.hokuyo-aut.jp/ Accessed: 06.08.2020.

Lauri, Mikko, Joni Pajarinen, and Jan Peters (2020). "Multi-agent active information gathering in discrete and continuous-state decentralized POMDPs by policy graph improvement". In: *Autonomous Agents and Multi-Agent Systems* 34.42. DOI: 10.1007/S10458-020-09467-6.

National Oceanic and Atmospheric Administration (2018). *How much of the ocean have we explored?* NOAA: National Oceanic and Atmospheric Administration. Available at: `https://oceanservice.noaa.gov/facts/exploration.html`, Accessed: 27.10.2019.

OECD: Organisation for Economic Co-operation and Development (2014). *The Space Economy at a Glance 2014*. Available at: `https://www.oecd-ilibrary.org/economics/the-space-economy-at-a-glance-2014/space-budgets-of-selected-oecd-and-non-oecd-countries-in-current-usd-2013_9789264217294-graph6-en`, Accessed: 27.10.2019. OECD Publishing.

Oliehoek, Frans A. and Christopher Amato (2016). *A concise introduction to decentralized POMDPs*. Springer, Cham. DOI: `10.1007/978-3-319-28929-8`.

Quigley, Morgan et al. (2009). "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan, p. 5.

Särkkä, Simo (2013). *Bayesian filtering and smoothing*. Cambridge University Press.

Thrun, Sebastian, Wolfram Burgard, and Dieter Fox (2005). *Probabilistic Robotics*. The MIT Press.

TurtleBot2 (n.d.). Open Source Robotics Foundation. Available at: `https://www.turtlebot.com/turtlebot2/`, Accessed: 30.10.2019.

Twigg, J. N. et al. (May 2012). "RSS gradient-assisted frontier exploration and radio source localization". In: *2012 IEEE International Conference on Robotics and Automation*, pp. 889–895.

van den Berg, J., Ming Lin, and D. Manocha (2008). "Reciprocal Velocity Obstacles for real-time multi-agent navigation". In: *2008 IEEE International Conference on Robotics and Automation*, pp. 1928–1935.

Virtanen, Pauli et al. (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17, pp. 261–272. DOI: `https://doi.org/10.1038/s41592-019-0686-2`.

Wang, Minlue, Richard Dearden, and Nick Hawes (Sept. 2015). "Robot plans execution for information gathering tasks with resources constraints". In: *2015 European Conference on Mobile Robots (ECMR)*. Available at: `https://www.researchgate.net/publication/308732137_Robot_plans_execution_for_information_gathering_tasks_with_resources_constraints`, Accessed: 16.11.2019, pp. 1–6. DOI: `10.1109/ECMR.2015.7324216`.

Wu, Fang-Jing and Tie Luo (2014). "WiFiScout: A Crowdsensing WiFi Advisory System with Gamification-Based Incentive". In: *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*, pp. 533–534.

— (2015). "Infrastructureless signal source localization using crowdsourced data for smart-city applications". In: *2015 IEEE International Conference on Communications (ICC)*, pp. 586–591.

Zafari, Faheem, Athanasios Gkelias, and Kin Leung (Sept. 2017). "A Survey of Indoor Localization Systems and Technologies". In: *IEEE Communications Surveys and Tutorials* PP. DOI: `10.1109/COMST.2019.2911558`.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Software System Entwicklung selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den 17.08.2020

Tobias Krüger

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 17.08.2020

Tobias Krüger