UNIVERSITY OF TURKU

Reinforcement Learning based positive phototaxis of mobile robot with minimal sensing

Faculty of Technology

Master's thesis

Onni Wuoti

26.5.2026

Turku

Master's thesis

Subject: Mechanical Engineering

Author(s): Onni Wuoti

Title: Reinforcement Learning based positive phototaxis of mobile robot with minimal sensing

Supervisor(s): Wallace Moreira Bessa, Gabriel Da Silva Lima

Number of pages: 77 pages

Date: 26.5.2026

In this thesis a Q-learning based reinforcement learning algorithm was created to perform gradient following behavior in the form of phototaxis with a mobile robot. The framework was made to work under minimal sensing which used a single sensor for reward value tracking. To learn preferences over active and passive phases of movement, a four-state space was used by the algorithm. The proposed methodology contrasts previous approaches by using a single sensor instead of multiple, and by relying on learned control instead of direct motor control.

The algorithm was tested in both simulation and an experimental setup to determine robustness and to test the generalization of the algorithm. The results were evaluated based on the robot's movement towards the highest reward point and a one meter goal area around the peak. After arrival the robot spent majority of its time near the peak, and it learned to move towards its current heading when oriented towards the highest reward area. The results demonstrated consistent gradient following behavior for phototaxis with only a single sensor in both simulated and experimental environments.

In the future this algorithm should be researched further by filtering sensor readings, attempting different parameter combinations, and experimenting with the usage of multiple agents or reward sources.

Keywords: phototaxis, reinforcement learning, mobile robot, Q-learning, Markov switching, gradient following

Generative Artificial Intelligence was used for grammar tweaking and for structural clarity

Diplomityö

Oppiaine: Konetekniikka

Tekijä(t): Onni Wuoti

Otsikko: Vahvistusoppimiseen pohjautuva positiivinen fototaksia mobiilirobotille minimaalisella havainnoinnilla

Ohjaaja(t): Wallace Moreira Bessa, Gabriel Da Silva Lima

Sivumäärä: 77 sivua

Päiväys: 26.5.2026

Tässä diplomityössä testattiin Q-oppimiseen perustuvaa vahvistusoppimisalgoritmia gradientin seuraamiseen fototaksian muodossa mobiilirobotilla. Runko toimii minimaalisen sensoritekniikan avulla, joka käytti yhtä sensoria palkinnon arvon seuraamiseen. Algoritmi käytti neljän tilan tila-avaruutta oppiakseen mieltymyksiä, milloin valita aktiivinen tai passiivinen vaihe liikkeelle. Ehdotettu menetelmä eroaa edellisistä lähestymistavoista käyttäen yhtä sensoria usean sijaan ja käyttämällä opittua ohjausta suoran moottoriohjauksen sijaan.

Algoritmin häiriönsietokyvyn määrittämiseksi sekä sen yleistettävyyden testaamiseksi muissa ympäristöissä, se testattiin sekä simulaatiossa että kokeellisessa ympäristössä. Saadut tulokset arvioitiin perustuen robotin liikkeeseen kohti korkeinta palkintopistettä, ja yhden metrin kokoista maalialuetta huipun ympärillä. Robotti vietti suurimman osan ajastaan huipun läheisyydessä saapumisen jälkeen ja oppi liikkumaan kohti nykyistä kulkusuuntaansa, kun se oli kohti korkeinta palkintoaluetta. Tulokset osoittivat fototaksisen gradientin seurannan olevan mahdollista yksittäisellä sensorilla sekä simuloiduissa että kokeellisissa ympäristöissä.

Tulevaisuudessa algoritmia tulisi tutkia suodattamalla sensorin lukemia, testaamalla erilaisia parametrien yhdistelmiä, ja kokeilemalla useamman agentin tai palkintolähteen käyttämistä.

Avainsanat: fototaksia, vahvistusoppiminen, mobiilirobotti, Q-oppiminen, Markov vaihtelu, gradientin seuranta

Generatiivista tekoälyä käytettiin kieliopilliseen hienosäätöön ja rakenteelliseen selkeyteen

Table of contents

List of abbreviations and symbols

RL	Reinforcement Learning
DPG	Deterministic Policy Gradient
DRL	Deep Reinforcement Learning
MDP	Markov Decision Process
DR	Domain Randomization
ARL	Adversarial Reinforcement Learning
TL	Transfer Learning
MoCap	Motion Capture
DTMDP	Discrete Time Markov Decision Process

v	linear velocity
ω	angular velocity
ω_R	right wheel angular velocity
ω_L	left wheel angular velocity
r	robot wheel radius
L	distance between wheel centres
ϕ	robot heading angle
$\theta(x, y)$	bearing from heading to target center
x_c	x-coordinate of goal
y_c	y-coordinate of goal
σ	standard deviation of noise
ξ	standard normal random variable
λ	spatial decay coefficient
L_0	maximum available reward
f_t	phase indicator
$r_t^{\uparrow}$	binary reward improvement indicator
s_t	current state
a_t	action
α	Markov switching probability

β	Markov switching probability
R_t	scalar reward
R_{old}	reward memory
$Q(s,a)$	action-value function
α_{qt}	learning rate
γ	discount factor

1 Introduction

In the modern world, robots are becoming an increasingly visible part of everyday life. Whether it is food-delivery bots going door-to-door, or self-driving cars taking over the streets, the change in the direction of robotics is clearly underway. The usage of robots that are capable of automatic operation, known as mobile robots [1], is a constantly increasing area of interest for many sectors in the field of technology. The ways in which autonomy is achieved, and the methods for robot control include many different possibilities and has a constant need for innovation to create better and more advanced systems. Autonomous robots have numerous applications including, but not limited to, assistive robots in environments with human interaction [2], construction [3], and exploration tasks such as rescuing operations or patrolling [4]. What most of the tasks have in common is the need for robots to have knowledge of the environment and the ability to independently move and navigate within it while maintaining safety and fulfilling its objectives [4].

Some common methods to achieve autonomous robot control include Machine Learning methods like Supervised Learning, where training data is used to teach correct labels and classes, Reinforcement Learning (RL), where reward systems are used for learning, and Unsupervised Learning, where training data is used but it is unlabeled and finding the correlation is the algorithms job [5]. Other methods include more classical control strategies like Model Predictive Control (MPC), where a mathematical model is used to predict future behaviour, while working under physical and safety constraints incorporated into the optimization [6], and reactive control methods, where actions are based on current sensor data and the robot's reflexes to progress step by step towards its target [7]. Challenges in autonomous control include dynamic environments, where the robot has to work with unpredictable circumstances, limitations in its sensing abilities, and constraints in computing and processing. Even though classical control methods provide uncertainty handling via tools such as probabilistic filters, learning-based methods can offer advantages in real-time handling of dynamic environments [8], which motivated the usage of them in this thesis in the form of reinforcement learning.

Gradient following behaviour to seek a source has many practical applications where the goal is to find a zone of interest automatically. These include inspection tasks like gas leakage [9] or identifying odor sources indoors [10], search and rescue operations [11] where unsafe environments require robots for seeking, and environmental monitoring [12] where sources represent anomalies or pollutants. This thesis addresses the gradient following behaviour by contributing a Q-learning framework working under minimal sensing constraints, which demonstrates the usage of a single sensor in a source seeking application.

To work with Reinforcement Learning, the goal of this thesis is to create a framework where a reinforcement learning algorithm is used to achieve gradient-following behaviour. A gradient in the context of this thesis is the spread of the reward that the RL algorithm is receiving. The gradient is created by a source, which in the case of phototaxis will be a light source. An efficient algorithm will learn to move towards the center of the reward source as that has the highest concentration of reward. The framework uses only the minimal information it is given and does not have any prior knowledge about the reward source position, or the robot's own position within the world. After reaching the goal, the robot's ability to demonstrate retention of its position is also considered, as it provides further information regarding the framework's robustness. Different variations are compared as the effect that slight changes in reward shaping makes should be determined in order to create the most robust algorithm. By developing this framework, the thesis aims to demonstrate a simpler solution to achieving phototaxis behaviour, relying only on minimal information and hardware. To test the algorithm in this thesis, experiments were conducted in both simulated and experimental environments to examine the sim-to-real problems which may occur, including the physical constraints when compared to a perfect simulated environment. This was done to determine usability in more unpredictable and noisy environments.

The usage of a learning-based framework opens opportunities for further development in uncertain and changing environments, as the system is not tied to explicit model usage. By lowering the computational complexity and the need for sensors, using RL becomes more accessible to a wider range of robots. This also creates openings for

more diverse applications within the real-world, as simpler setups can be considered as use cases while still maintaining robustness of the framework.

1.1 Research objectives and contribution

The objective of this thesis is to explore the area of autonomous exploration by using learning-based methods, more specifically to answer the following research questions:

- RQ1: Can a mobile robot achieve gradient following with minimal information?
- RQ2: How do reward design choices affect the robustness and navigation performance of the algorithm?

By answering these research questions, the thesis contributes to RL discussion by demonstrating that navigation is possible while only using a singular sensor and a four-state space. This contrasts with RL approaches which often require multiple sensors with more advanced information like LiDARs and have the need for sensor fusion which makes the algorithms computationally expensive. The robot's movement will be only directed via switching between two phases, either being passive where the robot focuses on finding new headings, and active where the robot moves towards its current heading. By implementing the algorithm in an experimental environment, the algorithm is validated further and tested against uncertainty. The system is represented in four states which come from the active/passive phases of the robot and the increase/decrease of the reward. This is sufficient for the task as knowing the change in reward is all that is needed to determine whether the movement is towards the reward source, and the phases provide the necessary actions that are available. Although this simplification does create some information loss when considering aspects such as the distance to the source or coordinates, the simplicity of the Q-table results in a lower computational load which is relevant when the aim is to work with minimal sensing and it creates good opportunities for usage on embedded hardware.

1.2 Structure

The structure of this thesis is organized in the following way. The background and theory sections are covered in Chapter 2, which gives a general overview of the state of the art

in the areas of taxis movement, minimal sensing, learning-based control of robots, and reinforcement learning in robotics. The chapter provides the necessary theoretical information to understand the working principles involved, and the areas taken into consideration when developing the approach of this thesis. After providing sufficient background information, the physical system used in this thesis is described in Chapter 3. The chapter provides the required information for recreating the used setup, and the necessities involved within in it.

Chapter 4 goes through the actual framework used as methodology in this thesis. The chapter shows the structure of the algorithm and explains the components it is based on. After this the results of the simulated and experimental versions are shown and analysed in Chapter 5, which shows comparison and deployment of the algorithm moving from simulation to reality. Conclusions of the thesis and potential future directions are covered in Chapter 6, which summarizes the contributions of this thesis.

2 Background and Literature Review

Taxis is a well-known phenomenon especially in biology. It has also been translated into robotics in some cases. In this chapter, some sources of inspiration are delved into and closely related areas in robotics are also covered, including learning-based systems, minimal sensing and reinforcement learning. By doing this literature review on the topic, a better understanding of the state of art is achieved which helps in deepening the understanding of this thesis.

2.1 Taxis in biology and robotics

The term taxis means an organism's movement directed by the effects of an external stimulus. In nature this can be seen as an animal's movement towards zones of interest for example warmth or food sources. Examples of different variations of taxis include chemotaxis the movement towards a chemical source, phonotaxis the movement towards sound signals, and phototaxis the movement towards light sources. More specifically, positive taxis refers to movement towards a source, and negative taxis away from it. Taxis behaviour is useful as it can be used by animals to help in actions such as foraging and navigation. However, the behaviour is not fixed and is often transformed by the experiences animals have from performing different tasks. For example, in a study about honeybees their role in a colony had an impact on how the bees reacted to taxis. Nurse bees that spent a large amount of time inside were found to be less reactive to light stimuli when compared to foraging bees working outside. [13], [14]

Chemotaxis is often researched as a part of cells and their working principles. One particular study focused on the determining factors of cell behaviour during chemotaxis, and whether spatial or temporal sensing would be used. Temporal sensing refers to using information from two different time points during a cell's movement, whereas spatial sensing refers to the cell using different parts of its surface simultaneously to determine the change in its gradient. The study made important findings in identifying key factors of the cell behaviour. One of these was the ratio between the cell's speed and its diameter, for which a higher ratio seemed to make

temporal sensing more common. Some features were also listed as temporal sensing being more robust when affected by noise. This was explained by the average taken during it when compared to the spatial sensing approach which uses the difference. [15] The examples of these sensing types is relevant to the study, as they are important elements of gradient sensing which is very much at the core of every type of taxis application.

During the chemotaxis movement, the cells often use a strategy known as run-and-tumble. Cells that have been noted to use this strategy include bacteria, primordial germ cells, and immune cells. It is a known phenomenon which resembles a biased random walk at its core. When using run and tumble, the cells also use temporal sensing to determine whether they are moving in the right direction. The working principle of run and tumble is simple and can be explained as the two phases it has. When in run mode, the concentration of the attractant is measured and if rising the direction is kept the same. Cells also have the ability to adapt to the higher concentrations it encounters, by doing which it recalibrates its sensors avoiding over saturating inputs. The tumbling phase is activated in cases where the concentration starts to decrease. When tumbling, the cell starts to move in reverse which causes a small tumbling effect, after which the cell starts moving in a new random direction. [16] In this thesis the idea of run and tumble is used in the context of using different phases for movement where there are practically both run and tumble states. This is discussed in detail later in the thesis.

Applying the working principles of taxis to robotics is by no means a new concept. A study done by Rañó [13] developed a version of this where taxis was used for the creation of a control scheme of a non-holonomic dual-drive robot which follows standard differential-drive system equations. In their study the taxis was modelled after a scalar function where the maximum spot or the source, was placed in the origin of the workspace. For the setup the dual-drive robot was equipped with two sensors, where each sensor was connected to one motor respectively. This was done to emulate bilateral symmetry and to have sensors directed towards the direction of the robot. Having one sensor per motor creates a simple working principle where the connections were decreasing, so that a higher sensor reading corresponded to a lower turning rate

for the wheel connected to that sensor. This allows easy directional manipulation directing the robot towards the higher reward. On top of this as the reward got larger, the turning rates of both wheels were lowered so that the robot would slow down the closer to the origin it got. The resulting model worked with any starting pose and was able to always drive the robot to the maximum of the stimulus. The tests were only performed via simulation, and no physical implementation of the controller was developed. The provided study is an example of the most straightforward version of a taxis robot, known as the Braitenberg vehicle.

The Braitenberg vehicles have one or more sensors with wheels independent from one another. The sensors are directly connected to the motors individually, manipulating their actions and the robot's movement. The sensors can either be connected ipsilaterally or contralaterally, and the actuation either slows them down or speeds them up. This creates a very simple intelligence version for the robot, which is capable of tracking sources and moving towards them. [17] Although the Braitenberg vehicles are easy to use and quick to set up, they are not quite as advanced as other methods, as they are only following the circuit they have been provided. When applying to real world robotics, it is important to consider the uncertainties that come with real hardware and environments. When compared to a learning-based algorithm, the Braitenberg vehicle falls short in areas where for instance a weaker motor or changing environment is involved, as the hardwired connections in a Braitenberg vehicle do not allow any feedback system to compensate for changes in the system. In a learning-based system the robot is taught to understand the rewards and the goals, which gives it more adaptability when in comparison.

Most importantly in the context of this thesis, the concept of phototaxis has its own implementations in both nature and robotics. In nature, phototaxis is an important aspect of many microorganisms that they use for autonomous navigation tasks. Different organisms use different strategies for this behaviour, including intensity-based tumbling or more complex internal reorientation mechanisms using internal feedback loops to direct their movement. The working principles creating the behaviour need an element motivating the orientation of the organisms. Only altering the velocity depending on brightness will not direct the organism towards the gradient source, but it

needs something more to give it determination. [18] Organisms can orientate themselves towards a light source by using aligning torques [18], and in robotics the behaviour can be achieved by using learning-based approaches, which is the focus of this thesis. Applying phototaxis to robotics has been investigated both experimentally with a singular mobile robot to harness natural light indoors for recharging [19], and numerically with a virtual robot swarm to perform negative phototaxis and move the whole swarm away from a light source [20]. Out of these two case studies, the first one is in closer relation to what this thesis is attempting to achieve, and inspecting its capabilities and working principles serves as a good comparison and helps understand areas of innovation.

The phototaxis robot created by Vaussard et al. [19] used four photodiodes to perform spatial sensing and guided the robot by using a simple P controller to continuously reorientate the robot towards the strongest signal measured by the diodes. The robot was equipped with a solar panel on top of it, and the purpose of the study was to allow the robot to perform self-charging indoors by navigating to spots of sunlight. After arrival the robot would rotate the solar panel precisely for optimal charging. The study achieved high efficiency raising the battery percentage, and that 16 hours of recharging allowed one hour of movement. In regard to phototaxis, two parts of the setup are the most relevant to this thesis. The method for determining the gradient, and the reorientation of the robot towards areas which increase the received values. These can be generalized to spatial sensing and manual reorientation based on the data from multiple sensors.

2.2 Minimal sensing in robotics

In robotics, a lot of the core mechanism in autonomy comes from two parts of the robots, sensors and actuators. The selection of these can be detrimental to the capabilities of the robot and change its properties noticeably. In minimal sensing the information availability is limited purposefully to the minimal amount required by an application. In the most extreme scenario, this means using a singular sensor. Minimal sensing has both pros and cons, which are important to acknowledge and understand before considering its implementation.

Some of the pros attached to minimal sensing have been noted to be the following:

- Simpler, more straightforward setup, which is a good quality when considering using multiple robots at once as the setup becomes easier to implement [21].
- Reduced costs as less material is needed, although having minimal sensing can also mean using a more expensive and advanced singular sensor instead of multiple simpler ones [22], [23].
- A simpler sensing architecture has been noted to be less susceptible to failure, and it has good grounds for scaling up to a collaborative robot team [22].
- They are also robust against sensing uncertainty as no communication and data comparison between different sensors is needed [22].

On the other hand, minimal sensing inherently brings some downsides as well which include some of the following:

- Having less sensors inherently lowers the capabilities of the robot and the information it gets. This can be seen as fundamental limitations in performance when compared to using more sensors [24]. An example of this that can be pointed to is the lack of knowledge in global turning direction [22].
- Using minimal sensing can lead to greater efforts in the designing process, so that the capability of sensing is enough [25].
- Also the dexterity of a robot can take a hit, as having minimal sensing often involves focusing on a single optimized feature [25].

Minimal sensing leads to the incomplete knowledge of the surrounding environment. This partial observability leads to the difference between the observed environment and what really exists. In minimal sensing the partial observability is created on purpose with the intention of reducing the need for multiple sensors. In partially observed environments a robot's movement and decision making has to rely on some history of its previous actions, so a memory needs to be maintained. This can complicate planning and learning, as exploration, learning, and memory reasoning has to all be

done simultaneously by the robot. Also, it may lead to extensive sampling to gather enough information for the right actions. This makes controlling more difficult as the latent state is not visible, and the robot does not know its exact conditions. The observed information can also be mapped as one-to-many, where two conditions of the robot might actually be the same in reality. [26], [27] Control in partially observed environments needs to account for the limited information that is available. Using methods such as reinforcement learning has good potential in these environments as they are made to work with noisier and incomplete information [28].

Minimal sensing has been used in robotics for a variety of different applications. A great example is the creation of a twining robot arm investigated by Naselli et al. [25]. The purpose of the study was to mimic the functionality of a climbing plant species. The design of the robot arm involved taking advantage of embodied intelligence by using changes within the arm's structure which reduced the need for sensors. This was inspired by biology where often structures have varying flexibility in their body, being stiffer at the bottom gives them the support and flexibility at the tip allows them to still control objects. The arm used a single air-pressure sensor and based on the changes in thickness along the arm it was able to twine around supports. This reduced the need for multiple sensors, demonstrating the capabilities of using embodied intelligence as a design strategy in minimal sensing. Minimal sensing has also been applied in other instances to robotic applications. Examples include gas leak searching swarm of drones equipped with four single-beam range sensors [29], mapping and localization of vacuum cleaners equipped with only infrared and ultrasonic sensors [30], and cluster formation of drone swarm with the smallest possible sensing range of 2 units and inability to communicate with each other [31]. The examples show good design strategies for implementing minimal sensing into applications.

Design strategy is an important aspect of any creative task and to understand them in more detail, the reasoning for the chosen designs should be discussed. In all of these studies the limitations came mostly from hardware restrictions or informational constraints. The nano-drones used for gas leak search all had very strict payload restrictions, weighing only around 50 grams by themselves. Thus, a heavy camera setup was not an option, and lighter noisier sensors had to be used. The study demonstrates

exploiting swarm redundancy as a design strategy which increases tolerance to sensor uncertainties. [29] The localization and mapping of vacuum robots used a fundamental assumption that the environments were rectilinear. It was also restricted by the expenses as the robot vacuums were intended for commercial purposes, so only low-cost sensors were used. By using environmental assumptions as design strategy, the inputs could be simplified and localization was achieved. [30] The self-organizing swarm of drones was a simulation-based experiment where 2 sensing units had no physical sensor connection. Still the acknowledgment of this study is important as it highlights how the usage of rules for local interaction as a strategy can overcome the minimal sensing capabilities to achieve uniform global actions. [31] The examples presented here are all examples of how minimal sensing can be implemented by designing systems and strategies that consider the lack of knowledge received via sensory inputs.

Although increasing sensory input offers a variety of different applications that take advantage of the information, using minimal sensing is still an important section of robotics which should be considered. Using minimal sensing has its limitations including incomplete information, dexterity of a robot, and greater design efforts, which highlights the motivation to create learning-based controllers for robots under these constraints.

2.3 Learning-based control of robots

Controlling robots in uncertain environments requires capabilities of adaptability and adjustability. Information received from traditional sensors such as ultrasonic or laser sensors can often be insufficient, which creates limitations where the working environment has to remain unchanged [32]. Traditional methods often rely on defined models which creates limitations in their capabilities, including difficulties in adaptability to the environment. The limitations for classical control methods of robots include the lack of robustness caused by uncertainties, tuning complexity, and the need for accurate system modelling. In recent years the transition to learning-based control frameworks has been notable and has given a new direction for robot control. Conventional linear methods work well in stable working conditions, which is often not

the case in robotics. Although adaptive control methods designed for uncertain environments exist, the usage of learning-based approaches is a suitable solution for them offering more flexibility. Newer robots are often more complex requiring connected algorithms and introduce nonlinear systems with uncertain environments, which creates the motivation to use learning-based controllers. [33]

To accomplish learning outcomes, two known methods are introduced. Both model-based and model-free learning are methods suitable for predicting a robot's actions and adjusting based on relevant information. In the model-based method, a representation of the working environment is created, which can then be used for calculations to make a prediction of future values. In model-free learning the working principle is quite the opposite where cached information of the environment is collected and valued. Model-based algorithms generally speaking direct actions towards a certain goal, having their environmental representations work as an internal map. In contrast to this, model-free algorithms create rules on what actions to take based on the collected information, and work while being detached from possible outcomes. Computationally speaking model-based strategies are statistically efficient when doing prediction, although they can be somewhat computationally demanding. Large calculations are often needed when evaluating all possible future choices since it includes a far-reaching tree of options. On the other hand, the predictions are often accurate. Model-free strategies have to make the predictions based on previously encountered values and are not easily adaptable to internal state and environmental changes as they have to experience the change for a long enough time for it to have an effect. Learning is not as statistically efficient as calculations are not as complex either, and the accuracy depends highly on the estimations made by the system as it needs time to learn. [34], [35]

When considering learning-based control, it is important to understand and acknowledge some of the core categories involved in it to help distinguish them and to evaluate which one to use. One of these categories is supervised learning, which is based on using learning sets to map inputs to outputs [36]. Another category is adaptive control where the parameters of the controller are updated continuously to optimize the controller in a changing environment. It does not need prior information and can perform parameter estimation in real-time for unknown parameters. [37] In addition

there is also imitation learning, where mapping of observations and actions is learned from existing demonstrations. The potential of imitation learning is in the idea of reducing design problems and reward shaping to just giving demonstrations as material. [38] Intelligent control is a fundamental category of control where different artificial intelligence methods are used to achieve tasks with complex goals, which includes fuzzy logic and neural networks [39]. Finally, one of the most prominent categories in learning is reinforcement learning which is also the core strategy of this thesis. Reinforcement learning will be discussed separately later in this thesis, as it shall be delved into in more detail compared to the other learning strategies.

When using learning strategies, there are different options for how the different components of the system are integrated into the model. These learning architectures give different approaches for handling sensors, control models and learning systems. End-to-end architecture takes multiple sub-components and turns them into a singular learning task. Contrary to a modular pipeline, this means that for example to move a robot, the framework maps all received sensory data directly into control commands. This gives the robot good adaptability and requires less rule shaping for the movement. [40] On the other hand since everything is directly turned to a singular task, end-to-end systems inherently create a black-box effect where the reasoning is not clear for the action. A modular learning architecture decomposes a complex task to smaller portions that are handled by neural modules. They can work by either being trained on different sections of the same problem, or a set is trained on the same problem combining their estimations. A modular architecture allows easier interpretability, but training these chained modules globally can be challenging at times.[41] In addition to using learning architectures directly, often learning-based systems are combined with classical controllers to create a learning-assisted control scheme. An example of this is to use Model Predictive Control and reinforcement learning together. The goal of which is to capitalize on both of their strengths. With this a smoother and more efficient algorithm can be developed, although it can be slightly difficult to train. [42]

Learning-based control methods of robots have plenty of positive properties, but it is also important to make notice of the limitations that come with it. When using adaptive control, the system is always sensitive to any delays happening within the computation

or actuation which is a real concern especially if heavier adaptation is needed. On top of this, highly dynamic environments can cause modelling complexity and reduce robustness. [37] Imitation learning has its own issues as well. This includes potential issues with matching the learners' abilities to that of the teacher, in addition to this kinematic replication can be challenging as the learner only sees the effect of actions that are taken. Lastly, since the learner is observing the teacher there are inherent challenges with noise in readings [38].

After reviewing different aspects of learning-based control techniques, it becomes easier to choose different learning architectures and modelling options. In this thesis the learning algorithm has simple control objectives but understanding how different areas change the used learning techniques makes the choices more justified.

2.4 Reinforcement Learning in Robotics

Situational choice making is crucial in robot control and is at its best when doing it maximizes the possible reward for the robot. For this a great method to achieve results and improve performance is using reinforcement learning. In reinforcement learning the robot is not told which actions are going to be right, but instead it learns them by comparison of the different actions and their respective rewards over time. This allows the robot to develop more so on its own compared to other methods such as supervised and unsupervised learning. [43]

A system for reinforcement learning is composed of four main elements, including its working policy, the received reward signal, the used value function and the environment's model, although the model is not mandatory. At the core of the system is the used policy, which by itself already explains the behaviour of the model. It can be more or less mathematically complex, but as a generalization it maps what actions are the most desirable when in certain states of the perceived environment. In RL, states represent the possible scenarios the learning agent can be in, and the actions are the possible responses the agent can take to affect its environment. The reward function defines the model's goal as the system receives reward values from the environment which it aims to maximise or minimize over the course of its runtime. The reward is the main component affecting the changes in policy, where receiving low rewards will cause

the change of the policy in the future when faced with the same situation. Whereas the reward function is the immediate information, the value function represents a longer scope where the system considers what is desirable over time. In a reinforcement learning model this means understanding which states are desirable. Even if the immediate reward would be low, the value function might still prioritize certain states if the common consequence of them is other states with high rewards. Although important as well, the value function is second in line when compared to the reward signal. The value depends on the reward, and its purpose is to maximize the received reward in the end. Still the general purpose of these models is to maximize the value, since it brings the most reward in the long run. This can be challenging as unlike rewards the value is not received directly from the environment but must be estimated and calculated over time. The last category, the environment model, is a sometimes-used element of RL systems. It replicates the environment which can then be used for planning and for predicting following states. It is not used in all RL systems and divides the systems into different categories of either model-based methods where it is present or model-free which work without it. [43]

Incorporating RL into robotics has been relevant in recent years and this includes land-, air-, and water robotics. Reinforcement learning does not rely on using a known a model for its dynamics, but instead gathering information from the environment and learning based on that [44]. The learning is achieved by trial and error, which makes it appropriate for solving challenging tasks, and self-learning requirements. RL is a subsection of machine learning where the environment is used as the source of information in regard to collecting a cumulative reward and maximizing it [45].

In reinforcement learning the system uses a type of action space to meet the goals and requirements in an allowed way within its constraints. A common way to categorize these spaces is into three categories: the Discrete Action Space, the Continuous Action Space, and a Discrete-Continuous Hybrid action space. In a discrete action space, the actions can easily be counted as they are simple commands and choices. Meanwhile in the continuous action space the choices are described within a range for example a degree of movement within a range. The last version combines the two where actions are chosen discretely, but the parameters for them are continuously specified. [46]

The reinforcement learning policy can be either On-policy or Off-policy. Differences between these two come from how they update their parameters and thus affect the robot's movement. The concept behind on-policy algorithms is the evaluation of the policy used to control the robot, whereas off-policy algorithms use one policy to collect information in order to improve their target policy. [47] The differentiation between on- and off-policies is closely tied to the trade-offs that need to be made when focusing on either exploration or exploitation. The trade-off between the two comes from the need to discover new areas and options ensuring the robot is finding the best choices globally. To achieve this, the system has to decide between taking new unexplored steps and using what it already knows from experience. By doing this the evaluation of finding something potentially better and using the currently best options is done. A common way of solving this dilemma is by using off-policy learning where exploration noise is added to the policy, which can then be independently evaluated. [48] On-policy learning has its own challenges when addressing the dilemma, as the learning and exploration cannot be separated from one another. As a result of this the balance between exploration and exploitation must be found within a single process. This is factored in as an exploration cost within the on-policy learning algorithm, and the algorithm will often learn to avoid danger to explore parts. [49]

In different applications, different algorithmic solutions of reinforcement learning have been used. These algorithms have some differences, and they include some of the following. The SARSA algorithm uses an eligibility trace for the learning of the system which allows it to learn the sequence the actions are taken in. The algorithm is on-policy which refers to it using the actual positions of the system to update the learning. Being on-policy, the behaviour of the system changes as time goes on. It has some challenges including the analyzation of the bias created by unidentified samples and the dynamic policy changing. [50] The Deterministic Policy Gradient (DPG), is another algorithm using actor-critic methods where the systems states are mapped to the action continuously. A policy gradient has a parameter vector which is used to determine a selected action in a current state, and it is represented by the action-value function's gradient. An advantage of using a DPG when compared to stochastic examples is not needing to integrate over the action space, which makes it need less samples,

especially in larger action spaces. [45], [51] Finally an important solution commonly used is Q-learning. The core of Q-learning is learning the values of actions so that the optimal actions can be taken [43], which will be reviewed more later as one of the core components of this thesis.

The usage of deep neural networks within RL is known as Deep Reinforcement Learning (DRL), where deep learning and reinforcement learning are combined. The combination of deep neural networks and reinforcement learning allows the mapping of more complex and high dimensional inputs like camera images to direct actuation. The neural networks act as approximators for either value functions and policies or one of them, which are then used to find the actions which bring maximum reward. This makes the framework good for robotics, although it does have its own setbacks including issues in sampling and crossing the reality gap. DRL has multiple areas of application with promising results including robotics. In robotics DRL has been tested on some of the following areas: locomotion, drone racing and -acrobatics, navigation and road following, industrial assembly, and multiagent co-operation. [52], [53], [54]

To better comprehend the applications of RL in robotics, some examples are considered. This includes exploration tasks [55], object picking tasks, manufacturing and vision based controlling [45]. To deepen the understanding of the applications, a single example is analysed in more detail. A study done by Zheng et al. [56] explored the possibilities of improving RL capabilities when working with tasks involving multiple stages. The purpose was to combat challenges involving local maximums and unnecessary exploration by adaptation based on three states of up-stage, stuck and ahead. Based on different states the exploration rate is changed to either lock in a successful strategy, escape stalling, or refine working strategies. The RL algorithm was a Markov Decision Process where states, actions, likelihood of changing states after an action, reward function and the importance of future rewards shape the framework and operate with the goal of maximising the received reward. To help with progressing, the researchers made the reward do a high jump in value when successfully completing phases in its tasks. The provided formulation was tested in a simulated environment to complete tasks of placing an object inside of a drawer and retrieving an object from a closed box. The study was able to produce efficiency in samples one-third to one-fifth of

that what was needed normally, while also working with an over 40% higher success rate in randomly generated environments. The method also had better stability where the learning did not get stuck on local maximums. Acknowledging this study demonstrates how RL can be used in robotics to achieve higher efficiency, and to help with performing tasks within robotics.

The usage of reinforcement learning brings some inherent problems with it. One of these challenges is crossing what is known the reality gap and moving from simulated versions to real-life experiments. The gap is caused by differences in how the real environment and equipment is represented within the simulations, which in the worst case can lead to failing controllers on real robots. To test the reality gap, training and verification is first done in simulation, after which real deployment is tested and the reward difference between them is identified as the reality gap. The reason sim-to-real pipelines are often used in reinforcement learning come from the nature of training and learning. Running simulators to train controllers reduces safety risks that arise from training trial and error on physical robots, and it is also faster. After the controller has been tested and validated, the sim-to-real transfer pipeline is done. For representing the robot and the environment, a less detailed version of them is usually preferred. This is done to address the computational cost that comes with highly realistic and accurate simulation environments. However, this can often cause clear issues when transferring to the real world. Some methods have been introduced to help developers in bridging this gap. These include Adversarial Reinforcement Learning (ARL), Domain Randomization (DR), and Transfer Learning (TL) as examples. In ARL the system has a protagonist and an antagonist framework where the antagonist is trying to minimize the protagonist's reward. This trains the algorithm to work under uncertainties. In DR some parameters of the environment are randomized with the goal of losing minimal performance and to maximize the reward. TL is more traditional in a sense as it is used by doing generalization between tasks in the simulated and real world. [57], [58]

Some other challenges related to reinforcement learning in robotics have been noted to be learning from a limited number of samples, explaining policies and actions, and formulating reward functions. These are only a few of the challenges related to RL and its practical applications and addressing them is important in robotic control. [59] When

applying to real-world scenarios, shortcomings can arise from aspects such as low-frequency control. Controlling robots is often restricted by the hardware available and the sensing abilities of the used sensors. This can restrict the robot's movement frequency, limiting the capabilities of reinforcement learning as it often needs a high-frequency input to compensate for estimation errors. [60]

3 System Description and Problem Formulation

In order to execute the desired goals and make testing as easy as possible, the system used in this thesis was created separately and set up for running the algorithms. In this chapter, the different aspects of the system are described and introduced.

3.1 Mobile robot platform

For the phototaxis setup, a mobile robot was needed. The robot is used as a base for movement, sensors and communication. After considerations, iRobot's Create3 was chosen as the base robot. Choosing the Create3 allowed the usage of pre-existing knowledge on motor control, which made the setting up easier. The robot was equipped with reflective markers to enable the usage of a Motion Capture (MoCap) system. It was purely used for plotting of the robot's paths, and not for the control in any way. A TSL2591 digital light sensor was attached to the Create3 via a Jetson NX Orin board, which was added to the robot. The Create3 does also have its own sensors for hazard detection, which were used as an emergency recognition software. By doing this the robot was made to back away after hitting objects, but this was done only as an emergency addition as the testing field is kept empty during experimentations so ideally the hazard detection should not be activated during the experiments.

The features of the Create3 are very suitable for the task at hand, and the simulation of the robot was also straightforward. The Create3 is a differential drive robot, giving it ability to move forward, backwards, and rotate. To move the robot around, it takes in velocity commands, more specifically in the form of ROS2 topic /cmd_vel. Within the topic, linear and angular velocity commands are sent which in turn are translated to the robots forward/backward and rotational movement. In simulation, it is useful that the robot has built-in wheel encoders which enable the robots odometry estimation based on its wheels.

As a differential drive robot, the Create3 follows well researched kinematic models that have been tested on similar robots in the past. Both wheels are revolving around the same centre axis, which give the robot a clear centre point. In this setup, changing the speeds of the left and right wheels will create either linear or rotational movement

respectively. The kinematic model used for a differential drive robot to command forward/backward and rotational velocity is defined as follows:

$$v = r\left(\frac{\omega_R+\omega_L}{2}\right) \quad (1)$$

$$\omega = r\left(\frac{\omega_R-\omega_L}{L}\right) \quad (2)$$

Where v is the overall linear velocity, ω is the overall angular velocity, ω_R and ω_L are the left and right wheel angular velocities, r is the radius of the robots' wheels and L is the distance between the wheel centres. The kinematic model is adopted from other studies implementing the same equation for their differential drive robots. [61], [62] Using this model and the cmd_vel topic, the Create3 moves in the following way. When an action is decided upon, the cmd_vel topic receives a new geometry_msgs/Twist message. The Twist message sends the linear and angular velocity in a format where both are represented by the commanded x, y, and z values respectively. As the Create3 is a non-holonomic robot, only the values for linear x (v) and angular z (ω) velocities are used. Linear y and z cannot be used as the robot cannot move sideways or change its height, and angular x and y are ignored as the robot cannot roll or pitch.

To connect the ROS2 message to the kinematic model, the robot uses inverse kinematics to translate linear and angular velocity to right and left wheel velocities. By inspecting the Equations (1) and (2), the velocities ω_R and ω_L can be represented as:

$$\omega_L = \frac{1}{r}\left(v - \frac{\omega L}{2}\right) \quad (3)$$

$$\omega_R = \frac{1}{r}\left(v + \frac{\omega L}{2}\right), \quad (4)$$

which is what the robot uses for its movement based on the sent messages. Since the kinematic model is a commonly used one, any robot following the same kinematics and controls could be used instead which is good for the expandability of the setup.

For the Create3, a pre-existing simulated ROS2 package is available, which was used in the simulated version of the setup. It creates all the same topics and same properties of the robot directly, which gives the best starting point for setting up the environment. Being nearly a circle with a radius of around 15 cm and having very straight forward

control mechanisms, the robot was a good fit for the setup. The Jetson NX Orin board was added to the robot as an additional processing unit. The purpose the board serves is the integration of the light sensor into the setup. The board was powered via the Create3's internal battery, and for communication it was set to use a matching ROS domain id. The connection was handled via a local Wi-Fi connection. Being a powerful board, in the future the Jetson NX Orin can support further expansions in addition to the one sensor that is currently used. The simulation is an idealized version of the Create3 where no noise, wheel slip, or other factors are considered. This creates a small gap between how the real-world is represented and what the simulation is, which is something that is kept in mind during the implementation. Outside of that, the simulated Create3 still follows the same kinematic models and uses the same ROS topics, which makes working between simulation and reality easier.

The systems architecture has three parts: the Create3, the Jetson NX Orin, and a laptop running Ubuntu 22.04. The laptop acts as a main workstation for the architecture, where the Q-learning algorithms and velocity commands are sent directly from it over the ROS2 network. All the data from the Create3 and the Jetson board is received via ROS2, and the information is used directly that way. The Create3 works as the systems actuator, demonstrating the movement and learning based on the algorithms running on the laptop. The Jetson board connects the sensor to the system and publishes its received data. The reason that the algorithms are being run on the laptop comes down to increasing computational efficiency, and also ease of use since checking logs, plotting and debugging is more straightforward on a local computer rather than a remote access one. To help with visualization the system architecture is shown in figure 1.

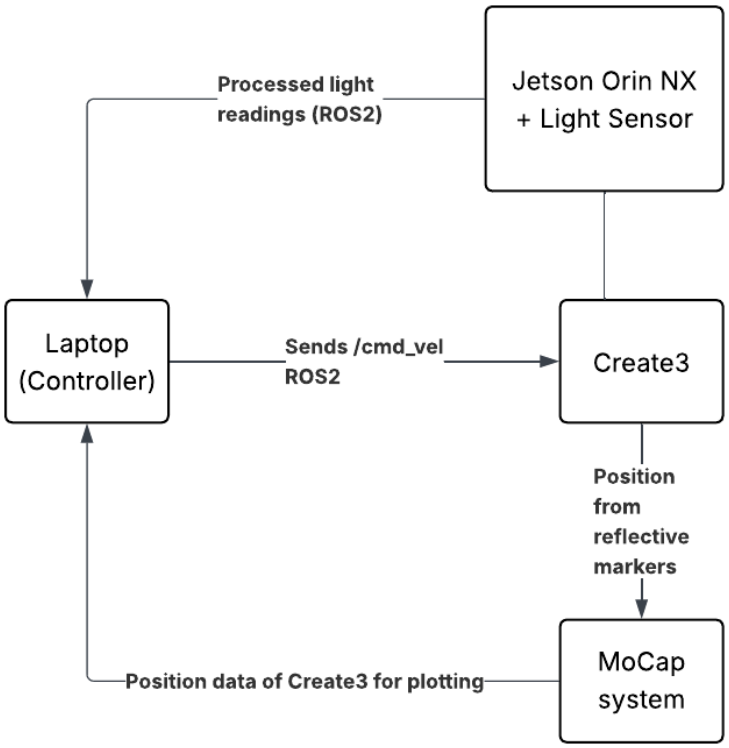

Figure 1: The system diagram

3.2 Sensor configuration and implementation

The robot uses a TSL2591 digital sensor which is connected to the Jetson board via an i2c connection. It produces lux readings which are used directly to translate the brightness of the robot's current position to a reward reading that is used within the learning algorithm. No smoothing or filtering is considered in the current setup. The decision to exclude smoothing and filtering of the sensor readings was made for two reasons. First, by using raw values of the sensor, the testing is done in a worst-case scenario. If working under these conditions, it makes it easier to claim robustness of the algorithm. Second, filtering would introduce a tuneable variable that has the potential to compensate for issues within the algorithm itself. By not filtering the value, the focus is kept on the used Q-learning algorithm, and the performance is not dependant on the filtering variables.

after some considerations. By not filtering the sensor readings the algorithm is forced to work under suboptimal conditions, which demonstrates the robustness of the

algorithm. By running the algorithm while using raw readings instead, the most amount of uncertainty is provided to the algorithm, and it is challenged more.

Different placements of the sensor were considered when creating the setup. The difference between having the light sensor place either in front, at the back, on the sides, or in the middle means a difference in what each movement will represent to the robot. Determining the appropriate sensor placement is discussed later in this thesis as results are created and different options are evaluated in more detail.

3.3 Experimental and Simulated environment

The experiments were done in a laboratory environment, with uniform lighting, even ground, and no obstacles. Here the robot has the least restrictions in its movement, and the experiments can focus on demonstrating the algorithm itself.

For the simulations, two slightly different environments are created where the reward gradient models differ slightly. In one version the light intensity readings of the physical light source were used to create a version of a simulated light source, and in the other a Gaussian bell curve function was used to simulate the reward source. This creates two environments where one is ideal and optimized, whereas the other is closer to reality and creates a slightly different gradient to traverse.

The ideal reward function follows a standard 2D Gaussian bell which has a defined center point. The formula for the reward function is constructed of three different parts: noise, orientation factor, and the spatial decay. The orientation factor alters the reward function to consider the orientation of the robot when calculating the reward. The orientation factor is defined in Eq. 5 as:

$$f(\phi) = \begin{cases} 1.0 & if\ orientation\ is\ not\ enabled \\ \max\left(0.1, cos\left(\phi - \theta(x,y)\right)\right) & if\ orientation\ is\ enabled \end{cases}, \quad (5)$$

Where $\theta(x,y) = \text{atan2}(y_c - y, x_c - x)$ is the bearing from the heading to the target center and x_c and y_c are the coordinates of the reward source. The noise can be expressed as:

$$n = \sigma \cdot \xi, \quad (6)$$

Where σ is the standard deviation of the noise and $\xi \sim \mathcal{N}(0,1)$. And the complete equation for the reward is shown in Eq. 7.

$$C(x, y, \phi) = f(\phi) \cdot (L_0 + \sigma\xi) \cdot \exp(-\lambda[(\mathrm{x} - \mathrm{x_c})^2 + (y - y_c)^2]), \quad (7)$$

Where $\exp(-\lambda[(\mathrm{x} - \mathrm{x_c})^2 + (y - y_c)^2])$ is the spatial decay, λ is the spatial decay coefficient and L_0 is maximum reward available. For the runs within this thesis, the function was used with the following numerical values: $L_0 = 40,\ x_c = 3.5, y_c = 3.5,\ \sigma = 0.1, \lambda = -0.5$, and the noise variance is $\sigma^2 = 0.01$

The second simulated environment was constructed based on actual light intensity readings within the experimental arena. To do this, the light sensor was moved in straight lines across the light gradient, collecting information on the intensity values along the way. These values were then used as pairs of distance and reward, which were used to fit a cubic spline through them.

As the light intensity values only reached a certain point in the arena, a large range of values in the simulation would be zero. For simulation purposes it is more useful to have some extension to the gradient as the purpose of this thesis is the realisation and demonstration of a gradient following algorithm, and exploring areas outside of the gradient is not within the scope. To tackle this, the simulated light source was given an extension in the form of a similar Gaussian curve that was used in the optimal reward function setup. This method enabled the robot to receive reward readings from further away, and the algorithm to have more valuable information to work with. The physical properties of the light source determine how reward is distributed across the workspace, and to visualize it better the comparison of the reward gradients can be seen in figure 2. In Fig. 2 the simulated light source's gradient curve is plotted with the optimal Gaussian reward function. For the Gaussian reward function, the best and worst cases are plotted since orientation acts as a factor for the received reward at each coordinate, and the band between the best and worst possible rewards is shown.

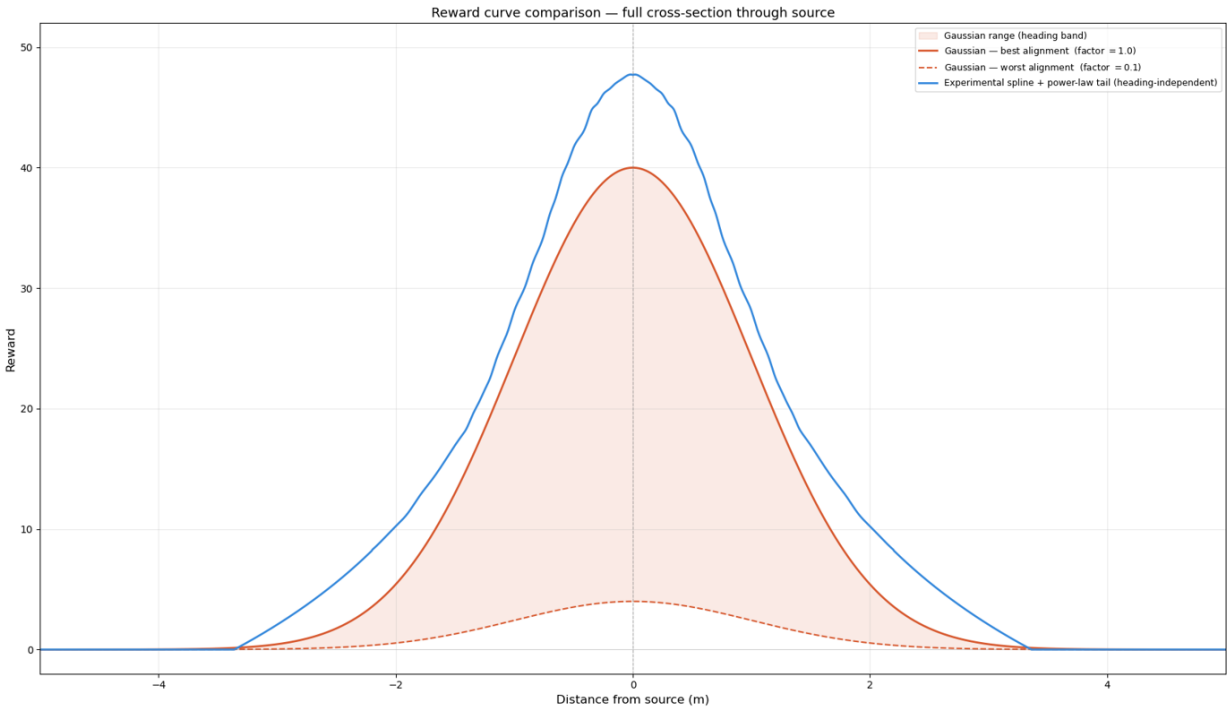

Figure 2 Comparison of reward gradients

As can be seen from Fig. 2, the reward gradients are somewhat similar when considering the best case for the Gaussian reward. One of the biggest differences between them is seen when comparing the peak value, as the experimentally measured gradient is not as smooth which is expected.

Even though the second simulated reward shape was constructed based off of experimental measurements, the curve is not exactly identical to what is actually used in the experimental setup. Ideally the gradient of the light source would be perfectly symmetrical, but due to restrictions in setup construction the actual light gradient is more elliptical in shape. This is due to the placement of the light source which required the light to be directed in an angle as it was connected to the corner of the arena, and that way the gradient would reach further out.

4 Learning-Based Phototaxis Methodology

The next step in the process was the development of the reinforcement learning framework used for the phototaxis experiments. The choices made and details about the algorithm are reviewed in the following chapters, which gives clarity on the methodology used within this thesis.

4.1 Reinforcement Learning Framework

The goal of the RL framework proposed in this thesis is gradient climbing to find a reward source, where the reward values come from a reward function in simulation and from sensor readings in the physical setup. In both versions, the reward values have the same role within the RL algorithm. The agent of the algorithm is either the simulated or physical mobile robot which is performing the task directed by the framework. When implemented correctly, the RL framework learns to choose states between active and passive based on the reward it is receiving, directing the robot towards its goal. The decisions made in this task are based on the previous ones to help the agent maximize the rewards and optimize performance, which makes it a sequential decision-making process [63].

The RL framework has no additional knowledge about its global position, and it only directly interacts with its environment which is either a light field or a testing arena. Only a singular sensor is used, so the framework works under partial observability created by the minimal sensing choices made. Using minimal sensing creates uncertainties as only the singular reward values are used to make all the decisions, which the framework has to account for by balancing the trade-offs in exploration and exploitation when selecting actions.

The approach chosen for the RL framework is Q-learning as the system is model-free with a discrete action space, which fits the approach as Q-learning is also suitable for the minimal sensing constraints presented within the system. The action space of the system consists of discrete commands where the system either forces a phase switch or uses natural switching between the active and passive states. The usage of sequential decisions with clear states and received rewards leads to the framework

translating well into a Markov Decision Process. Further details involving the framework are presented below in regard to the Markov Decision Process and Q-learning aspects.

4.2 Markov Decision Process Formulation

The Markov Decision Process (MDP) is a mathematical framework used in the modelling of decision-making where uncertainty and randomness is considered. In this framework only the current state and action of the system matter. In its simplest form, the MDP works as a discrete time step process known as Discrete Time Markov Decision Process (DTMDP). This version works in the following way: the system is being observed at different time steps, where a state is observed for the system. After this, an action is chosen for the system from a set of actions. Following that, a reward is given to the system, and a state transfer is performed at the next time step with a defined probability for transition. The DTMDP can be represented as:

$$\{S, A(i), p_{ij}(a), r(i,a), V\},$$

where S is the state space, i is the state, a is the action, $A(i)$ is the set of actions, $p_{ij}(a)$ is the transition probability, $r(i,a)$ is the reward, and V is a defined objective. [64]

For the formulation of the setup, reinforcement learning was used via action- and state spaces. In the action space of the algorithm, the controller makes a choice between binary actions $a_t \in \{0,1\}$ during usage where:

$a_t = 1$ is a forced switch where f_t is set to $1 - f_t$ and vice-versa

$a_t = 0$ has no forced switching and allows the default Markov switching to take place

Even when $a_t = 0$, switching may still occur on the phases based on the two-state Markov switching, where:

$$\Pr(f = 0 \mid f = 1) = \alpha \qquad \Pr(f = 1 \mid f = 0) = \beta, \tag{8}$$

which is only applied at decisions.

Based on the phases and a binary valued indicator for improvement of the algorithm, the state space is represented in a 4-state space where r and s represent the state space as:

$$r^{\uparrow} = \mathbb{I}[C_t > C_{t-\Delta t}], \quad s_t \in \{0,1,2,3\} \equiv (f_t, r_t^{\uparrow}) \tag{9}$$

Which creates the state space:

$$s_t = \begin{cases} 0 & f_t = 0, r_t^{\uparrow} = 0 \\ 1 & f_t = 0, r_t^{\uparrow} = 1 \\ 2 & f_t = 1, r_t^{\uparrow} = 0 \\ 3 & f_t = 1, r_t^{\uparrow} = 1 \end{cases} \tag{10}$$

The state space translates to four settings where the robot is either:

- Passive and not improving
- Passive and improving
- Active and not improving
- Active and improving

Being either in the passive or active phase, the robot is either passively reorientating to find a heading with better rewards or actively executing with more confidence towards its current heading. This results in the movement resembling a run-and-tumble type movement where the passive phase gives the robot a higher possibility to collect information regarding its surroundings.

The reward of the framework is tracked from a gradient which constructs of coordinates and headings in simulation and direct sensor readings in the experimental setup. Based on the values received, the change is shown as C_t and $C_{t-\Delta}$. Used in $r_t^{\uparrow} = \mathbb{I}[C_t > C_{t-\Delta}]$, the scalar reward is set by using the following formulas depending on the phase of the algorithm:

Passive phase ($f_t = 0$):

$$R_t = \begin{cases} 0 & if\ C_t > C_{t-\Delta} \\ -2 & otherwise \end{cases} \tag{11}$$

Active phase ($f_t = 1$), uses a memory variable:

$$R_t = \frac{R_{old}}{2} \tag{12}$$

The value of R_{old} is updated in the passive phase which results in bad outcomes lowering its value, carrying the effects to future decisions. When then changing to the active phase, the reward is carried over, when it is constantly halved. By doing this the reward encourages the active movement that the system has and teaches it to move actively towards the reward source. This was chosen as if instead a fresh start was taken every time for the reward, it could lead to more second guessing as the system has nothing to carry over its previous movements.

4.3 Q-learning Algorithm

Q-learning is one of the earlier inventions in RL. It is an off-policy temporal difference algorithm, where the optimal value for action is received from a learned action-value. In this off-policy temporal difference algorithm, a learned action-value function is used to receive the optimal value. In Q-learning the simplicity of updating the algorithm was a big achievement, and it showed convergence evidence early on. All that was needed from the learning algorithm for correct convergence was the updating of all pairs continuously. When using Q-learning, a negative reward of -1 is given when completing desired advancements, and a higher negative value when something unwanted and penalty-inducing is done. To better understand Q-learning, it is relevant to address the epsilon-greedy method. The epsilon-greedy method is a strategy used in reinforcement learning, where most of the time the best-known strategy will be taken while occasionally still trying random options as well. This creates a great balance between exploration of new information and exploitation of what is already known to be good. When using epsilon-greedy methods, the value of epsilon represents the percentage of time that a random action will be taken. Thus, the higher the value of epsilon, the more random the actions are going to be. [43]

The learning algorithm used in this thesis is also based on an epsilon-greedy Q-learning method. In the used algorithm the actions were selected using the following formulation:

$$a_t = \begin{cases} \text{Uniform}\{0,1\} & \text{with probability } \epsilon, \\ \arg\max_{a\in\{0,1\}} Q(s_t, a) & \text{with probability } 1-\epsilon. \end{cases} \quad (13)$$

The value of epsilon was of great consideration for the phototaxis algorithm, and after considerations, a few options for epsilon value usage were created. It is important to note that the epsilon-greedy method does have its downsides as well. One noted flaw comes from the guaranteed randomness of the method, which inherently will create the real possibility of sometimes picking the wrong choice even when the robot has learned a lot and is already picking the right choices [43]. The first suggested option is the usage of a static epsilon value, to guarantee a static version of the exploration/exploitation program. The second solution is using a slowly decaying epsilon value, so that when looking for the source, as time went on and expectedly the robot would find the source, the epsilon would start to go down, and random actions would not be taken as much. The third option is the usage of two modes, stay and explore, where based on the current reading of the robot's light sensor the epsilon value would be higher or lower. Keeping a higher epsilon value with lower light readings was considered as then the exploration would be based on the reward that the robot is getting. Out of these the first option was chosen as the main objective of this thesis is the realisation of the RL algorithm. It was decided that altering parameters during the run would potentially mask any flaws within the algorithm and manually fix issues, so a static value made more sense. By using a static epsilon value throughout the experiments, the exploration rate stays the same, and the behaviour can explicitly be tied to the Q-learning algorithm.

The Q-learning was managed via an action-value function which was initialized to zeros $Q(s,a) \in \mathbb{R}^{4\times 2}$. During operation the updates were done after readings from the light sensor were received as rewards R_t and the next updated state was received as s_{t+1}:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_{qt}\left(R_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right), \quad (14)$$

Where the learning rate and the discount factor of the equation are α_{qt} and γ respectively.

To better understand the step-by-step process of the algorithm, a flowchart can be seen in figure 3 which illustrates the process in low-level detail.

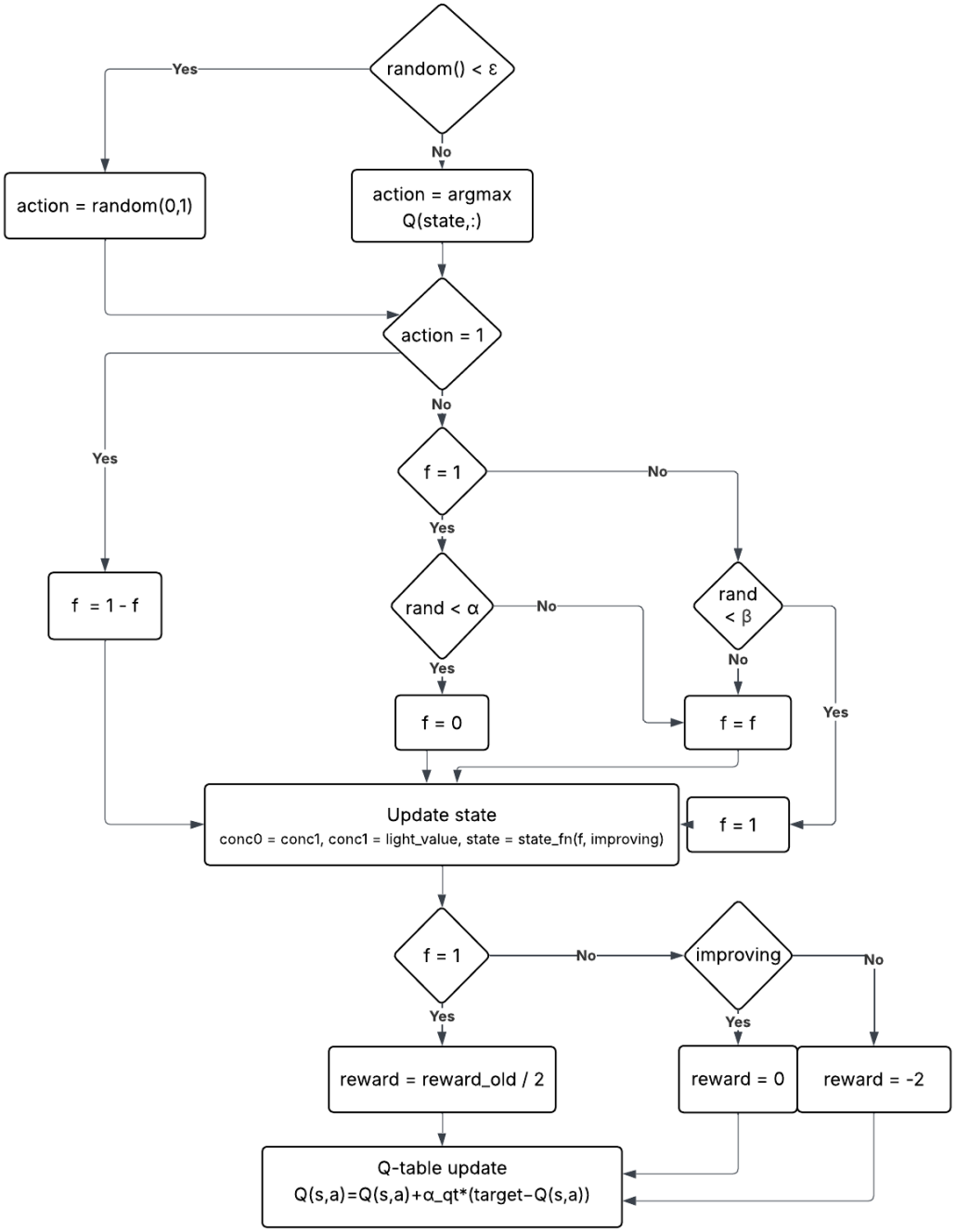

Figure 3: The step-by-step process of the learning algorithm

4.4 Control Policy and Implementation

To implement the algorithm with an experimental mobile robot, a flaw in the system needs to be addressed. Since the heading changes affecting the robot are completely random, nothing will create a significant pull towards the gradient in the experimental setup. The reward structure in the simulations uses the robot's current heading and calculates part of the reward based on the orientation of the robot in relation to the goal

coordinate. This behaviour is not possible to implement in an experimental setup with an actual light source as the reward. The robot receives only a reward reading of the light intensity, and no additional information about its coordinates, or the light source's position. When following only a singular light source reading, some considerations should be addressed. This includes situations where the same value can represent different points within the workspace where different actions should be taken. The convergence of the system is not guaranteed as it needs a sufficient gradient to follow, and the robot works based on the available states and the received reward it has. As discovered before the movement for phototaxis needs motivation to drive its orientation [18], and this needs to be addressed for the experimental setup. To address this difference in the reward structure, the embedded intelligence created by sensor placement is tested. By placing the sensor in the front section of the robot, the assumption is that a reorientation will change the reward value similarly to the simulated perfect gaussian curve reward structure. This creates enough information for the robot to learn when staying active and following its heading is beneficial and the gradient climbing behaviour is achieved.

To bridge the sim-to-real gap within the thesis, this setup was also tested in simulation where the orientation-based reward was flagged false, and instead the reward was taken approximately 14 centimetres from the robots current coordinate. In theory this should replicate the experimental setup and make testing more robust for the experimental environment.

4.5 Expected Behaviour and limitations

The expectation is that the robot will learn and perform a successful 'run-and-tumble' type execution towards the reward source. During the execution the states defined by the Markov decision process will be given to the robot based on its learning via the Q-learning based reinforcement learning algorithm. The robot will move along the gradient that a light source produces, ultimately converging to stay near the highest concentration spot of it, i.e. the brightest spot (or the goal area). It is to be expected that when staying around the goal area indefinitely, the robot will occasionally move out of the region if using a static epsilon value within the Q-learning.

The limitations in the experiments come from physical properties of the robot and also the limitations of the algorithm. On the physical side, one of the biggest limiting factors is the received information from the sensor. The sensors sensitivity and noise can cause faulty readings as the sensor gives jumpy readings even when standing still. Although the jumps are not massive, their effect is still relevant. Another limitation is to be expected from the used light source, as the stronger and easier to read the gradient is, the better the result will be. To help with this, the testing could potentially be done in a dark room. On the algorithm side, the limitations are a result of the choices made in the structure of the model. Having a smaller state- and action space contribute to the limited generalization of the information received. The usage of ROS communication will always produce some latency which is important to note, and in general commanding the robot has its own latency.

Using a single sensor is evidently one of the largest limiting factor of the work. Having only one sensor, the robot cannot perform directional guidance based on sensor readings. Instead of this, it has to learn to guide its direction via the passive and active phases it has. When forcing a switch between the phases, the robot is testing new directions and moving without certainty. Contrary to that, the usage of natural switching lowers the chance that the phase will be switched. By learning when to result to natural switching based on the rewards that the robot is getting, the robot learns the correct movement that the task requires. The sensor can also have noisiness within its signal which makes learning harder, as by using multiple readings the value could be averaged, and more stable information could be received.

5 Results

The results of the experiments done using the created algorithm are discussed in this section. The section includes the definitions for success in the experiments and what values are considered when evaluating the experiments. Results for both the simulated environment and the experimental laboratory environment are presented, to highlight differences in the setups. Multiple experiments were conducted in both versions to perform parametric evaluations and to find the best possible versions of each setup.

5.1 Evaluation metrics

When conducting the experimentations, specific aspects were looked at to determine the effectiveness of the algorithms. Some of the most important things to consider when evaluating the results are decided to be:

- Type of path taken
- Time to reach general goal area, and time to reach the exact goal
- The behaviour at the goal area
- Repeatability

In the context of the evaluation metrics, the general goal area refers to the most central area, or approximately 1 meter, around the highest peak of the reward gradient. The type of path the robot takes is highly dependent on its randomness caused by the number generators and communication methods used. Although a certain path and behaviour is never guaranteed, it is still good to visually inspect how the robot gets to where it is going. The parameters affecting the type of path chosen are mainly the ones directly connected to the robots linear and angular velocities. Higher base speeds will inherently create longer strides as the robot has time to move more, and lower diffusion noise will create smaller reorientations. Even though these are very important and effective, possibly the most important parameter affecting the robots trajectory is the step decision size. Giving the robot more time before making another decision has direct consequences on the movements length. On top of the movement, the robots

time taken is also always taken to consideration. The challenge of finding the perfect balance between effective exploration while maintaining proper hovering over the goal abilities proved itself to be difficult. As important as it is for the robot to find the source quickly, it is also crucial to acknowledge the importance of the behaviour at the source. On top of staying at the source, the robot has to also be robust against unlucky sequences it will take. Since randomness is a part of the algorithm, and the epsilon-greedy approach is used, the robot will inherently make insufficient decisions. During these it is important that the robot does not lose its track completely but instead manages to regroup and settle down again.

The evaluation metrics and methods were the same for both the simulation, and the experimental setup. As both were implementing the same robot, and also had versions implementing similar reward reading gradients, the hypothesis was that the movement will also be very similar.

5.2 Simulation results

The simulations showed promising results to begin the work. The simulation uses the predefined ROS2 package for the Create3 developed by iRobot. The launch files from the package provide a fully functional simulated version of the Create3 robot, with ROS2 topics published at launch. This launch was combined with a python code implementing the used reinforcement learning algorithm on the robot's published topics.

For the simulation results, a few different versions are looked at as data, in order to analyse the robustness of the algorithm itself. The first runs are done using an ideal reward function, using a gaussian curve to shape the reward gradient to make sure the algorithm works. After this, the second runs are done with a simulated light source closer to reality. The simulated light source is sending reward values as before however, the difference comes from the gradient it has. The gradient for the simulated light source was created by measuring real light intensity values of a lamp setup in the experimental version of the setup. By doing this, testing the algorithm becomes closer to reality, and the results that simulation gives are more likely to be feasible to duplicate in the experimental setup as well.

Since one of the determining factors of the Gaussian reward gradient is the orientation factor, the algorithm is experimented both with and without it. This is important, as altering between considering orientation and not fundamentally changes the behaviour of the algorithm. Based on the results the adaptation to an experimental environment will follow the same principles by using embedded intelligence. First the algorithm is tested with the orientation factor as true, after which test runs are done again with the orientation factor as false.

The results of the first tests done with the optimal gaussian reward function are shown in the following graphs. For evaluation, the following aspects were graphed:

- The trajectories of the robot and the reward area, which shows all five test runs at the same time and the trajectories taken. The reward source colouring is explained in the graph, and the gradient can visually be inspected. The trajectories are shown in figure 4.
- Graphs for both x and y position of the robot over time with a dashed line showing the target coordinates, to better visualize the robots evolvement towards and around the reward area. The x and y positions are shown in figures 5 and 6, respectively.
- Arrival time of robot, which shows the first time a ‘capture’ region is crossed which was determined to be a 1-meter radius around the reward source. The arrival of time is seen in figure 7.
- The exact time the goal coordinate was crossed, where 0.1 meters was determined to be close enough to have crossed the goal which is seen in figure 8.
- Retention after arrival, which shows the percentage of time the robots spends within the capture zone after arrival. The retention can be seen visualized in figure 9.

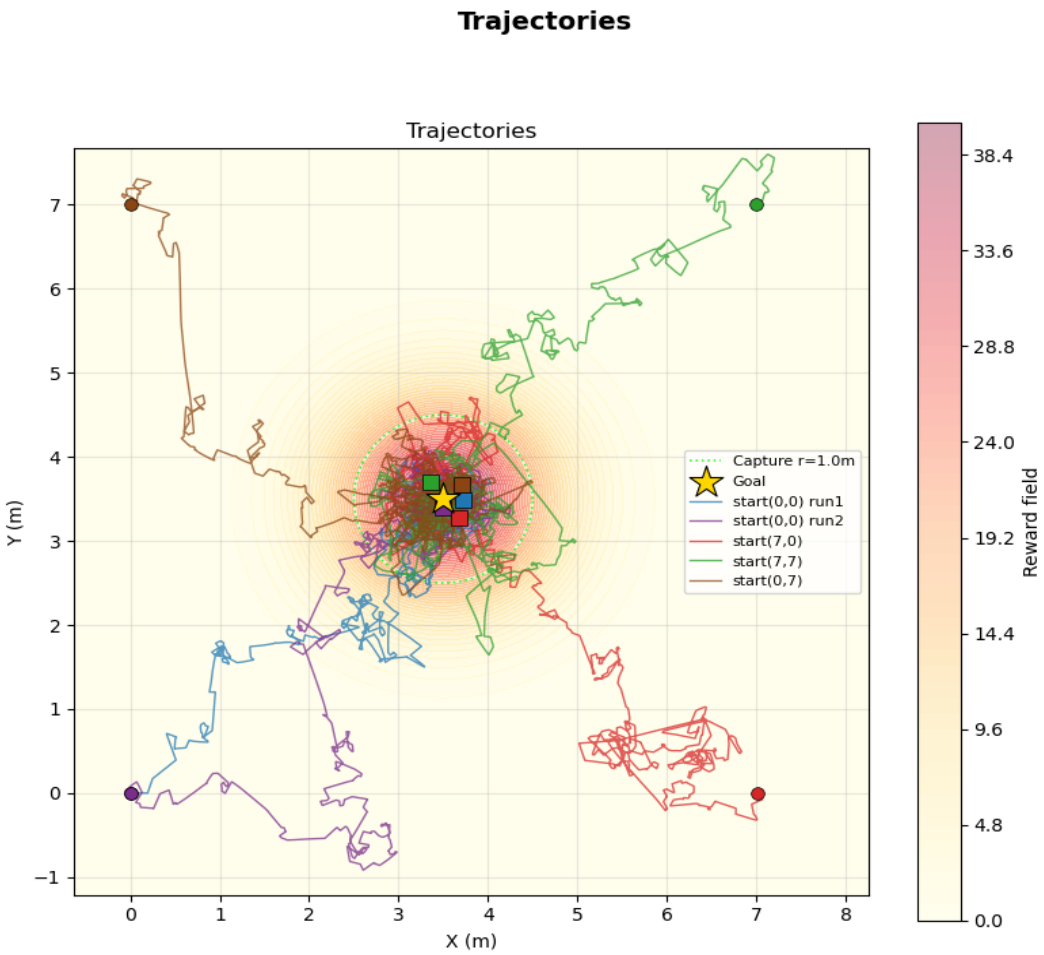

Figure 4 trajectories taken during five simulation runs

Five trajectories were chosen for the simulated runs of the thesis. The choices came down to testing different starting positions, and also different starting orientations of the robot. The same starting position was given to two runs from coordinate (0, 0), with opposite orientations (run 1 facing 0 degrees and run 2 facing 180 degrees). This tests the algorithms capability of recovering from worse readings early on as the first movement has a high chance of moving the robot in a negative reward direction. The other starting positions of (7, 0), (0, 7), and (7, 7) all had the same starting orientation of 0 degrees. This way a good spread of different circumstances were tested, and the robustness of the algorithm could be proved.

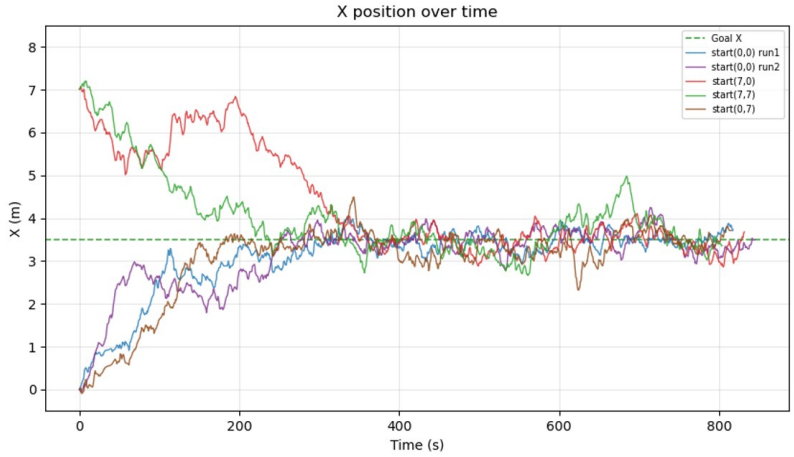

Figure 5 Track of x coordinate value over time

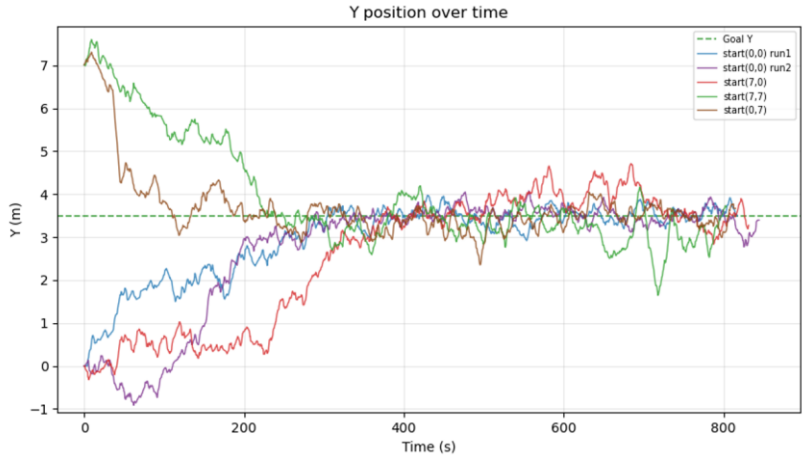

Figure 6 Track of y coordinate value over time

Tracking the value of the robot's x and y coordinate gives valuable insight as an addition to the visual inspection of the trajectories. All of the simulations were run for approximately 800 seconds, or around 13 minutes. This was done to give the robots not only enough time to travel and find the source, but also to demonstrate the convergence of the movement around the reward source centre.

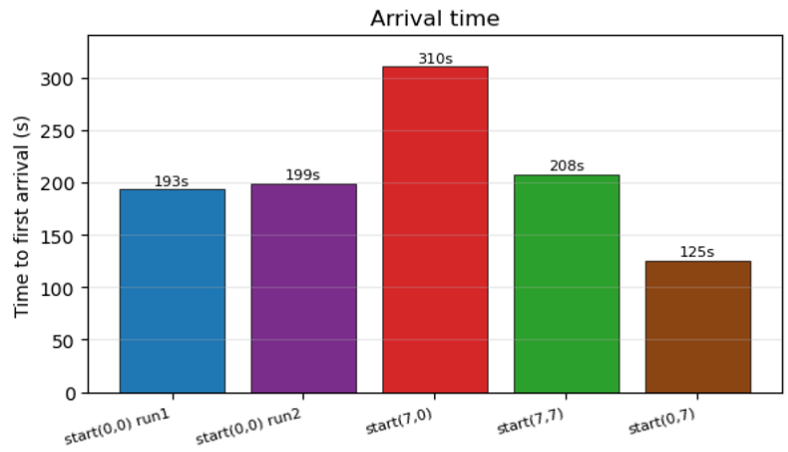

Figure 7 Time of arrival for each run

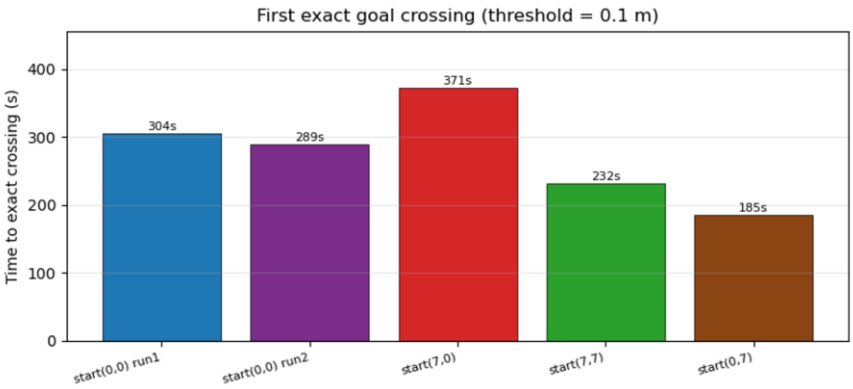

Figure 8 First time of goal crossing within 0.1m

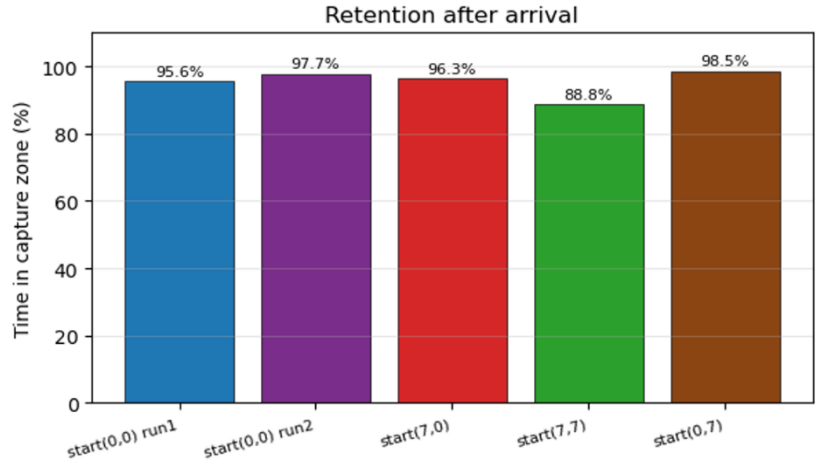

Figure 9 Retention at source after arrival

To help with result analysation, the arrival time, first goal crossing time, and retention percentage are good datapoints as they show information from the trajectories that might be hard to determine by visual inspection.

Inspecting the trajectories of the simulations and the bar charts of data gives two clear insights into the algorithm: the robot can find the centre of the reward source, and it also learns to remain nearby the centre. However, there are some caveats that can be noticed from the data graphs. The algorithm can produce very varying trajectories, which can occasionally lead to much slower convergence, higher exploration, and more time spent on suboptimal exploratory behaviour. The clearest example of this in the test runs is run 3 with starting coordinate (7, 0). After initially making good progress towards the source, it struggled to stay on the path and instead spent a lot of time exploring the same area which can also be seen in the time of arrival being noticeably higher than other runs at 310 seconds. The main culprit for this behaviour is the random reorientations the robot takes during the experiment. The randomness creates an inherent chance that continuous headings chosen are all bad and unhelpful. Although the algorithm is strong enough to correct those mistakes in the long run, it can create disruptions on the robots trajectory. Another caveat is the retention that happens close to the reward source centre. Even though the Q-learning teaches the robot which actions are favourable, the randomness creates an arrangement where bad decisions will always be made at some point. On top of randomness, staying near the reward centre creates harder readings for the Q-learning as smaller movements can create larger errors, as either passing the peak or looking away from it is much easier to do compared to being far away. This leads to deviations taken from the centre area, which need to be corrected by the robot. The algorithm did prove its robustness by handling these scenarios and still converging back towards the reward centre, and the retention percentage did stay high for all of the runs. Every single run was able to cross the exact goal coordinate with close enough distance, which shows the strongest source for the pull. There was still a clear worst run, being run 4 starting from coordinates (7, 7) which had a retention rate of 88,8% compared to other runs were the rate was < 96%. Even then the robot's ending position is very near the goal coordinate, which demonstrates the ability to recover from bad sequences.

To analyse the effectiveness of the learning capabilities of the algorithm, the actions the robot is taking were tracked. The two actions of natural and forced switching can be directly translated to the robots confidence in its current movement. Using action 0 and

resorting to natural state switching means that the robot is receiving positive feedback for its current actions and should continue executing in its current path, choosing action 1 forces a state switch where the robot reorientates and explores more. To visualize this confidence, the heading error towards the goal coordinate is compared to the preferred action used. By definition, the confidence to move should be higher the smaller the heading error is, so the passive state should be the highest during that time period. The plot for this can be seen in figure 10.

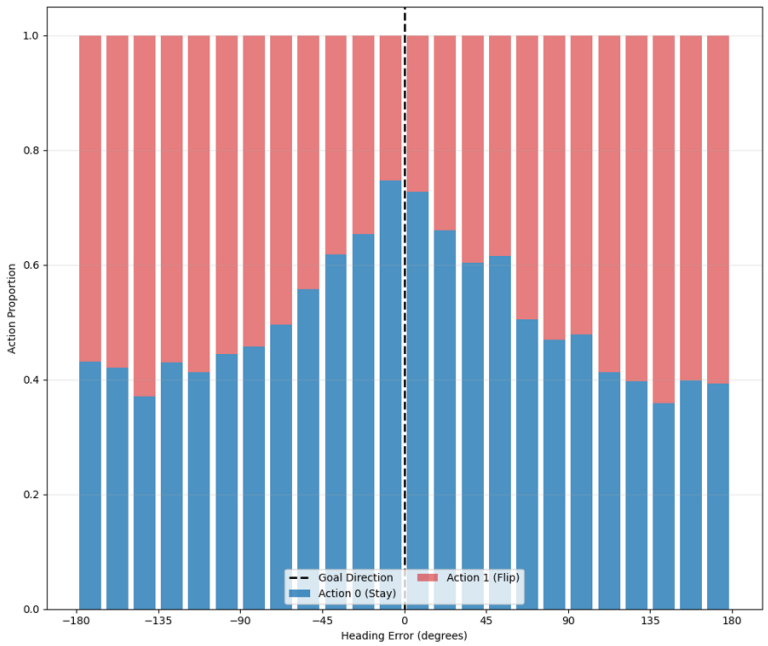

Figure 10 Distribution of actions chosen based on heading error

As it can be seen from Fig. 10, there is a clear peak for the action 0 the closer to heading error is to 0, meanwhile higher action 1 usage is seen on the larger heading errors of the robot. This corresponds to the expected behaviour and indicates the robots learning algorithm is providing useful results, providing high confidence when oriented towards the goal coordinate.

The next step is to test the algorithm, but this time without the orientation factor, letting the algorithm work its way towards the source purely based on its position in relation to the reward source. The same four starting coordinates are used for the runs, and the same information is plotted to allow direct comparisons between the two versions as all of the tests were run for the same amount of approximately 800 seconds or 13 minutes. The trajectories of these runs can be seen in figure 11.

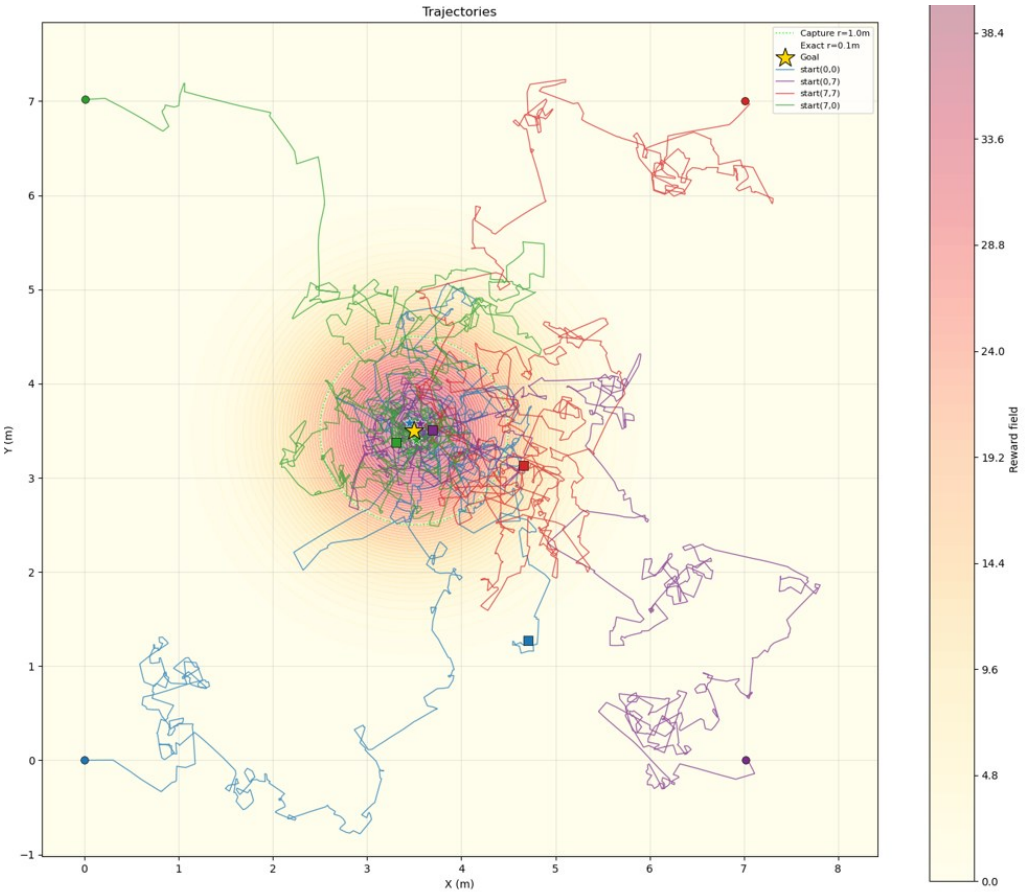

Figure 11 Trajectories of robots when orientation set to false

When compared to previous results, a difference can be noted as some of the runs end outside of the capture zone, and the runs do not seem as condensed as before. This alone begins to show the difference between using the orientation factor, but further analysis shall be conducted. The x and y coordinates of the robots over time can be seen in figures 12 and 13 respectively and as expected, the variations outside of the goal line in both is clear.

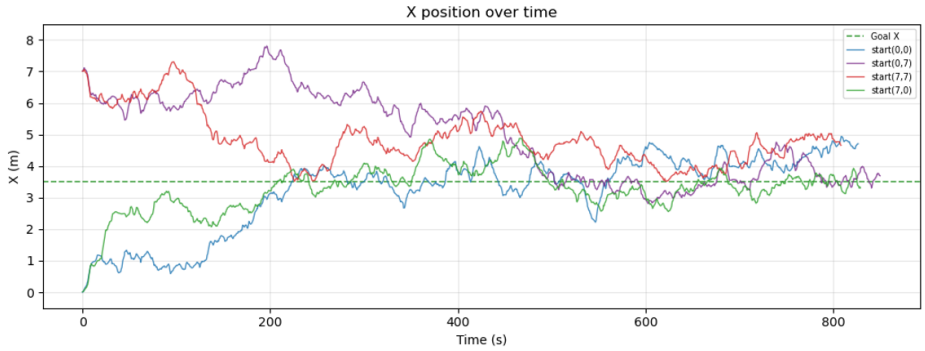

Figure 12 Position X when orientation false

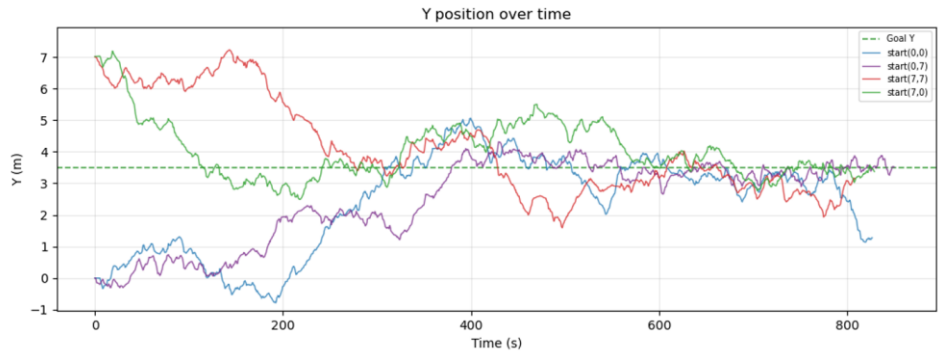

Figure 13 Position Y when orientation false

The trend of the runs can be analysed from these graphs alone as the robot always gets close to the source but does not stick too close to the actual goal line in either x or y plots. Using the same bar charts as before for the arrival time, time of first crossing and retention after arrival give the exact information needed to weigh the importance of the orientation factor. The plots for which can be seen in figures 14, 15 and 16 respectively.

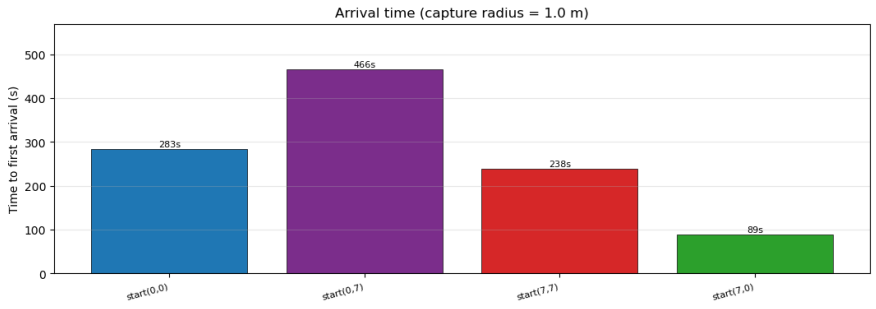

Figure 14 Arrival time when orientation false

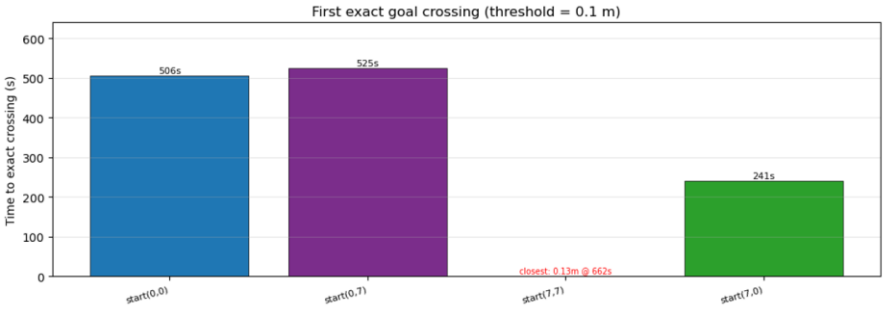

Figure 15 Time of first crossing when orientation false

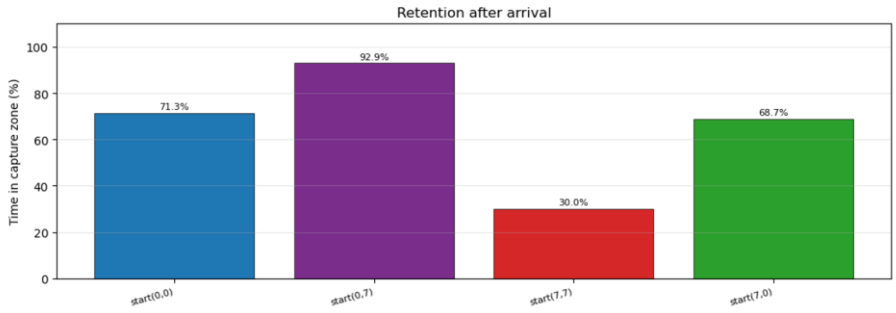

Figure 16 Retention after arrival when orientation false

Here it can be noted that not using the orientation factor has some noticeable limitations within the experiments. The arrival time is somewhat similar to the previous set of runs, where the first set of runs had a 207s mean $\pm$ 66.3s and the new runs have a 269s mean $\pm$ 155.3s, but the difference in first crossing and retention makes a clearer distinction between the two methods. This also shows the higher variance that can be noticed in the second set of runs. Not only were two of the crossing times a lot larger than any from the previous runs, run 3 with a start at (7, 7) never even crossed the goal coordinate and only got to 0.13 meters at the closest. In addition to this, the retention after arrival went to as low as 30%, which is drastically worse than the original runs. Interestingly enough, run 4 with a start (7, 0) performed exceptionally well in terms of reaching the capture area and crossing the goal coordinate. However, even that run lacked the retention after arrival. The actions chosen compared to the heading error of the robot is visualized again in figure 17 to realise how factoring in orientation affects the learning capabilities of the algorithm.

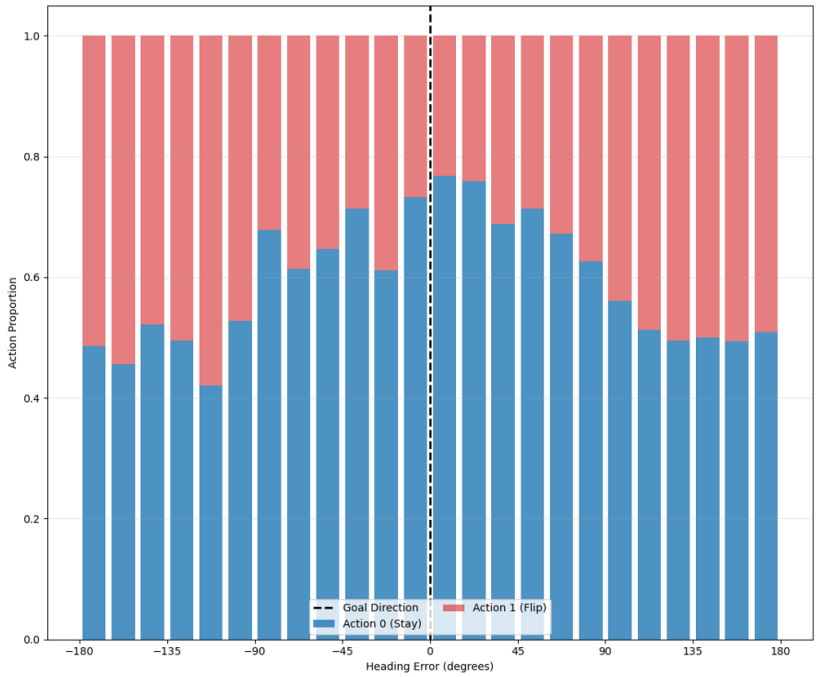

Figure 17 Distribution of actions over heading errors when orientation not considered

This time the distribution of actions over heading errors is not as distinct as when the orientation is considered. The performed runs were capable of having some general movement towards the goal coordinate which can also be seen in the action graph where the highest percentage of action 0 is seen when the heading error is the smallest. However, the main difference can be seen when looking at the general distribution which shows a much flatter curve with higher percentages of action 0 on a wider spread than before. Even though the performance was worse, even with the orientation off it can be noticed that the learning algorithm is functioning during the process.

Based on the two sets of experiments, it was concluded that the orientation should be taken into consideration when using the algorithm. The next step is to use this information to create a simulated version of the actual environment, via the simulated light source defined before. Since the orientation factor is now included, it has to be adapted into the simulated reward gradient which can only be received as pure positional values. To implement the orientation factor, embedded intelligence was used by placing the sensor in front of the robot. This way reorientations alter the received reward value, where orientating away from the source gives a lower value than before. In simulation this was done by taking the reward value of the simulated light source

approximately 14 cm in front of the robot, as that is close to the radius of the Create3 robot.

Multiple runs were performed to test the robustness of the setup, and once again important information is plotted for better visualization and analysation. The trajectories for the simulated light source experiments can be seen in figure 18.

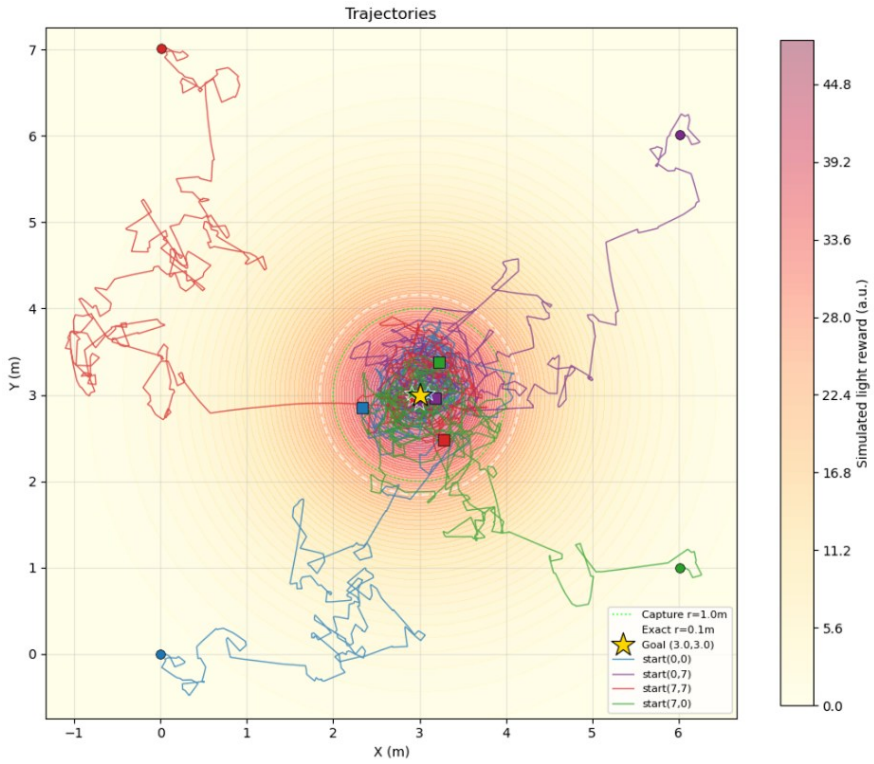

Figure 18 Trajectories of the runs with simulated light source

Using the described methodology to run simulated light source experiments more similar to the actual experimental setup show good trajectories converging towards the reward source. Further analysis can be done by inspecting plots showing similar information as before where the evolvement of x and y coordinates over time, time of arrival in the capture area and first crossing of goal coordinate, and retention percentage are shown. These can be seen in the following figures 19, 20, 21, 22, and 23.

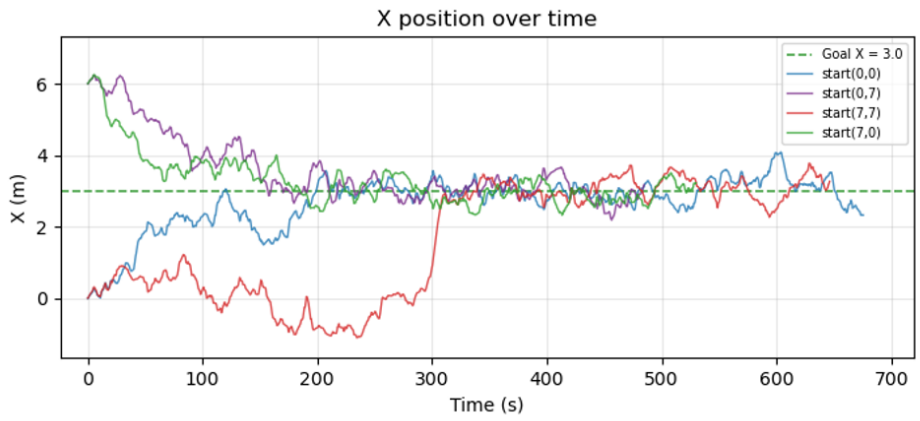

Figure 19 Evolvement of X over time with simulated light

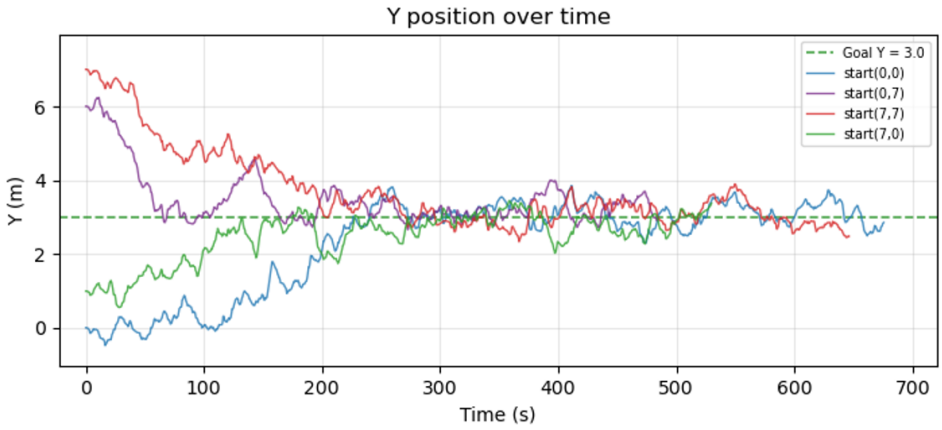

Figure 20 Evolvement of Y over time with simulated light

All of the trajectories converged to close proximity of the goal coordinates which is seen more clearly in the x and y plots over time All of the runs were not exactly as long, with some being around 500 seconds, but that was determined to be acceptable as the algorithm had already undergone multiple tests where robustness had been checked. The convergence was also strong enough that the runs were determined to be running well. The deviance from the goal coordinates is not large enough to raise questions about the algorithms performance, and the results are accepted.

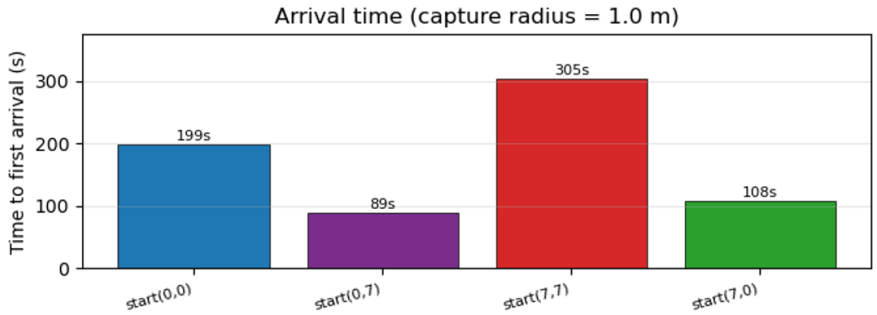

Figure 21 Time of arrival in capture zone with simulated light

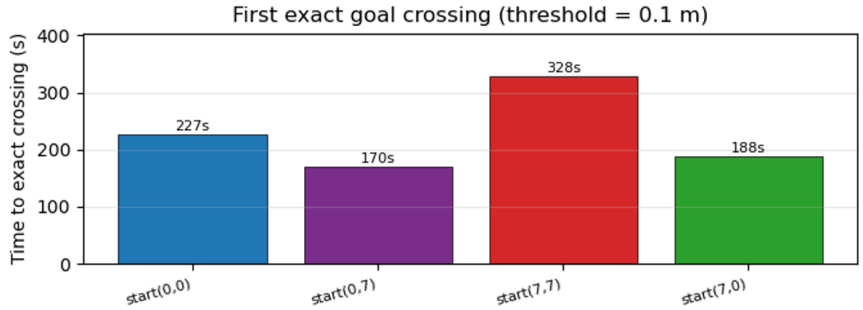

Figure 22 Time of first goal crossing with simulated light

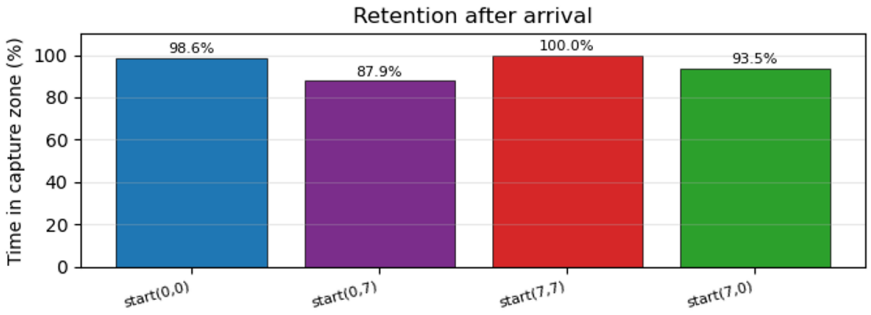

Figure 23 Retention of runs within the capture zone with simulated light

The effectiveness of the trajectories becomes clearer as the time plots shown above are analysed. Arriving at the capture zone and crossing the exact goal coordinates provided were both achieved well and in good time. In addition to this, the retention percentage after arrival showed good values for all of the runs. The most notable jump in time taken was run 3 with start (7, 7) which was 100 seconds slower than the next slowest run. On the other hand, that particular run had the single highest retention rate at 100%, and the starting point was also the furthest out of all the experiments. Finally, the action distribution was once again plotted and can be seen in figure 24. This time since the orientation is no longer considered in any way when calculating the reward, the heading

error was tracked as a separate variable giving no additional information for the robot about its movement.

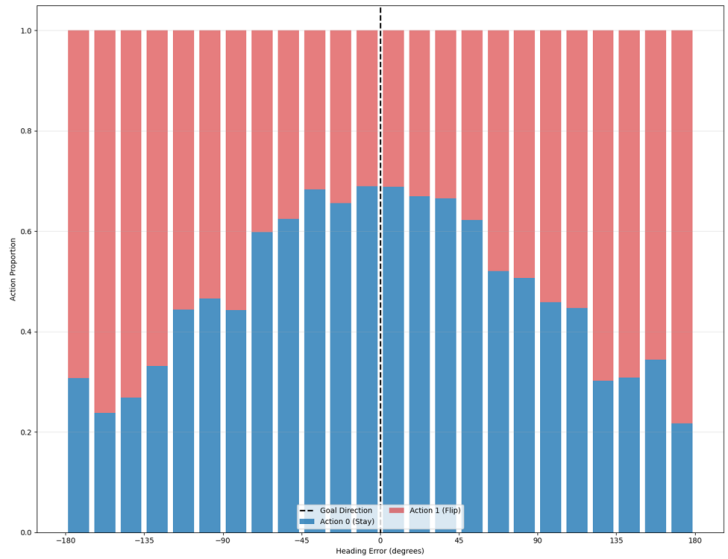

Figure 21 Distribution of actions over heading error using the simulated light source

Looking at the shape of the action distribution, it can be analysed that the action 0 is again chosen with the highest percentage when the heading error is close to zero, and that action 1 is preferred with higher heading errors. The graph shows a well distributed curve which further confirms the feasibility of the physical setup and the experimentations were concluded to be a success and solid proof of the algorithms capabilities.

As the simulations conducted showed good behaviour for the algorithm, the next step is to implement the simulated light source version to a laboratory experimental environment. By doing this, the algorithm feasibility can be tested, and actual usage is demonstrated. Since the simulated light source setup was made to be as close to reality as possible, the assumption is that crossing the sim-to-real gap should be easier when compared to directly changing from the perfect Gaussian reward function model to a real robot.

5.3 Experimentation results

The tests for the phototaxis were done in a drone arena of a laboratory. The arena is 6m x 6m x 4m in size, which is very suitable for the Create3. The arena is empty when used,

so that no obstacle avoidance needs to be considered for now, as it is outside the scope of this thesis. Pictures of the experimental arena and robot can be seen below in figures 25 and 26 respectively.

(a) side view of robot **(b)** Top view of robot

Figure 25 Pictures of experimental robot setup

Figure 26 The experimental environment where robot is performing phototaxis

A factor to be considered within the experimental results is the angle of the light source. As discussed before, the light had to be directed in an angle to cover a larger area of the

arena as there was no current way of placing the source directly on top of the arena, as can better be seen in the photo. This provides a reward gradient where one side has a smoother gradient, and moving away from the steeper side causes a more abrupt drop in lux readings. This was not considered a major issue as the algorithm's core functionality of following the gradient could still be tested. Even when crossing out of the steep side of the reward center, the algorithm would still receive accurate information about the reward going down. The magnitude of the drop is not as important as only the current and previous value are constantly compared. However, since travelling back towards the light source from the steeper side has a larger change in gradient it can create challenges for the algorithm as the robot receives a less informative gradient when compared to the smoother side.

Since there is no longer a precise goal coordinate, tracking the efficiency and successes of the algorithm were changed slightly. The highest lux reading available was tested before running the algorithm to determine the setups desired goal. Even though the reward gradient's shape is not symmetrical like in the simulation, the retention and crossing of exact goal was kept the same by tracking the radius around the highest light reading. This can lead to slight inaccuracies caused by the steep drop discussed before, but since the robot's goal is to arrive and revolve around this goal destination, a circular capture zone was deemed fitting. The retention was tracked in similar fashion to before where after crossing the capture radius for the first time the percentage of time spent inside the area is tracked. The chosen capture radius is again 1.0 meters, and the exact crossing radius is 0.1 meters as keeping the ranges the same makes the experimental runs directly comparable to the simulated runs before.

The trajectories, x and y positions over time and arrival times were all tracked in the same way as was in the simulations. Different starting positions and orientations were used to test the robustness of the algorithm, and its functionality on an experimental setup. Since the light source was angled and shining directionally from one corner to the other, the reward gradient was the cleanest and clearest in that same direction. In the trajectories plot seen in figure x, this can be visualized as the light shining from coordinate (6, 6) towards the coordinate (0, 6). The main affects this has is where the tests are done from. Since the purpose is to demonstrate the gradient-following

abilities, the starting positions were chosen closer to the path the light source was creating. Hence why no tests were started from position (0, 0) for example, the gradient just is not visible enough there because of the light source.

A few test runs are presented as a proof-of-concept to examine the usability of the algorithm in an experimental environment. By doing multiple runs, the robustness is tested and the behaviour is confirmed. The trajectories of the test runs can be seen in figure 27

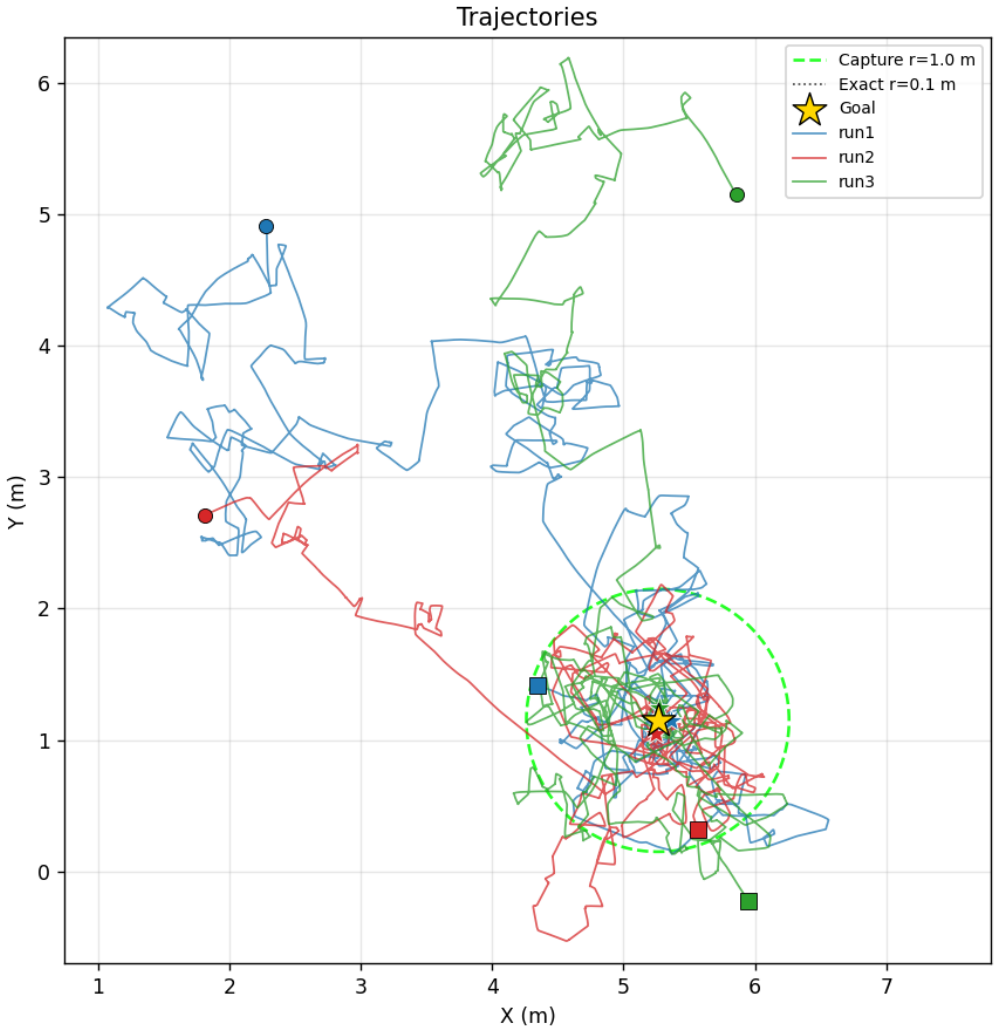

Figure 27 Trajectories of the experimental runs

The trajectories show how different runs starting from slightly different positions all found their way to the light source in the end. Visually speaking the trajectories are all very different as some take much larger detours and others traverse more directly to the source. All executed trajectories successfully reached their objective, as they all

converged to the desired position when evaluated over the full runtime. Compared to before the gradient is not coloured in on the trajectory map, as the exact gradient cannot be mapped as a uniform circle. The difference in gradient discussed before is also a contributor to why some runs required more exploration before finding an adequate direction. The progress is again slightly easier to visualize when looking at the position graphs of x and y in figures 28 and 29 respectively.

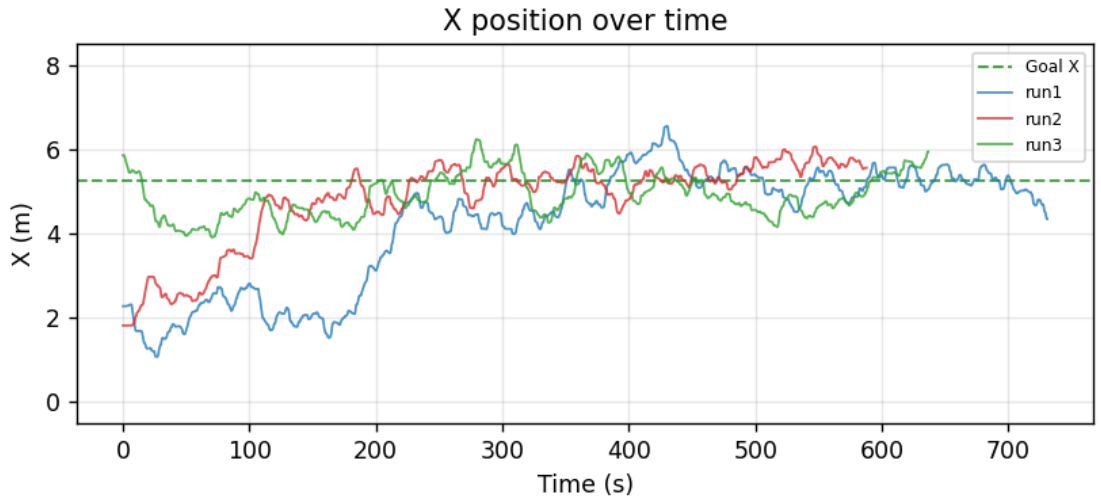

Figure 28 Position x of runs over time

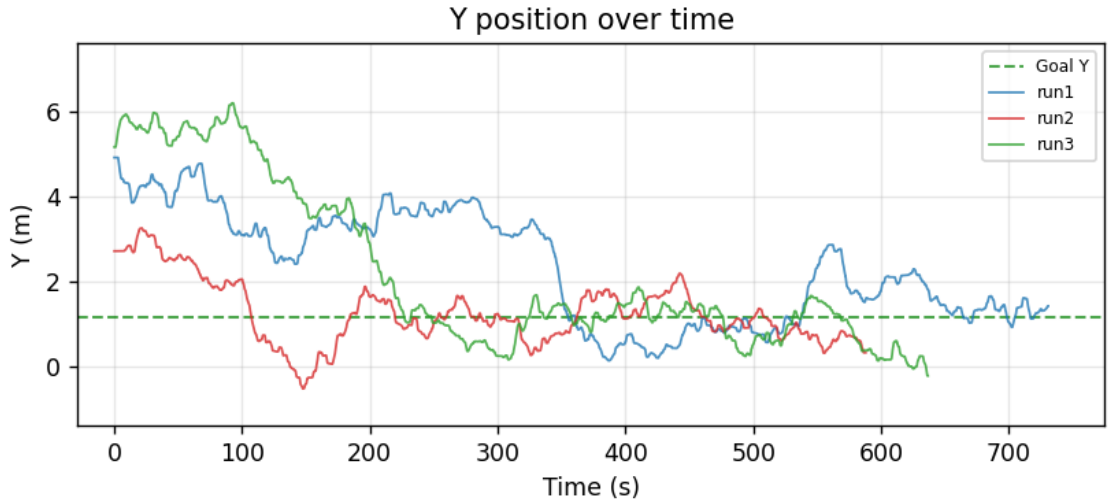

Figure 29 Position y over time

The experimental test runs were not exactly as long as the simulated runs. The cutoff is earlier at around 700 seconds (compared to the previous 800 seconds), where some runs were finished even earlier than that. This is partially due to the used gradient which is smaller than the practically infinite simulated reward gradient. Caused by this, the distance that needs to be travelled within the runs also become shorter.

The evolution of both x and y coordinates in all of the runs resolves close to the goal coordinates. The runs are not staying as close to the goal coordinates when compared to the simulations which is something that was expected. The change of gradient and physical inconsistencies were thought to be the main culprits. A clearer picture of the runs is painted by analysing the arrival times within the capture and exact radiuses, and the retention of the runs. The arrival time in the capture radius and the first exact goal crossing can be seen in figures 30 and 31 respectively, while the retention percentage after arrival can be seen in figure 32.

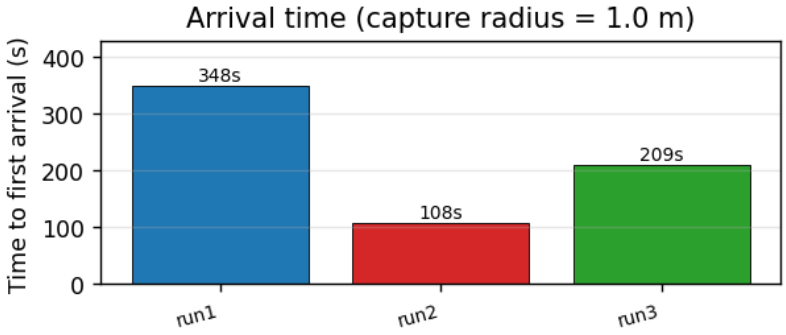

Figure 30 Arrival time within the capture radius in the runs

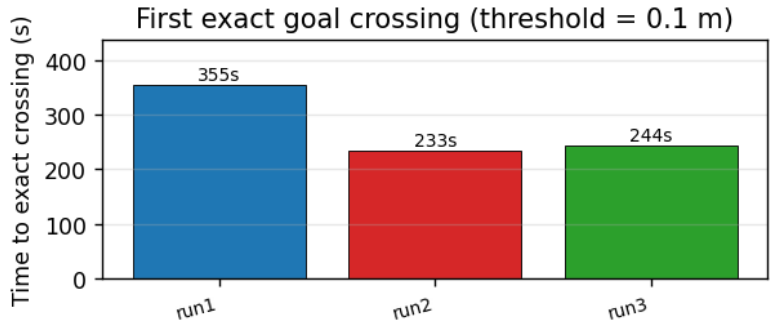

Figure 31 Time of crossing the exact goal radius in the runs

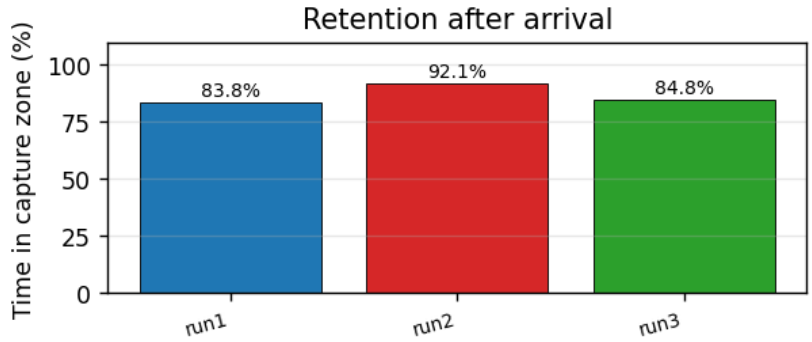

Figure 32 Retention percentage of the runs within the capture radius

Two important distinctions can be made from these graphs. Every run managed to cross the exact radius which means that the algorithm was always able to follow the gradient all the way to the reward source center. The retention of the runs is above 80% for all, which shows that the behaviour close to the reward source center is as desired, where the robot stays close to the center. To confirm the learning capabilities are still working in the experimental setup, the action distribution over heading error can be seen in the following figure 33.

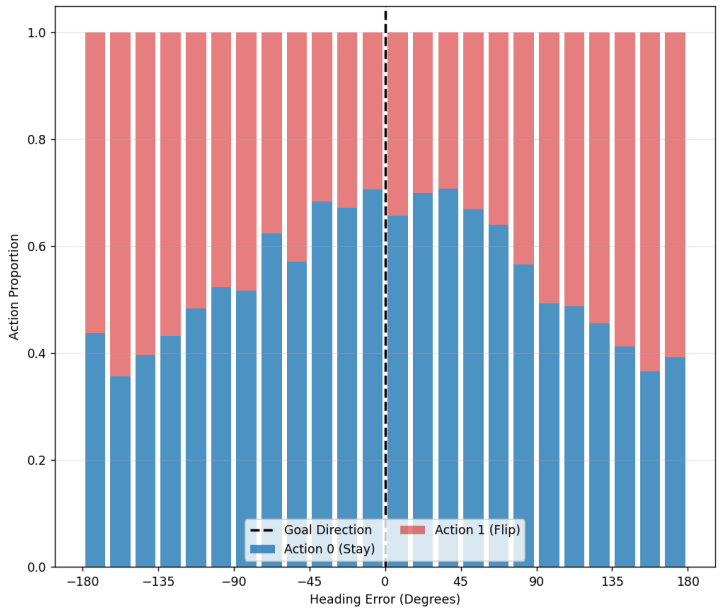

Figure 33 Distribution of actions in relation to heading error of runs

As can be seen the algorithm still has the highest action 0 percentage when the smallest heading error is calculated, which confirms that the algorithm still learns when to flip its states.

Completing the runs in the experimental environment demonstrate the algorithms feasibility within the real-world which opens the discussion for applications in the future and shows the robustness of the algorithm.

5.4 Discussion of results

As predicted, the results for both setups looked somewhat similar. The created algorithm did its core function well which is to follow a reward gradient. The biggest issue in setting up the environment was the actual reward gradient's visibility and usability. The exploration capabilities of the algorithm were limited as outside of

determining increase/decrease within a gradient, the algorithm does not have environmental information encoded within the states. While exploring, no information that would inform the robot of already travelled paths is stored. Despite this, while traversing within a gradient, the algorithm consistently produces movement towards the reward source center.

This was better realised in the simulated setup using the gaussian default reward function, which produced evenly spread reward values across the entire workspace. In this version the robot was able to find its way to the centre of the reward source from multiple starting points and orientations consistently.

Based on comparisons done from the results, a conclusion can be made that the simulation provides better results. This is something that was expected as physical experimentations always include physical constraints and differences. Things affecting the experimental runs include the battery percentage of the robot, which is constantly changing, communication latencies and errors that can happen when using ROS2 nodes and topics wirelessly, and noise that can be received from anything within the environment that can affect sensor readings. As seen in figure 26 the experimental environment had the laboratory lights kept on. This was a result of testing where turning all ambient lights off resulted in worse readings when moving the robot along the light gradient. Although the sensor was working correctly and had desired qualities, improving the readings when working in the dark should be considered. This would involve tuning the sensor and interpreting its readings.

To improve the experimental results, many changes could be implemented. This includes filtering out light readings and also using different light sources. As discussed before, the available gradient has a heavy influence on how well the algorithm runs and using larger more evenly spread light sources is very likely to produce better results. In addition to this, filtering out light readings by using methods like sliding window could make the information received from the light source more useful. By averaging out the values over x amount of time, the jitteriness caused by noise could be reduced resulting in more accurate readings which are more informative for the robot. Filtering out light values does have a downside which is the inherent latency it introduces. In a

consistently updating RL system, introducing latency is not ideal, but if the quality of the values increases enough it could be worth it.

One of the better implications for the algorithms functionality are the plots depicting actions taken when compared to the heading error of the robot towards the goal. By showing the distribution of action choices based on the heading error, the learning algorithms contribution can be visualized and understood. Although a purely reactive algorithm could potentially achieve results in following a gradient, the plots tell us that the usage of a learning-based framework correctly orientates the robot to move towards the goal coordinate. The action plots also further confirm the significance of placing the sensor in front of the robot. Directional knowledge was achieved without using the robots positional information within the algorithm, which is crucial for the feasibility of the algorithm. Although the action plots show a clear pattern where staying in the same state is preferred when heading towards the coordinate, it can be noticed that the algorithm never chooses the action 100% of the time. This comes down to the inherent randomness of the algorithm provided by the epsilon greediness used within it. Using a decaying epsilon value could potentially increase the efficiency of the algorithm as changes would not be forced a set amount of time.

In addition to altering the epsilon, testing should be conducted for different parameter combinations. Changing the values of α, β, and γ directly affects the expected behaviour of the algorithm. Altering the probability of state switching via α and β could result in less redundant exploring with more determined movement and different combinations of the parameters would result in very variable results. As it currently stands, the algorithm uses a low γ value which corresponds to focus on immediate reward. Increasing the parameters value to focus more on long-term rewards has potential to alter the algorithms functions significantly.

Similar conclusions to previous studies were made as the orientation factor was deemed a crucial aspect of the algorithm. The robots heading needs to be controlled in some way, as having no direction for reorientation causes each movement to be almost entirely random which results in insufficient exploration.

The amount of experiments analysed and plotted in this chapter is not entirely corresponding to what was done during development. The results shown are runs that were done in a set environment, but plenty of testing was done with differing parameters, starting positions, and gradient formulations. In the experimental environment the runs shown here were ones that were done in one of many testing days where the light was angled exactly in the same way and the ambient lighting was the same. Because every time the setup was rebuilt there was not a 100% guarantee that the setup is exactly the same and so only the chosen set is presented as proof of concept. The testing involved more experiments where the light was placed differently, and the algorithm was run with slightly differing parameters. Since majority of the time the results were promising and the right behaviour was noticed, the chosen set of runs was decided to be enough to act as a proof of concept for the algorithm. The same can be noted about the simulations which had even more rigorous testing in different scenarios.

6 Conclusions and future work

The work in this thesis demonstrated the usage of a Q-learning based reinforcement learning algorithm to perform gradient following positive phototaxis on a mobile robot via minimal sensing with a simple state space. The Q-learning worked by performing temporal sensing where reward values at different positions in space were compared. The demonstrations included four different scenarios where the algorithm was evaluated in both simulation and an experimental setup. The four different scenarios tested were simulated reward function with and without the consideration of the orientation, simulated light source and -sensor setup with values based on real-world data, and the physical experimental setup.

The results of the experimentations showed that the algorithm was able to follow a gradient successfully and keep hovering around the light source. The findings point to the main culprit of success being the structure of the received signal, where stronger gradients resulted in more consistent results. Including the orientation factor within the reward structure was found to be highly important for the algorithm as it allowed the movement to have intention in its direction and without it, the efficiency of the algorithm worsened noticeably. When performing the experimentations in both simulation and physical setups, the behaviour was comparable and the algorithm was able to follow the provided light gradient in different scenarios. The best results were received via simulation, however implementing the algorithm in the physical setup confirmed that even under the environmental constraints brought by crossing the sim-to-real gap the behaviour is still retained which shows robustness of the algorithm. Some notable uncertainties affecting the experimental setup included the uneven light gradient, changes in battery percentages, and sensor noise, which all had their own effects on the experiments.

Since the achieved behaviour was accomplished by only using minimal sensing and a four-state space with two movement phases, it demonstrates that even without more complex sensor technology like LiDARs, the proposed behaviour of gradient following can be accomplished by using RL as an alternative. The robustness of the thesis' approach was confirmed by sim-to-real successes where the algorithm performed in

real-world uncertainties. The approach highlights the computational simplicity required, and the minimal hardware required makes the approach scalable and accessible with resource constrained setups.

6.1 Future work

The framework provided in this work has great potential when considering its future directions. Since the algorithm is only affected by the gradient that it is receiving, it creates opportunities for expanding the applications that the algorithm is used within. Although demonstrated in this thesis with a light source to perform phototaxis, the source for the gradient can be practically anything as long as a gradient exists. This includes other taxis tasks such as the ones discussed early in this thesis like thermotaxis and chemotaxis. Testing this approach is fairly straightforward as it would only require a new source and sensor for which the same algorithm could be used.

To expand even further, the usage of multiple agents should be considered. By combining information received from multiple robots, the quality of the reward gradient could be increased, and the effectiveness of the algorithm could thus be enhanced. Combining local and global knowledge about reward areas opens the discussion for implementations that involve local maximums. The current implementation of the algorithm does not have the capabilities to know whether a reward center it finds is the actual goal that it is supposed to reach, since if moving away from a point starts to decrease the reward, the algorithm will guide the robot back towards the peak of that gradient. As it currently stands, the robot would have to travel far enough from the local maximum by pure chance where its reward reading would then start to climb again. The idea of using multiple agents could help combat this issue where the other agent would be able to confirm this information that a greater reward exists. Implementing multiple agents is slightly more complex as the shared reward knowledge would need to be implemented into the algorithm.

In other applications, the local/global maximum problem can be shifted to the identification of multiple reward sources from different source types. This could either translate to the usage of conflicting sources, or multiple positive sources. For this, the rewards from different sources would have to be stored separately in some way, and in

the case of smaller and larger positive sources, the context would have to be considered. By using the context of the rewards, a smaller reward reading from a different source type could be deemed more valuable which would make its overall reward higher. By achieving this the robot would be able to navigate more efficiently and consider more variables when performing its exploration and source seeking. This approach would involve installing different sensors and using different sources, which is fairly straightforward to try, as then the rewards for each source could be made correspondingly.

One of the natural directions that should be considered in the future is the involvement of deep learning techniques that were also briefly discussed in the literature review. The usage of deep Q-learning gives the opportunity to use more complex state representation instead of the four states used in this thesis, and to incorporate aspects such as obstacle avoidance better. [65] In this thesis the usage of deep Q-learning could improve the state representation providing more information to help with exploration abilities or sensitivity to noise. Incorporating deep learning has the potential to address some of the challenges related to minimal sensor information available, and the discussions around reward structures would still be applicable which makes it a natural upgrade.

As one of the most prominent forces in technology right now, generative AI is an aspect that should also be considered in the context of this thesis. Although not the biggest limitation in this thesis, the usage of Generative AI has been noted to be a good tool for addressing the sim-to-real gap. This is achieved by generating environments that are intricate and helpful for training of better actions and strategies. Specifically in RL the usage of Generative models could be adapted to enhance reward structures which is closely related to subjects of this thesis as well. On top of this, generative models are noted to be applicable to data augmentation where features and data could be generated from limited sensor information. [66] For this thesis, the connection to generative AI comes from enhancing the available sensor information of the system. If possible, to improve reward structures, and help with experimentation by creating differing gradients, the testing of the algorithm could see improvements. This is also something to consider when working with minimal sensing and partial observability.

References

[1] S. Cebollada, L. Payá, M. Flores, A. Peidró, and O. Reinoso, 'A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data', *Expert Systems with Applications*, vol. 167, p. 114195, Apr. 2021, doi: 10.1016/j.eswa.2020.114195.

[2] M. Misaros, O.-P. Stan, I.-C. Donca, and L.-C. Miclea, 'Autonomous Robots for Services—State of the Art, Challenges, and Research Areas', *Sensors*, vol. 23, no. 10, p. 4962, May 2023, doi: 10.3390/s23104962.

[3] H. Ardiny, S. Witwicki, and F. Mondada, 'Are Autonomous Mobile Robots Able to Take Over Construction? A Review', *International Journal of Robotics Theory and Applications*, vol. 4, no. 3, pp. 10–21, 2015.

[4] F. Rubio, F. Valero, and C. Llopis-Albert, 'A review of mobile robots: Concepts, methods, theoretical framework, and applications', *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839596, Mar. 2019, doi: 10.1177/1729881419839596.

[5] M. Rybczak, N. Popowniak, and A. Lazarowska, 'A Survey of Machine Learning Approaches for Mobile Robot Control', *Robotics*, vol. 13, no. 1, p. 12, Jan. 2024, doi: 10.3390/robotics13010012.

[6] M. Rosenfelder, H. Carius, M. Herrmann-Wicklmayr, P. Eberhard, K. Flaßkamp, and H. Ebel, 'Efficient avoidance of ellipsoidal obstacles with model predictive control for mobile robots and vehicles', *Mechatronics*, vol. 110, p. 103386, Oct. 2025, doi: 10.1016/j.mechatronics.2025.103386.

[7] E. Baklouti, N. B. Amor, and M. Jallouli, 'Reactive control architecture for mobile robot autonomous navigation', *Robotics and Autonomous Systems*, vol. 89, pp. 9–14, Mar. 2017, doi: 10.1016/j.robot.2016.09.001.

[8] M. A. Taleb, G. Korsoveczki, and G. Husi, 'Automotive navigation for mobile robots: Comprehensive review', *Results in Engineering*, vol. 27, p. 105837, Sep. 2025, doi: 10.1016/j.rineng.2025.105837.

[9] A. E. Adegunsoye, B. Ubochi, J. Macaulay, and K. F. Akingbade, 'A hybrid gradient climbing algorithm for a swarm robot-based gas leak detector', *IJRA*, vol. 13, no. 3, p. 255, Sep. 2024, doi: 10.11591/ijra.v13i3.pp255-263.

[10] M. Jabeen, Q.-H. Meng, T. Jing, and H.-R. Hou, ‘Robot odor source localization in indoor environments based on gradient adaptive extremum seeking search’, *Building and Environment*, vol. 229, p. 109983, Feb. 2023, doi: 10.1016/j.buildenv.2023.109983.

[11] Y. Liu and G. Nejat, ‘Robotic Urban Search and Rescue: A Survey from the Control Perspective’, *J Intell Robot Syst*, vol. 72, no. 2, pp. 147–165, Nov. 2013, doi: 10.1007/s10846-013-9822-x.

[12] A. Dhariwal, G. S. Sukhatme, and A. A. G. Requicha, ‘Bacterium-inspired robots for environmental monitoring’, in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004*, New Orleans, LA, USA: IEEE, 2004, pp. 1436-1443 Vol.2. doi: 10.1109/ROBOT.2004.1308026.

[13] I. Rañó, ‘ON TAXIS FOR CONTROL AND ITS QUALITATIVE SOLUTION ON MOBILE ROBOTS’, *IFAC Proceedings Volumes*, vol. 40, no. 15, pp. 1–6, 2007, doi: 10.3182/20070903-3-FR-2921.00003.

[14] E. Webster, L. Ng, K. Prendergast, and S. R. Howard, ‘The effects of temperature, urbanisation, and artificial light on phototaxis in eusocial and non-eusocial bees’, *Journal of Insect Physiology*, vol. 169, p. 104927, Mar. 2026, doi: 10.1016/j.jinsphys.2025.104927.

[15] R. Z. Tan and K.-H. Chiam, ‘A computational model for how cells choose temporal or spatial sensing during chemotaxis’, *PLoS Comput Biol*, vol. 14, no. 3, p. e1005966, Mar. 2018, doi: 10.1371/journal.pcbi.1005966.

[16] Z. P. Li and M. Sixt, ‘Cell migration: How animal cells run and tumble’, *Current Biology*, vol. 35, no. 18, pp. R890–R892, Sep. 2025, doi: 10.1016/j.cub.2025.08.016.

[17] D. Shaikh and I. Rañó, ‘Braitenberg Vehicles as Computational Tools for Research in Neuroscience’, *Front. Bioeng. Biotechnol.*, vol. 8, p. 565963, Sep. 2020, doi: 10.3389/fbioe.2020.565963.

[18] C. Lozano, B. Ten Hagen, H. Löwen, and C. Bechinger, ‘Phototaxis of synthetic microswimmers in optical landscapes’, *Nat Commun*, vol. 7, no. 1, p. 12828, Sep. 2016, doi: 10.1038/ncomms12828.

[19] F. Vaussard, P. Rétornaz, M. Liniger, and F. Mondada, ‘The Autonomous Photovoltaic MarXbot’, in *Intelligent Autonomous Systems 12*, vol. 194, S. Lee, H.

Cho, K.-J. Yoon, and J. Lee, Eds, in Advances in Intelligent Systems and Computing, vol. 194. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 175–183. doi: 10.1007/978-3-642-33932-5_17.

[20] A. R. Cheraghi, F. S. Vila, and K. Graffi, 'Phototactic Movement of Battery-Powered and Self-Charging Robot Swarms', in *2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, Singapore, Singapore: IEEE, Jul. 2020, pp. 73–79. doi: 10.1109/ACIRS49895.2020.9162628.

[21] T. Alam and L. Bobadilla, 'Multi-Robot Coverage and Persistent Monitoring in Sensing-Constrained Environments', *Robotics*, vol. 9, no. 2, p. 47, Jun. 2020, doi: 10.3390/robotics9020047.

[22] S. Suri, E. Vicari, and P. Widmayer, 'Simple Robots with Minimal Sensing: From Local Visibility to Global Geometry', *The International Journal of Robotics Research*, vol. 27, no. 9, pp. 1055–1067, Sep. 2008, doi: 10.1177/0278364908095833.

[23] G. C. Ampuero, G. Hermosilla, G. Varas, and M. T. Clark, 'Deep Reinforcement Learning for Sim-to-Real Robot Navigation with a Minimal Sensor Suite for Beach-Cleaning Applications', *Applied Sciences*, vol. 15, no. 19, p. 10719, Oct. 2025, doi: 10.3390/app151910719.

[24] A. Majumdar, Z. Mei, and V. Pacelli, 'Fundamental limits for sensor-based robot control', *The International Journal of Robotics Research*, vol. 42, no. 12, pp. 1051–1069, Oct. 2023, doi: 10.1177/02783649231190947.

[25] G. A. Naselli, R. B. N. Scharff, M. Thielen, F. Visentin, T. Speck, and B. Mazzolai, 'A Soft Continuum Robotic Arm with a Climbing Plant-Inspired Adaptive Behavior for Minimal Sensing, Actuation, and Control Effort', *Advanced Intelligent Systems*, vol. 6, no. 4, p. 2300537, 2024, doi: 10.1002/aisy.202300537.

[26] L. Michael, 'Partial observability and learnability', *Artificial Intelligence*, vol. 174, no. 11, pp. 639–669, Jul. 2010, doi: 10.1016/j.artint.2010.03.004.

[27] Q. Liu, A. Chung, C. Szepesvari, and C. Jin, 'When Is Partially Observable Reinforcement Learning Not Scary?', *Proceedings of 35th Conference on Learning theory, Proceedings of Machine Learning Research*, vol. 178, pp. 5175–5220, 2022.

[28] L. Meng, R. Gorbet, M. Burke, and D. Kulić, 'Multi-step first: A lightweight deep reinforcement learning strategy for robust continuous control with partial observability', *Neural Networks*, vol. 199, p. 108521, 2026, doi: 10.1016/j.neunet.2025.108521.

[29] P. Arias-Perez, A. Gautam, M. Fernandez-Cortizas, D. Perez-Saura, S. Saripalli, and P. Campoy, 'Exploring Unstructured Environments Using Minimal Sensing on Cooperative Nano-Drones', *IEEE Robot. Autom. Lett.*, vol. 9, no. 12, pp. 11202–11209, Dec. 2024, doi: 10.1109/LRA.2024.3486212.

[30] S. Baek, T.-K. Lee, O. Se-Young, and K. Ju, 'Integrated On-Line Localization, Mapping and Coverage Algorithm of Unknown Environments for Robotic Vacuum Cleaners Based on Minimal Sensing', *Advanced Robotics*, vol. 25, no. 13–14, pp. 1651–1673, Jan. 2011, doi: 10.1163/016918611X584622.

[31] L. Wang, Y. Yu, D. Zhang, and H. Yang, 'Self-organized Rectangular Aggregation of Swarm Robotics with Minimum Sensing Range', in *2023 42nd Chinese Control Conference (CCC)*, Tianjin, China: IEEE, Jul. 2023, pp. 6099–6104. doi: 10.23919/CCC58697.2023.10240675.

[32] J. Wu, Z. Jin, A. Liu, L. Yu, and F. Yang, 'A survey Of learning-Based control of robotic visual servoing systems', *Journal of the Franklin Institute*, vol. 359, no. 1, pp. 556–577, Jan. 2022, doi: 10.1016/j.jfranklin.2021.11.009.

[33] J. Rakhmatillaev, V. Bucinskas, and N. Kabulov, 'An integrative review of control strategies in robotics', *Robot. syst., appl.*, vol. 5, no. 2, pp. 50–74, Dec. 2025, doi: 10.21595/rsa.2025.25014.

[34] P. Dayan and K. C. Berridge, 'Model-based and model-free Pavlovian reward learning: Revaluation, revision, and revelation', *Cogn Affect Behav Neurosci*, vol. 14, no. 2, pp. 473–492, Jun. 2014, doi: 10.3758/s13415-014-0277-8.

[35] N. Drummond and Y. Niv, 'Model-based decision making and model-free learning', *Current Biology*, vol. 30, no. 15, pp. R860–R865, Aug. 2020, doi: 10.1016/j.cub.2020.06.051.

[36] M. Cord and P. Cunningham, Eds, *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*. in Cognitive Technologies. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2008. doi: 10.1007/978-3-540-75171-7.

[37] T. P. Nascimento and M. Saska, ‘Position and attitude control of multi-rotor aerial vehicles: A survey’, *Annual Reviews in Control*, vol. 48, pp. 129–146, 2019, doi: 10.1016/j.arcontrol.2019.08.004.

[38] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, ‘Imitation Learning: A Survey of Learning Methods’, *ACM Comput. Surv.*, vol. 50, no. 2, pp. 1–35, Mar. 2018, doi: 10.1145/3054912.

[39] S. N. Vassilyev, A. Yu. Kelina, Y. I. Kudinov, and F. F. Pashchenko, ‘Intelligent Control Systems’, *Procedia Computer Science*, vol. 103, pp. 623–628, 2017, doi: 10.1016/j.procs.2017.01.088.

[40] M. Xie, W. Yu, H. Jin, W. Li, and X. Chen, ‘CBAM-ST-GCN: An enhanced DRL-based end-to-end visual navigation framework for mobile robot’, *Neural Networks*, vol. 198, p. 108622, 2026, doi: 10.1016/j.neunet.2026.108622.

[41] H. Sun and I. Guyon, ‘Modularity in Deep Learning: A Survey’, in *Intelligent Computing*, 2023, pp. 561–595. doi: 10.1007/978-3-031-37963-5_40.

[42] J. Wu, Z. Jin, A. Liu, L. Yu, and F. Yang, ‘A hybrid deep-Q-network and model predictive control for point stabilization of visual servoing systems’, *Control Engineering Practice*, vol. 128, p. 105314, Nov. 2022, doi: 10.1016/j.conengprac.2022.105314.

[43] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, 2nd edn. Cambridge, Massachusetts: The MIT Press, 2018.

[44] L. Brunke *et al.*, ‘Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning’, Dec. 06, 2021, *arXiv*: arXiv:2108.06266. doi: 10.48550/arXiv.2108.06266.

[45] B. Singh, R. Kumar, and V. P. Singh, ‘Reinforcement learning in robotic applications: a comprehensive survey’, *Artif Intell Rev*, vol. 55, no. 2, pp. 945–990, Feb. 2022, doi: 10.1007/s10462-021-09997-9.

[46] J. Zhu, F. Wu, and J. Zhao, ‘An Overview of the Action Space for Deep Reinforcement Learning’, in *2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*, Sanya China: ACM, Dec. 2021, pp. 1–10. doi: 10.1145/3508546.3508598.

[47] B. Belousov, H. Abdulsamad, P. Klink, S. Parisi, and J. Peters, Eds, *Reinforcement Learning Algorithms: Analysis and Applications*, vol. 883. in Studies in

Computational Intelligence, vol. 883. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-41188-6.

[48] M. Li, T. Huang, and W. Zhu, 'Adaptive exploration policy for exploration–exploitation tradeoff in continuous action control optimization', *Int. J. Mach. Learn. & Cyber.*, vol. 12, no. 12, pp. 3491–3501, Dec. 2021, doi: 10.1007/s13042-021-01387-5.

[49] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, 'Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms', *Machine Learning*, vol. 38, no. 3, pp. 287–308, Mar. 2000, doi: 10.1023/A:1007678930559.

[50] S. Zou, T. Xu, and Y. Liang, 'Finite-Sample Analysis for SARSA with Linear Function Approximation', *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 8668–8678, 2019.

[51] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, 'Deterministic Policy Gradient Algorithms', in *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014, pp. 387–395.

[52] E. F. Morales, R. Murrieta-Cid, I. Becerra, and M. A. Esquivel-Basaldua, 'A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning', *Intel Serv Robotics*, vol. 14, no. 5, pp. 773–805, Nov. 2021, doi: 10.1007/s11370-021-00398-z.

[53] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, 'Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes', *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 8, no. 1, pp. 153–188, May 2025, doi: 10.1146/annurev-control-030323-022510.

[54] A. Plaat, *Deep Reinforcement Learning, a textbook*. Singapore: Springer Nature Singapore, 2022. doi: 10.1007/978-981-19-0638-1.

[55] L. C. Garaffa, M. Basso, A. A. Konzen, and E. P. De Freitas, 'Reinforcement Learning for Mobile Robotics Exploration: A Survey', *IEEE Trans. Neural Netw. Learning Syst.*, vol. 34, no. 8, pp. 3796–3810, Aug. 2023, doi: 10.1109/TNNLS.2021.3124466.

[56] D. Zheng, J. Yan, T. Xue, and Y. Liu, 'State-Dependent Maximum Entropy Reinforcement Learning for Robot Long-Horizon Task Learning', *J Intell Robot Syst*, vol. 110, no. 1, p. 19, Mar. 2024, doi: 10.1007/s10846-024-02049-8.

[57] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, ‘Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning’, *IEEE Access*, vol. 9, pp. 153171–153187, 2021, doi: 10.1109/ACCESS.2021.3126658.

[58] R. Tiwari, S. Khapre, and A. Singh, ‘Reinforcement learning in robotic systems : A review on sim-to-real transfer’, *Robotics and Autonomous Systems*, vol. 198, p. 105327, Apr. 2026, doi: 10.1016/j.robot.2025.105327.

[59] G. Dulac-Arnold, D. Mankowitz, and T. Hester, ‘Challenges of Real-World Reinforcement Learning’, Apr. 29, 2019, *arXiv*: arXiv:1904.12901. doi: 10.48550/arXiv.1904.12901.

[60] T. Toner, M. Saez, D. M. Tilbury, and K. Barton, ‘Opportunities and challenges in applying reinforcement learning to robotic manipulation: An industrial case study’, *Manufacturing Letters*, vol. 35, pp. 1019–1030, Aug. 2023, doi: 10.1016/j.mfglet.2023.08.055.

[61] V. O. Mihalca, D. M. Anton, R. C. Ţarcă, and Ş. Yıldırım, ‘Simulating Simple Tasks for a Kinematic Model of Differential-drive Mobile Robots’, in *2023 17th International Conference on Engineering of Modern Electric Systems (EMES)*, Oradea, Romania: IEEE, Jun. 2023, pp. 1–4. doi: 10.1109/EMES58375.2023.10171646.

[62] N. H. Thai, T. T. K. Ly, H. Thien, and L. Q. Dzung, ‘Trajectory Tracking Control for Differential-Drive Mobile Robot by a Variable Parameter PID Controller’, *IJMERR*, pp. 614–621, 2022, doi: 10.18178/ijmerr.11.8.614-621.

[63] J. He *et al.*, ‘Curiosity-Driven Testing for Sequential Decision-Making Process’, in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, Lisbon Portugal: ACM, Apr. 2024, pp. 1–14. doi: 10.1145/3597503.3639149.

[64] Q. Hu, *Markov decision processes with their applications*. in Advances in mechanics and mathematics, no. v. 14. New York: Springer, 2008.

[65] P. K. Mohanty, A. K. Sah, V. Kumar, and S. Kundu, ‘Application of Deep Q-Learning for Wheel Mobile Robot Navigation’, in *2017 3rd International Conference on Computational Intelligence and Networks (CINE)*, Odisha: IEEE, Oct. 2017, pp. 88–93. doi: 10.1109/CINE.2017.11.

[66] F. Rashidieranjbar, A. Farhadi, A. Zamanifar, and M. Sorouri, ‘A systematic literature review: Generative artificial intelligence applications for ground mobile robot navigation’, *Computers and Electrical Engineering*, vol. 130, p. 110906, Feb. 2026, doi: 10.1016/j.compeleceng.2025.110906.