LANDMARK-BASED AGGREGATION METHOD FOR ROBOT SWARMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ARASH SADEGHI AMJADI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

JUNE 2021

Approval of the thesis:

LANDMARK-BASED AGGREGATION METHOD FOR ROBOT SWARMS

submitted by **ARASH SADEGHI AMJADI** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Mehmet Ali Sahir Arıkan
Head of Department, **Mechanical Engineering** _____

Assist. Prof. Dr. Ali Emre Turgut
Supervisor, **Mechanical Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Ahmet Buğra Koku
Mechanical Engineering, METU _____

Assist. Prof. Dr. Ali Emre Turgut
Mechanical Engineering, METU _____

Assoc. Prof. Dr. Mehmet Bülent Özer
Mechanical Engineering, METU _____

Assoc. Prof. Dr. Ender Yıldırım
Mechanical Engineering, METU _____

Assist. Prof. Dr. Kutluk Bilge Arıkan
Mechanical Engineering, TED University _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Arash Sadeghi Amjadi

Signature :

ABSTRACT

LANDMARK-BASED AGGREGATION METHOD FOR ROBOT SWARMS

Amjadi, Arash Sadeghi
M.S., Department of Mechanical Engineering
Supervisor: Assist. Prof. Dr. Ali Emre Turgut

June 2021, 69 pages

Aggregation, a widely observed behavior in social insects, is the gathering of individuals at any location or on a cue. The former being called self-organized aggregation, and the latter being called cue-based aggregation. One of the fascinating examples of cue-based aggregation is the thermotactic behavior of young honeybees. Young honeybees aggregate on optimal temperature zones in the hive using a simple set of behaviors. The state-of-the-art cue-based aggregation method BEECLUST was derived based on these behaviors. The BEECLUST method is a very simple yet very capable method with favorable characteristics such as robustness to noise and simplicity. However, the BEECLUST method does not perform well in low robot population densities. In this thesis, inspired by the navigation techniques used by ants and bees, a self-adaptive landmark-based aggregation method is proposed. In this method, robots use landmarks in the environment to locate the cue once they "learn" the relative position of the cue with respect to the landmark. Robots were utilized with odometry sensors to make the calculation of traveled distances possible. With the introduction of an error threshold parameter, the method also becomes adaptive to changes in the environment. In order to make robots robust to sensor noises and free of fine-tuning, reinforcement learning algorithm was employed to aid robots in coping better with

uncertainties. In order to solve exploration-exploitation dilemma in reinforcement learning, a new cyclical update schedule was proposed.

Through systematic experiments in kinematic and realistic simulators and real swarm robots with different parameters, it was observed that using the information of the landmarks makes the proposed method outperform other state-of-the-art cue-based aggregation methods such as BEECLUST and ODOCLUST in all the settings. It was also shown that utilizing reinforcement learning in the proposed aggregation method had a 20% performance increase in non-stationary environments. Additionally, reinforcement learning made the proposed method more robust to odometry noise reaching up to 30% performance increase.

Keywords: Bio-inspired, Swarm Robotics, Cue-based Aggregation, Landmark-based Navigation, Reinforcement Learning, Self-adaptive

ÖZ

ROBOT SÜRÜLERİ IÇİN KONUM NOKTASI TABANLI TOPLANMA YÖNTEMİ

Amjadi, Arash Sadeghi
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Ali Emre Turgut

Haziran 2021 , 69 sayfa

Sosyal böceklerde yaygın olarak gözlemlenen bir davranış olan kümelenme, bireylerin herhangi bir yerde veya bir işarette toplanmasıdır. İlki kendi kendini organize eden toplama olarak adlandırılır ve ikincisi işaret tabanlı toplama olarak adlandırılır. Işarete dayalı kümelenmenin örneklerinden biri, genç bal arılarının termotaktik davranışıdır. Genç bal arıları basit bir dizi davranış kullanarak kovandaki en uygun sıcaklık bölgelerinde toplanır. Son teknoloji işaret tabanlı toplama yöntemi BEECLUST, bu davranışlara dayalı olarak türetilmiştir. BEECLUST yöntemi çok basittir ve gürültüye karşı dayanıklılık ve uygulama kolaylığı gibi olumlu özelliklere sahip bir yöntemdir. Ancak, BEECLUST yöntemi, düşük robot yoğunluklarında iyi performans göstermez. Bu yazıda, navigasyon tekniklerinden esinlenerek Karıncalar ve arılar tarafından kullanılan, kendi kendini ayarlayan bir dönüm noktası tabanlı toplama yöntemi önerilmiştir. Bu yöntemde robotlar konum işaretine göre yer işaretin göreceli konumunu "öğrendikten" sonra, yer işaretin yerini belirlemek için ortamdaki konum işaretlerini kulanırlar. Katedilen mesafelerin hesaplanmasını mümkün kılmak için robotlarda odometri sensörleri kullanıldı. Bir hata eşiği parametresinin eklenme-

siyle, yöntem aynı zamanda çevre değişikliklerine ayarlanabilir hale getirilir. Robotları odometri sensör parazitine karşı sağlam ve ince ayardan arındırmak için, robotların belirsizliklerle daha iyi başa çıkmasına yardımcı olmak için takviyeli öğrenme algoritması kullanıldı. Takviyeli öğrenmede keşfetme-sömürme ikilemini çözmek için yeni bir döngüsel güncelleme programı önerilmiştir.

Kinematik ve gerçekçi simülatör ve gerçek sürü robotlarında farklı parametrelere sistematik deneyler yoluyla, ortam bilgilerinin kullanılmasının önerilen yöntemin tüm diğer son teknoloji işaret tabanlı toplama yöntemleri olan BEECLUST ve ODOCLUST'tan daha iyi performans gösterdiği gözlemlendi. Ayrıca önerilen toplama yönteminde takviyeli öğrenmenin kullanılmasının durağan olmayan ortamlarda 20%'lik bir performans artışı sağladığı da gösterilmiştir. Ek olarak, takviyeli öğrenme, önerilen yöntemi 30%'a varan performans artışına ulaşan odometri gürültüsüne karşı daha sağlam hale getirdiği gözlenmiştir.

Anahtar Kelimeler: Biyo-ilham, Sürü Robotik, İşaret tabanlı Toplama, Landmark tabanlı Navigasyon, Takviye Öğrenme, Kendinden ayarlanabilir

To Farshad Arvin, who changed my life, and to my parents, who supported me in my new life.

ACKNOWLEDGMENTS

Foremost, I would like to thank my supervisor, Dr. Ali Emre Turgut, for his generous supports throughout my academic career. The experience and knowledge that I gained through my researches with him were priceless. I would also like to express my great gratitude to Dr. Farshad Arvin for giving me the chance and opportunity to conduct my researches. I literally owe all of my academic successes to his kind support and guidance. Moreover, I would like to thank Dr. Erol Şahin for giving me the chance to work in the KOVAN lab and paving my way to implement my ideas and conduct my researches.

Likewise, I would like to thank Cem Bilaloğlu for aiding me in the hardware implementation of robots and assisting me with real-world experiments. Without his efforts and aids, bringing ideas of this work to real robots would not be possible. Furthermore, I gained valuable experiences in authoring academic papers through my collaboration with Mohsen Raoufi and Seongin Na; thanks for your contributions. Also, thanks to Dr. Zhengtao Ding for introducing me to Dr. Farshad Arvin.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

LBA	Landmark-based Aggregation
RL	Reinforcement Learning
LBA-RL	Landmark-based Aggregation with Reinforcement Learning
VDBE	Value-Difference Based Exploration

CHAPTER 1

INTRODUCTION

Aggregation is an essential behavior for social animals. It is the gathering of individuals into a single aggregate. This behavior is beneficial for the survival of insects since they gain extra abilities such as protection against predators [2] or resistance to adverse environmental conditions [3]. Different kinds of animals from amoeba [4] to cockroaches [5] perform aggregation. In nature, aggregation is observed in two different ways; self-organized and cue-based. In self-organized aggregation, an aggregate is formed independently from the environment [6]. However, in cue-based aggregation, an aggregate is formed based on an external cue, such as temperature as seen in honey bees [7].

Inspired by the thermotactic behavior of young honey bees [8], a cue-based aggregation method, BEECLUST, was proposed [9]. By systematically studying the behavior of actual honey bees in the laboratory setting, it has been observed that randomly moving honey bees stop once they meet another honey bee. They stay longer if the temperature of the region (intensity of cue) is higher. By transferring the aggregation behavior of honey bees to swarm robots, robots were able to imitate the cue-based aggregation of honey bees [10]. The BEECLUST method due to its simplicity showed its effectiveness in real-world applications such as in contamination source detection in extreme environments [11]. Contamination sources like chemical waste were considered as the cue, and the robots aggregated on the cue and cleaned it based on BEECLUST.

Environment plays an important role in the lives of animals. It provides food, protection, and most notably, it helps with navigation. As an example, ants release pheromone trails in their environment to help them to move between their nest and

food source [12]. Besides, honey bees are also known to use the environment for navigation [13] by employing both visual [14], and olfactory cues [15, 16] to aid them in their foraging task. By learning the distance and angle with respect to certain landmarks, both honey bees and ants can find their way back home from a foraging site [17]. In swarm robotics, this kind of navigation was implemented mainly in foraging tasks such as an adaptive foraging method which was proposed based on landmark-based navigation [18] using RFID tags as landmarks.

In this thesis, inspired from the navigation techniques used by the ants and bees [17], a self-adaptive landmark-based aggregation method is proposed based on the BEECLUST method. The main motivation of the proposed method is to increase the performance of the BEECLUST method in low robot densities [19]. The performance of the method is tested and compared with the BEECLUST and ODOCLUST [1] method in static and dynamic environments with different robot densities, cue sizes, and noise. Initially, a path integration approach is used for achieving landmark-based navigation in cue-based aggregation. Then, the proposed Landmark-based Aggregation (LBA) method is further improved by using Reinforcement Learning (RL) [20] for making the proposed method more robust to sensory noises and reducing sensitivity to parameter tuning. The integration of the RL algorithm to the LBA method is called the Landmark-based Aggregation with Reinforcement Learning (LBA-RL)

This thesis is organized as follows: In the next chapter, the state of the literature on cue-based aggregation and related works to landmark-based aggregation are presented. The contribution of this work is also stated. In Chapter 3, the LBA method and its modified version, the LBA-RL method, are presented and formulated. In Chapter 4, experimental setups are introduced. In Chapter 5, the results of the conducted experiments are presented and discussed. Finally, Chapter 6 concludes the studies and works done in this thesis.

CHAPTER 2

LITERATURE SURVEY

2.1 Cue-based Aggregation

The BEECLUST method has been studied for over a decade. In [21], the BEECLUST method was investigated in both static and dynamic environments. In the static environment, there was only one light source, while, in the dynamic environment, two light sources were located in the arena, and the intensity of the light sources was varied during the experiments. It was shown that the BEECLUST method was robust in dynamic environments. In [22], parameters of the BEECLUST method were studied systematically, and velocity and waiting time parameters were modified and tested in a dynamic environment. The results showed that both the aggregation time and aggregation performance improved. In [23], a fuzzy-logic-based aggregation method was proposed. Through experiments in single- and multiple-cue arenas with static and dynamic settings, the proposed method performed better than the BEECLUST method. The BEECLUST method was modified in [24] such that robots were able to calibrate the waiting time based on the intensity of the cue, which increased the aggregation performance. In a follow-up study [25], two additions were made to the BEECLUST method so that robots were able to measure local robot density and light intensity and share this information with their neighbors. Through experiments, it was shown that robots were able to adapt to dynamic lighting conditions, and, as a result, the performance of aggregation was improved in the low-density robot population. BEECLUST method was modified in [1] such that the "wait" state was changed with the "seek" state. When a robot encountered another robot, instead of waiting, it sought the location of its last encounter with another robot for a predefined amount of time. Experiments revealed that the proposed method performed better than the

BEECLUST method in low robot population density setups. While all the aforementioned works considered behavioral homogeneity, in [26] a swarm with two different behavioral groups, one with the tendency to aggregate in the area with high illuminance and the other with the tendency to aggregate in the area with a low illuminance, were created using the BEECLUST method. Through experiments, it was shown that the performance of aggregation depends on the density of robot population. Similarly, in [27] four different behavioral groups (goal finder, wall follower, random walker, and immobile agent) were created, and the BEECLUST method was applied. Evolutionary experiments showed that a certain combination of behavioral groups (in the descending order of ratios: wall follower, immobile agent, random walker, and goal finder) gives the best aggregation performance.

2.2 Landmark-based Navigation in Swarm Robotics

Different landmark-based navigation techniques have been adopted in swarm robotics. Using pheromones is one such technique applied by different means in different studies [23, 28, 29, 30]. In [31], an artificial pheromone system [32, 33] with an LCD and a USB camera was used. Through systematic experiments, the effects of different parameters such as evaporation and diffusion rates were studied. It was shown that pheromone-based landmark navigation improved aggregation performance. In [18] an adaptive foraging method using landmark navigation was proposed. The method was based on the path integration capabilities of bees. RFID tags and other agents were used as landmarks, and it was shown that the proposed method improved the foraging performance. In [34] different ways of realizing landmarks, including the QR-codes in a real environment was evaluated in different settings, and it was concluded that the feasibility of using a particular landmark realization depends on its complexity and the computational capabilities of the robot.

2.3 Reinforcement Learning

2.3.1 Reinforcement Learning in Swarm Robotics

In nature, a way for animals and humans to learn optimal behaviors in their habitats is by receiving positive and negative signals. Their brain interprets signals like food and pleasure as rewards and signals such as hunger and pain as punishment. Thus, animals choose a behavior that maximizes their received rewards [20]. Inspired by this behavior in animals, reinforcement learning (RL) paradigm has been developed. In RL, an agent is expected to learn how to perform a task at a desired level by receiving appropriate reward signals. Nowadays, reinforcement learning is used in a wide range of research areas such as computer systems [35], energy [36], finance [37], healthcare [38], robotics [39, 40], and transportation [41].

Multi-robot systems and swarm robotics also benefit from reinforcement learning for solving various types of problems. For instance, a guided approach was proposed in [42] for controlling swarm robots. Learning policies and control rules for a swarm of simple cooperative robots is a hard task due to distributed partial observability of the states. [42] proposed a guided method where a critic has access to all the global states of the agents during the training. In [43] for tackling obstacles in an unknown environment, a deep reinforcement learning-based collision avoidance was proposed. RL algorithms are a proper choice for problems that involve uncertainties and dynamic environments [44]. As an example, in [45] a multi-agent reinforcement learning approach was proposed for dealing with uncertainties in coalition task.

2.3.2 Exploration-Exploitation Dilemma

RL algorithms have an exploration-exploitation dilemma. Once an agent reaches a state, it should decide and execute an action. If an agent always takes random actions, it will not use its learned knowledge and this way, agent improves its knowledge about each action. On the other hand, if an agent always exploits its knowledge to choose the best action, it will not explore most of the environment's possibilities, and chances of received rewards restricted by local minimum are high. Therefore,

a balance is required between exploration and exploitation. A simple and effective solution for solving the exploration-exploitation problem is the ϵ-greedy method. In this method, ε is the probability of exploring and agent exploits its knowledge with a probability of $1-\varepsilon$. The parameter ε is in the range of $\varepsilon \in [0, 1]$, higher ε values leads to full exploration (completely random actions) and in lower ε values agent exploits its learned knowledge.

Typically, ε takes a constant value or decays over time. Taking ε as a fixed value will restrict the performance of the algorithm. Reducing ε over time has a positive effect for static environments and a negative effect for dynamic environments. In a static environment, reducing ε over time will cause convergence of policy. On the other hand, for a dynamic environment, if a change happens, robots will not be able to perform enough exploration to learn the new environment, and they will stick to their previous knowledge, which will heavily reduce the performance of the method. Therefore, there is a need for a different schedule for ε.

One approach for adaptive ε is to make it dependent on the certainty of an agent about its environment. In [46], a concept of Value-Difference Based Exploration (VDBE) was proposed, where agents' certainty about environment is measured by the temporal difference between two value functions. The main goal of the approach is that the more uncertain the agent is about the environment (higher value function differences), the more it should explore (higher ε value). The more certain the agent is about the environment (lower value function differences), the more it should exploit (lower ε value). The parameter σ with the range of $\sigma \in (0, \infty)$, which is called inverse sensitivity, plays an important role in the VDBE schedule. It adjusts how sensitive an agent should be to changes in the environment. For the VDBE schedule, σ is the only parameter that needs to be tuned. Higher σ values will act like constant ε, which means changes in the environment will not affect ε considerably. On the other hand, lower σ values will change ε drastically if even a small change happens in the environment. Thus, depending on the conditions of the environment, σ must be tuned to obtain the best performance. They compared VDBE with ϵ-greedy and Softmax policies on multi-armed bandit tasks. The average reward per time step for VDBE was 1.42, Softmax 1.38, and ϵ-greedy 1.35. In conclusion, VDBE outperformed other studied ε schedules.

Another work for adaptive ε parameter was done in [47]. They studied their method for both stationary and non-stationary environments. For non-stationary environments, they used an algorithm to detect the change point of the environment. They defined the variable Δ as the difference between the highest average rewards and the highest previous average rewards. While Δ was positive, ε was updated by a sigmoid function. Their method showed better performance than ϵ-greedy method in both stationary and non-stationary environments. They also compared their approach to VDBE. Results revealed that VDBE with $\sigma = 0.33$ could get a higher average reward faster and could detect the optimal actions quicker but settled in a lower value compared to their proposed adaptive ε method. However, VDBE with $\sigma = 0.04$ could select the optimal action better than all other experimented schedules.

In another work, the Bayesian perspective of ε was used to measure the uncertainty in the Q-value function [48]. A closed-form Bayesian model was proposed, based on the Bayesian model combination (ε-BMC), which enabled adapting ε according to past experiences from the environment. Their results demonstrated that ε-BMC could outperform fixed schedules for ε and state-of-the-art ε adaptation methods such as VDBE.

So far, studies that aim to solve the dilemma of exploration-exploitation balance and develop an adaptive ε in model-free reinforcement learning are discussed. In deep learning, a similar problem is encountered for the learning rate parameter. In Stochastic Gradient Decent (SGD), learning rate plays an essential role in neural networks learning abilities. In a recent work, the learning rate was let to oscillate between two predefined values [49]. They showed that the cyclical learning rate improves the performance of their algorithm. Besides, this scheduling method is relatively simple to implement compared to first and second-order adaptive learning rates. Their work on learning rates for neural networks inspired us to propose a cyclical schedule for ε. Also, the simplicity of the cyclical update schedule and the fact that it uses significantly low memory and less computational resources make it more appropriate for swarm robots.

2.4 Contribution

The contribution of this thesis is indicated as below:

1. To the best of our knowledge, this is the first implementation of landmark-based navigation in a cue-based aggregation setting. Landmarks are employed to increase the poor performance of the BEECLUST aggregation method in low robot density settings.
2. Advantages and restrictions of RL algorithms for increasing the aggregation performance of swarm robots under uncertainties and sensor noises are studied.
3. A cyclical parameter update was proposed for solving the exploration-exploitation dilemma [50] is proposed, which is more robust to fine-tuning and can adapt better to environmental changes, compared to other studied methods.

In conclusion, an aggregation method that can form a proper aggregate in low-density swarm setups with high adaptability to environmental change and low vulnerability to fine-tuning is proposed in this thesis.

CHAPTER 3

METHODOLOGY

3.1 BEECLUST Aggregation

The BEECLUST method is based on three behavioral states which were observed in honeybees [9]: 1) go forward, 2) avoid obstacles, 3) wait for a certain amount of time if another robot is encountered. The BEECLUST method is represented as pseudo-code in Algorithm 1. The waiting time is calculated according to:

$$w_s = w_{max} \frac{{I_c}^2}{{I_c}^2 + 5000} \quad , \tag{3.1}$$

where w_{max} is the maximum waiting time, which is set to $120\,\mathrm{s}$ and I_c is the cue intensity measured by the ground sensors of robots at the location of the collision and its range is $I_c \in [0, 255]$. Once the waiting time is over, robots turn $\theta°$ where θ is a random variable in the range of $[90°, 270°]$ with a uniform distribution.

Algorithm 1: BEECLUST

```
1  Procedure BEECLUST
2      go forward
3      if obstacle detected then
4          turn θ°
5      else if robot detected then
6          measure cue intensity, I_c
7          if I_c > 0 then
8              calculate w_s according to Eq. (3.1)
9              wait for w_s
10         turn θ°
```

3.2 ODOCLUST Aggregation

The main idea behind the ODOCLUST method was to increase the aggregation performance using odometry. In this method, robots explore the arena, and save the position of the last two collision points by using odometry. They go in between these two collision points for a specific amount of time called the timeout. In this thesis, the ODOCLUST method was implemented to compare our proposed method with a state-of-the-art aggregation method. In the original version of the ODOCLUST method [1], the timeout was decided randomly. However, since our thesis is focused on cue-based aggregation, the ODOCLUST method was modified and implemented such that the timeout duration was decided based on the cue intensity. Figure 3.1 shows the state machine of the modified ODOCLUST method.

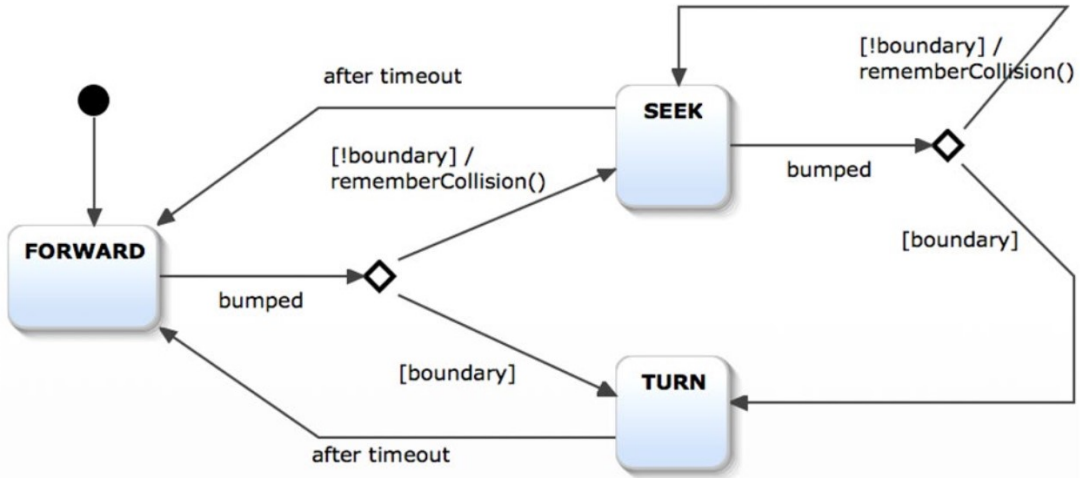


Figure 3.1: State machine of the ODOCLUST method. adopted from [1].

3.3 Landmark-based Aggregation

The Landmark-based Aggregation (LBA) method, shown in the Algorithm 2 is developed based on the BEECLUST method. It is assumed that there are landmarks and a cue present in the environment. Landmarks do not contain any *a priori* information about the location of the cue. They are just distinct identifiers detectable by the robots.

Algorithm 2: Landmark-Based Aggregation

```
1  Procedure LBA
2      go forward
3      if obstacle detected then
4          turn θ°
5      else if landmark k detected then
6          if S⃗^k_total exists then
7              go towards the cue using S⃗^k_total
8              if obstacle detected then
9                  increase error variable one step, e^k++
10                 if e^k ≥ τ_e then
11                     reset S⃗^k_total and e^k
12                 turn θ°
13             else if robot detected then
14                 measure cue intensity, I_c
15                 if I_c > 0 then
16                     calculate w_s according to Eq. (3.1)
17                     wait for w_s
18                 turn θ°
19         else
20             start calculating displacement vectors, S⃗^k_i
21     else if robot detected then
22         measure cue intensity, I_c
23         if I_c > 0 then
24             if S⃗^k_total does not exist then
25                 calculate and store S⃗^k_total
26             calculate w_s according to Eq. (3.1)
27             wait for w_s
28         turn θ°
```

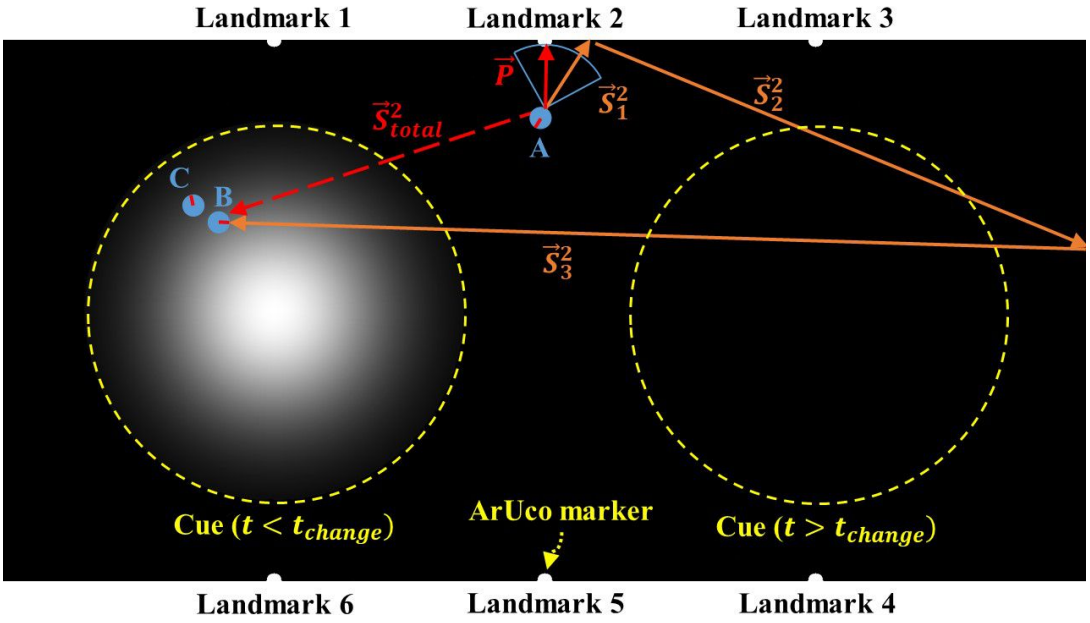


Figure 3.2: Demonstration of how $\vec{S}^2_{total}$ (dashed red vector) of a robot is calculated by integrating its displacement vectors $\vec{S}^2_i$ (orange vectors) in the LBA method. A: location of robot 1 at $t = t_1$, B: location of robot 1 at $t = t_2$, C: location of robot 2 at $t = t_2$. Robot 1 sums the displacement vectors that it traversed from time $t = t_1$ until it encountered another robot inside the cue in time $t = t_2$. The blue arc represents the camera filed of view of robot 1. Dashed yellow circle on the left demonstrates the cue before t_{change}. After environment changes, new location of the cue is the yellow circle on the right.

Initially, a focal robot, as in the BEECLUST method, starts to explore the environment randomly. When it detects one of the landmarks (ID, relative position and orientation of the landmark are extracted) for the first time, it starts to integrate all displacement vectors and continues to explore the environment randomly until it encounters another robot. Then, it checks the intensity at the location of the encounter. If there is a cue, then it stores the total displacement vector, which is determined by adding the individual displacement vectors, as shown in Figure 3.2. The resulting vector conveys information about the relative position of the cue with respect to the detected landmark. So, when the robot detects the same landmark for the next time, it uses the this vector to go directly to the cue without the need for further exploration, unlike the BEECLUST method.

The magnitude and angle of the displacement vectors are calculated using odometry. Odometry is implemented by counting the pulses from the optical encoder on each wheel. By doing so, it is possible for a robot to calculate the traveled distance as well as the rotated angle. The distance that a robot traveled from a given point can be derived by using the incremental travel distance for the left and right wheels (ΔU_L and ΔU_R) using the total wheel revolution as in [51]:

$$\begin{aligned} \Delta U_L &= c_m n_l, \\ \Delta U_R &= c_m n_r, \end{aligned} \tag{3.2}$$

where n_r and n_l are the pulse counts of the right and left wheels of a robot, respectively. c_m is the conversion factor from encoder pulses to linear wheel displacement and it is calculated as:

$$c_m = \frac{\pi D_n}{\alpha_{odometry} C_e}. \tag{3.3}$$

where D_n, C_e, and $\alpha_{odometry}$ are the nominal wheel diameter, the encoder resolution, and the speed reduction ratio, respectively. With ΔU_L and ΔU_R at hand, the linear displacement $s_{odometry}$, and angular displacement $\mu_{odometry}$ can be calculated as:

$$s_{odometry} = \frac{\Delta U_L + \Delta U_R}{2}, \quad \mu_{odometry} = \frac{\Delta U_L - \Delta U_R}{b}, \tag{3.4}$$

where b is the distance between two wheels of a robot.

It is assumed that robots can detect the relative position and orientation of the landmarks. So, when a robot detects a landmark, it calculates its position and orientation with respect to the coordinate system of the landmark as a reference and thereafter keeps track of its orientation using odometry.

The i-th displacement vector, $\vec{S_i^k}$, is represented as:

$$\vec{S_i^k} = l \angle \psi, \tag{3.5}$$

where k denotes the ID number of the landmark that the displacement vectors correspond to. l and ψ are the magnitude and angle of displacement vector in the landmark coordinate system, respectively. When a robot reaches the cue after detecting the k-th landmark, the total displacement vector $\vec{S_{total}^k}$ is calculated by integrating all the displacement vectors that were accumulated as:

$$\vec{S_{total}^k} = \sum_{i=0}^{z} \vec{S_i^k}, \tag{3.6}$$

where z is the number of displacement vectors that a robot has traveled to reach the cue after detecting the landmark. Each displacement vector is calculated between two consecutive turns of a robot. Figure 3.2 shows how $\vec{S}^k_{total}$ is calculated using three consecutive displacement vectors. The robot stores $\vec{S}^k_{total}$ and uses it to go directly to the cue for the next time when it detects the k^{th} landmark. It should be noted that the vector $\vec{P}$, which is the initial relative position of the robot with respect to the landmark, is also used to find the total displacement vector. This way, the base point of the resulting vector $\vec{S}^k_{total}$ is the landmark, rather than the robot, which makes the vector independent of the initial position of the robot.

Each robot saves a new displacement vector when it detects a new landmark and assigns this vector to the ID of the detected landmark. Hence, a robot is able to store as many $\vec{S}^k_{total}$ as the number of landmarks, m, in the arena: $\vec{S}^1_{total}$,$\vec{S}^2_{total}$, ...,$\vec{S}^m_{total}$.

To add the effect of noise due to detection and odometry, noise is added to the actual direction, ψ^k_i, and the noisy direction, $\tilde{\psi}^k_i$, is calculated as:

$$\tilde{\psi}^k_i = \psi^k_i + \sigma_n \eta^k_i. \tag{3.7}$$

In this equation, σ_n is the strength of the noise, and η is a uniformly distributed dimensionless random variable in the range $[-1, 1]$.

In order to make the LBA method adaptive, an error variable, e^k, and an error threshold, τ_e, are defined. Each e^k is associated with the corresponding $\vec{S}^k_{total}$ for each robot. Every time a robot moves based on $\vec{S}^k_{total}$ but does not reach the cue, the error variable e^k is incremented by one. τ_e denotes the maximum acceptable error for each $\vec{S}^k_{total}$. When the variable e^k becomes greater than the threshold τ_e ($e^k > \tau_e$), the robot does not use $\vec{S}^k_{total}$ anymore and the vector resets. When k^{th} landmark is encountered next time, $\vec{S}^k_{total}$ is recalculated.

An increment in error variable e^k may occur due to three reasons: 1) cue change, which happens when the location of the cue is changed in a dynamic environment, and the robot goes to the previous location of the cue; 2) noise in the measurements of environmental variables, like the relative orientation and position of landmarks; 3) inaccuracies in the odometry calculation, which can be the result of wheel slippage, or other external factors. When a robot travels relatively long distances, errors in the

odometry calculations accumulate, and the deviation of the calculated vector and the real displacement vector that a robot has traveled becomes significant. In all these cases, a robot may not reach the cue when using a particular displacement vector, hence resets the corresponding vector.

3.4 Landmark-based Aggregation with Reinforcement Learning

3.4.1 Q-learning

For finite MDP problems where reward function and transition function are unknown (model-free RL), Q-learning algorithm [52] can be used to derive optimal behavior for an agent. Q-learning is an off-policy reinforcement learning method [44], which uses Q-function for selecting optimal actions. For each state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$ at time t, Q-function $Q(s_t, a_t)$ outputs a scalar which determines how good is to take action a_t in state s_t, i.e., $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. For discrete state and action spaces, Q-function can be modeled as a table with states in row and action in columns, called the Q-table. Each agent updates its Q-table with a recursive update rule stated as:

$$Q^{new}(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha[r_t + \gamma \max_a\{Q(s_{t+1}, a)\}] \quad . \tag{3.8}$$

In this equation, the parameter α, which varies over the range of $\alpha \in (0, 1]$, is the learning rate that controls the learning speed of the agent. High learning rates result in quicker learning, but oscillations are more probable. A low learning rate guarantees convergence, but convergence speed will be slow [44].

Once an agent is in a state s_t, it uses policy function $\pi_t(s_t)$ to choose an action. A policy function maps the state space to an action space $\pi : \mathcal{S} \rightarrow \mathcal{A}$. In this thesis, ϵ-greedy policy [52] is used as the policy function. In this policy, the chance of taking a random action (exploration) is determined by the parameter $0 \leq \varepsilon \leq 1$ and the probability of taking a greedy action (exploitation) is $1 - \varepsilon$. In the greedy action, for a state s_t, agent chooses the action which has the highest value in Q-table.

$$a_t = \begin{cases} \text{random select } a_r \in \mathcal{A} & \text{with probability } \varepsilon \\ \max_a\{Q(s_t, a)\} & \text{with probability } 1 - \varepsilon \end{cases} \tag{3.9}$$

3.4.2 LBA-RL Method

The proposed method, Landmark-based Aggregation with Reinforcement Learning (LBA-RL), is based on the LBA method, Algorithm 3. Before detecting a landmark, a robot moves forward till it detects an object, it turns to a random angle if the detected object is an obstacle, and it stops and waits if the detected object is another robot. Waiting time is proportional to the cue intensity at the location of the collision of two robots, calculated based on Eq. (3.1). When a robot detects a landmark, it utilizes the Q-learning method to find the path to the region of the cue that has the maximum intensity. It should be remarked that in the LBA method, the total displacement vectors $\vec{S}^k_{total}$ only led to the cue, not to the part that has the maximum cue intensity. However, in the LBA-RL method, robots choose the action that leads to the highest Q-value among all other possible actions for a specific state. In the current reinforcement learning problem, states of the MDP model correspond to the landmarks $\mathcal{S} = \{landmark_1, landmark_2, .., landmark_m\}$ where m represents the total number of landmarks in the arena. Action space consists of discrete displacement vectors for each landmark $\mathcal{A} = \{\vec{S}_1, \vec{S}_2, ...\vec{S}_n\}$ where n is the total number of actions that a robot can perform. Each displacement vector can be written as: $\vec{S}^k_i = l(\vec{i}\cos(\psi) + \vec{j}\sin(\psi))$ where l and ψ are the length and angle of the displacement vector and $\psi \in \Psi$ (Ψ set of all possible angles) and $l \in L$ (L set of all possible length). Vectors $\vec{i}$ and $\vec{j}$ represent the unit vectors along x- and y-axis, respectively.

When a robot detects a landmark, it selects an action. This selection can be made randomly (exploration) or by exploiting previous experiences (exploitation), which is expected to give the highest Q-value for state s_t according to learned Q-table $Q(s_t, a_t)$ at time t. Whether a robot performs, exploration or exploitation is decided based on the ϵ-greedy policy Eq. (3.9). The cue intensity that a robot senses after following the chosen displacement vector, is taken as the reward, $r_t \in [-1, 255]$, used to update the Q-table according to Eq. (3.8). Action vectors start at the location of the landmark and end at a point in the arena, as shown in Figure 3.3.

A robot might get interrupted while executing an action by an obstacle or another robot. Suppose that a robot is interrupted by another robot. In that case, it stops the execution of the action, turns randomly, and continues its movement on the arena

without getting any reward and without updating its Q-table. But, if it gets interrupted by an obstacle, this means that the chosen action led to a collision with an obstacle; thus, that action needs to be punished. Therefore, a reward of $r_t = -1$ is given to the robot. For all other circumstances, the received reward is equal to the measured cue intensity, which is always in the interval of $[0, 255]$.

A landmark might be detected from different angles each time by the same robot. Each action vector, $\vec{S}_i^k$, starts from the location of the k^{th} landmark and points to a location in the arena. The objective of a robot is to reach that location. In order to reach that location, no matter from which position a robot detects the landmark, a similar technique also used in the LBA method is adopted here. Once a robot detects a landmark, k^{th} landmark, it measures its distance d and angle ϕ with respect to the landmark using the `solvePnP` function of the OpenCV library [53]. Based on the calculated ϕ and d, relative position vector of the robot with respect to the landmark can be defined as: $\vec{P} = d(\vec{i}\cos\phi + \vec{j}\sin\phi)$. By adding this vector to the chosen action, a robot will be able to reach the endpoint of the action vector by following the target vector $\vec{T}$ calculated as:

$$\begin{aligned} \vec{T} = \vec{S}_i^k + \vec{P} &= l(\vec{i}\cos\psi + \vec{j}\sin\psi) + d(\vec{i}\cos\phi + \vec{j}\sin\phi) \\ &= (l\cos\psi + d\cos\phi)\vec{i} + (l\cos\psi + d\cos\phi)\vec{j} \quad , \end{aligned} \tag{3.10}$$

where $\vec{i}$ and $\vec{j}$ are the unit vectors along the x- and y-axis, respectively.

To employ Q-learning in the LBA method, parameters γ, α, and ε of the equations (3.8) and (3.9) must be handled. As stated before, $\gamma \in [0, 1]$ is the discount factor and determines how important future rewards are. In our case, γ is set to zero. The reason for that is when a robot detects a landmark, using the ϵ-greedy policy, it chooses and executes an action. When the execution of the selected action is completed, the robot gets a reward, which is the cue intensity of the point where the action vector leads the robot to. After receiving this reward, the robot does not assume one of the states in $\mathcal{S}$, it rather turns to a random direction and moves straight until it detects another landmark. Therefore, taking action a_t at state s_t at time t, leads to the reward r_t but does not lead to transition to a next state s_{t+1}. In other words, a robot moves on the arena, detects a landmark, chooses an action, executes that action, receives a reward, and then moves on to the arena again until it detects another landmark and so

on. Consequently, agents are naturally short-sighted and only care about immediate rewards. In literature, such agents are named as myopic agents, and for these kinds of agents, $\gamma = 0$.

To deal with the learning rate parameter α, various heuristic, and adaptive schedules are suggested in both reinforcement learning and deep learning literature. In this thesis, since the environment is assumed to be non-stationary and noisy, a small, constant value for α is chosen as suggested in [44]. For noisy and non-stationary environments, low learning rate values will cause slow but guaranteed convergence. Therefore, the learning rate is set to $\alpha = 0.1$. Although a constant choice of α will restrict the convergence speed of the model and prevent it from reaching its best performance, studying adaptive and more advanced schedules for α is beyond the scope of this thesis, and it is considered as future work. At this point, considering the set values for the γ and α parameters, Eq. (3.8) reduces to:

$$Q^{new}(s_t, a_t) \leftarrow 0.9\, Q(s_t, a_t) + 0.1\, r_t \quad . \tag{3.11}$$

The VDBE schedule [46] is implemented for adjusting ε adaptively based on the robot's certainty about its environment. The optimal values suggested in [46] are used for the parameters of the method. However, the inverse sensitivity parameter, σ, still needs fine-tuning, which is one of the disadvantages of VDBE. The other disadvantage of VDBE is that each state requires its own independent ε, which makes ε a function of state, s. Consequently, as the number of states grows, the VDBE method will require more memory to store ε parameters for each state. For environments with a large state space size, the VDBE schedule consumes a considerable amount of memory to keep track of each ε independently. Therefore, we also implemented another method inspired by way of adjusting the learning rate in [49] for adjusting ε. In this method, ε is changed periodically regardless of the state.

$$\varepsilon_t = A\frac{(1 + \cos\frac{2\pi\lambda}{p})}{2} \quad , \tag{3.12}$$

where ε_t is the value of ε at time t and λ is the epoch. Epoch is increased by one each time a robot detects a landmark and executes an action. The parameters $A \in (0, 1]$ and $p \in (0, \infty)$ are the amplitude and period of the waves, respectively. The reason for adding 1 to the cosine term and dividing them by 2 is that normally $-1 \leq \cos(\lambda) \leq 1$, since $0 \leq \varepsilon_t \leq 1$, cosine is modified such that $0 \leq \frac{1+\cos(\lambda)}{2} \leq 1$ is within the

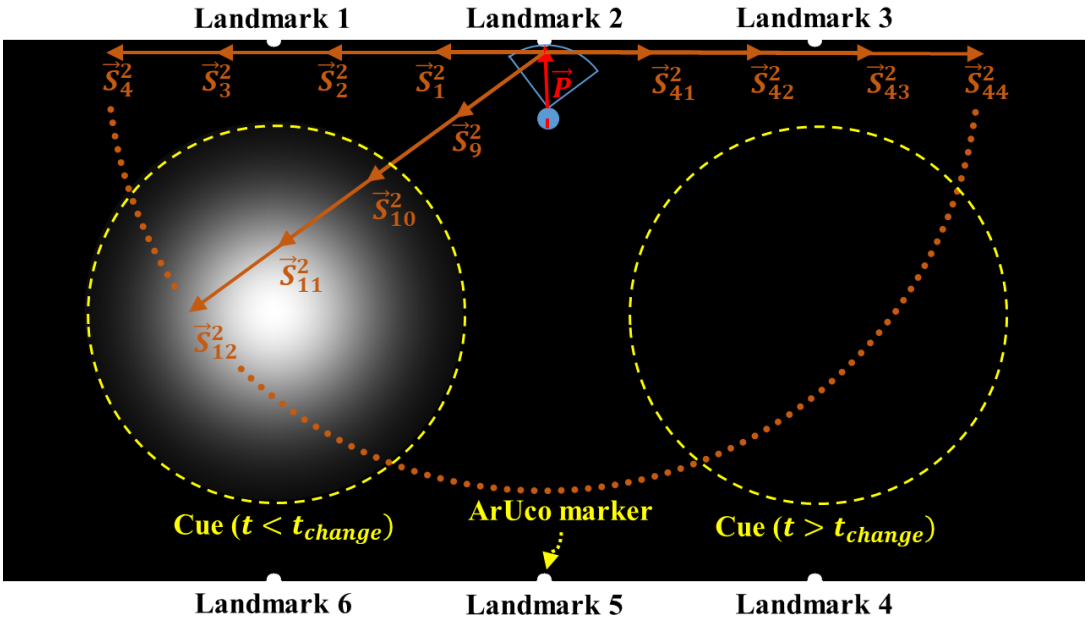


Figure 3.3: Action space of a robot in the LBA-RL method. The cyan vector represents the relative location of the robot with respect to the landmark. Each orange vector is the action ($\vec{S}_i^k$) that a robot executes by following $\vec{T} = \vec{P} + \vec{S}_i^k$ where superscript k represents the k^{th} landmark and index $i \in [1, 44]$ represents the i^{th} action. The blue arc represents the camera field of view of the robot. Dashed yellow circle on the left demonstrates the cue before t_{change}. After environment changes, new location of the cue is the yellow circle on the right.

range of ε. For simplicity of implementation, cosine waves were used. It must be noted the ε is the same for all the states, so this method of solving the exploration-exploitation dilemma is significantly cost-effective since it requires no memory and no heavy computations; hence, it is suitable for swarm robots.

In the proposed method, training and testing phases were not separated, and a robot learns and acts simultaneously. The reason for not considering a training phase is that the Q-learning algorithm modifies the Q-table according to the rewards received from the environment. If an arena were to be designed for the training phase, the Q-table of robots would learn and adapt to the conditions of the training arena. However, the test arena might have completely different conditions and dynamics. Therefore, the learned data in the training phase would be of no use in the test phase.

Algorithm 3: Landmark-Based Aggregation with Reinforcement Learning

```
1  Procedure LBA-RL
2      perform the BEECLUST method, algorithm Eq. (1)
3      if landmark k detected then
4          select an action based on ε-greedy policy Eq. (3.9)
5          update ε based on chosen schedule
6          calculate the displacement vector T⃗ according to Eq. (3.10)
7          turn ∠T⃗
8          while distance travelled < ‖T⃗‖ do
9              measure distance travelled according to Eq. (3.4)
10             follow T⃗
11             if robot detected then
12                 interrupt_flag ← 1
13                 /* do not update Q-value */
14                 measure cue intensity, I_c
15                 if I_c > 0 then
16                     calculate w_s according to Eq. (3.1)
17                     wait for w_s
18                 turn θ°
19                 break
20             else if obstacle detected then
21                 interrupt_flag ← 1
22                 r_t ← -1
23                 update Q-table according to Eq. (3.11)
24                 turn θ°
25                 break
26         if !interrupt_flag then
27             measure cue intensity, I_c
28             r_t ← I_c
29             update Q-table according to Eq. (3.11)
30     interrupt_flag ← 0
```

CHAPTER 4

EXPERIMENTAL SETUP

In this chapter, experiment settings for evaluation and investigation of the proposed method is provided. Kobot robots are chosen as the swarm robots for this study. Three different experimental platforms were used. First, kinematic simulations are employed where physical properties of robots are ignored and robots are modeled as circles with the same radius as a Kobot robot. These simulation are done via python programming language. Second, a realistic simulator is used where physical and three dimensional properties of robots are taken into account. Webots simulator [54] were used for this purpose. Third, real Kobot robots are used for experimenting studied methods to compare the results of simulators with real-world setups.

As for the performance metric, the normalized aggregation size (NAS), which is the number of robots aggregated on the cue divided by the population size, was used. For some experiments, the time evolution of the metric was shown to illustrate the transient behavior of the methods. For the others, the steady-state value of the metric was taken into account by averaging the last 100 samples of the metric. In this regard, the steady-state was assumed to be reached when the metric value reached and settled within the 5% range of its mean value of the last 500 samples.

In order to illustrate various aspects of the aggregation method, different experiments were implemented. All these experiments were repeated for a given number of Monte-Carlo trials in order to decrease the effect of unwanted factors on the evaluation, such as initial conditions and noise. More specifically, initial positions of robots within the arena followed a random distribution. Number of trials are stated in each experiment case.

In all kinematic simulation setups, robots were modeled as circles as shown in Figure 3.3. The parameters of the robots were chosen such that their size and speed were the same as the Kobot robots used in the real robot experiments. Radius of the robots were chosen to be $R_r = 0.06\,\mathrm{m}$, collision detection range was $R_{col} = 0.06\,\mathrm{m}$ with a $360°$ collision detection angle. The speed of the robots was $v = 0.14\,\mathrm{m\,s^{-1}}$. During the experiments, robots are considered to be homogeneous in terms of hardware and software, i.e., the aggregation method they are using.

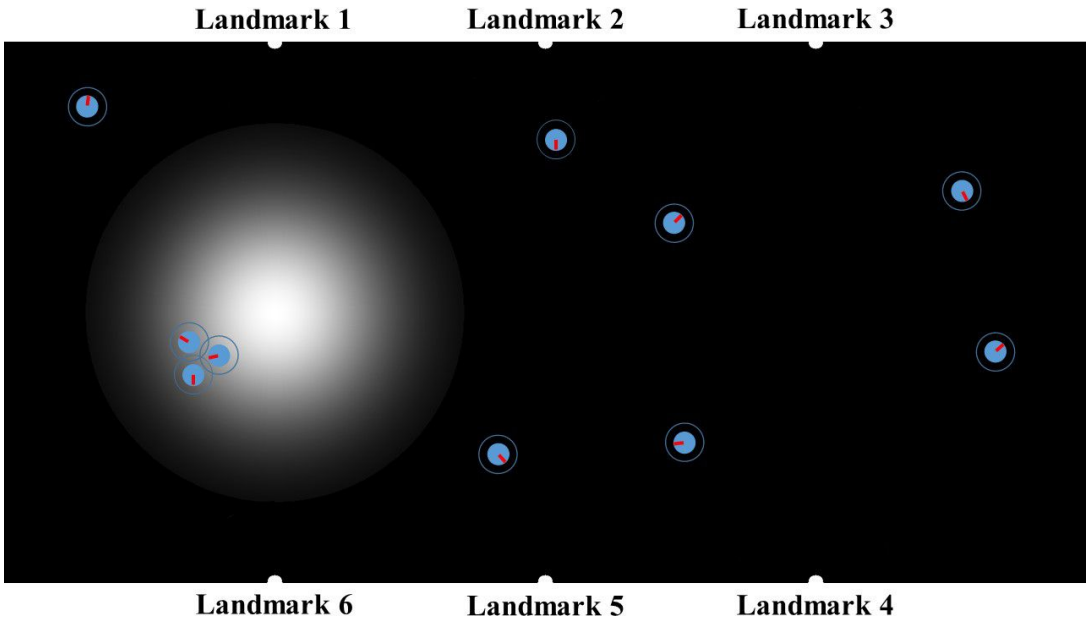
4.1 Landmark-based Aggregation

In the experimental analysis, two different simulation platforms were used. The first was the kinematic simulator, and the other was the realistic simulator. For the sake of comparison, in all the simulations, both the LBA method and the BEECLUST method were implemented. In kinematic simulator, the ODOCLUST method is implemented as well.

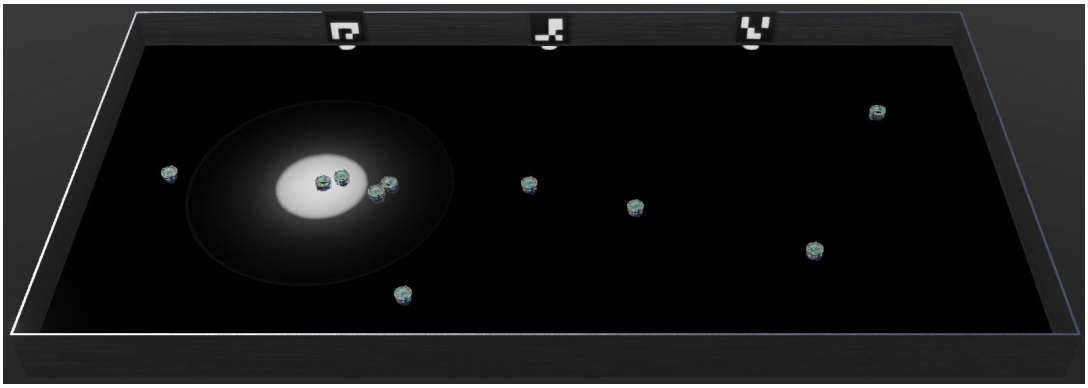
4.1.1 Kinematic Simulation

In kinematic simulations, shown in Figure 4.1(a), robots were modeled as two-wheeled agents that have a certain size, linear speed, and rotational speed but no dynamics.

A $5 \times 10\,\mathrm{m}^2$ rectangular arena was used for the simulations. A bright disk was placed on the arena as the cue. The intensity of the cue decreased gradually towards its perimeter. The distribution of the intensity of the cue along the radial direction was considered to be a 2D Gaussian centered at the cue. The cue radius was set to be $0.7\,\mathrm{m}$ in such a way that it was large enough to accommodate all robots on the cue [19]. The location of the cue was changed from $x = 2.5\,\mathrm{m}, y = 2.5\,\mathrm{m}$ (the left-hand side of the arena) to $x = 7.5\,\mathrm{m}, y = 2.5\,\mathrm{m}$ (the right-hand side of the arena at a predefined time t_{change} during the simulations that were being the dynamic experiments. In cases where the location of the cue is not changed during the simulation, like the first part of the above-mentioned setup, the experiment is considered static. Three distinct landmarks were placed evenly on each long side of the arena, making a total



(a) The Kinematic Simulator



(b) The Realistic Simulator

Figure 4.1: Simulation platforms. (a) The kinematic simulator. The blue inner and outer circles represent a robot and sensing range of its sensors, respectively. The red line represents the direction of a robot. Six white semicircles represent the location of landmarks. The cue is shown by a gray-white disc. (b) The realistic simulator. Robots and ArUco markers are simulated realistically.

Table 4.1: Standard values used in the LBA kinematic simulations

Parameter	**Description**	**Value / range**
w_a	arena width	$10\,\mathrm{m}$
h_a	arena height	$5\,\mathrm{m}$
R_c	cue radius	$0.7\,\mathrm{m}$
I_c	measured cue intensity	$[0\ \ 255]$
w_{max}	maximum waiting time	$120\,\mathrm{s}$
R_r	radius of a robot	$0.06\,\mathrm{m}$
R_{col}	range for robots and obstacles	$0.06\,\mathrm{m}$
N	number of robots	20
m	number of landmarks	6
τ_e	error threshold	4
σ_n	angular noise	15°

of six landmarks as shown in Figure 4.1(a). The detectable area of the landmarks is considered to be a rectangle with ($0.77 \times 1.55\,\mathrm{m}^2$). Since noise was taken into account, each experiment was repeated 20 times. The initial position and orientation of each robot were chosen randomly for each experiment. Unless otherwise noted, the standard values shown in Table 4.1 were used in all experiments.

- **Evaluation of the non-Adaptive LBA Method:** In these experiments, in order to study the LBA method without its adaptation capability, the error threshold was set to a very large number, such that error variables never reset, so that the LBA method became non-adaptive. The duration of the experiments was 80,000s, and the cue was changed at 40,000s. The static part of these simulations was taken longer than the other experiments in order to make sure that robots utilize all the landmarks during the first part of the experiments. Consequently, all the calculated total displacement vectors pointed to the old location of the cue in the second part of the simulation. This assumption was intentionally made to highlight the consequences of lack of adaptability for the LBA method.

- **Error Threshold Experiments:** In these experiments, the effect of error threshold on the performance of the adaptive LBA method was investigated using $\tau_e = \{1, 2, 3, 4, 6, 8, 10, 12\}$. The duration of the experiments was 80,000s and the cue was changed at 20,000s. In order to emphasize the effect of the error threshold, the angular noise value was taken larger than the standard value as $\sigma_n = 30°$. For the purpose of comparison, results of the ODOCLUST and BEECLUST aggregation methods are added as well.

- **Noise Experiments:** In these experiments, the effect of noise on the performance of the adaptive LBA method was investigated using $\sigma_n = \{0^\circ, 15^\circ, 30^\circ, \ldots, 180^\circ\}$. Only static experiments were performed with a duration of 20,000s.

- **Population Size Experiments:** In these experiments, the effect of population size was studied as in [19] using $N = \{10, 15, 20, 25, 30\}$ robots with a fixed arena size as shown in Table 4.1. The duration of the experiments were $20\,000\,\mathrm{s}$, and only static experiments were performed.

- **Cue Size Experiments:** In these experiments, the effect of cue size was investigated using $R_c = \{0.25, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 5.0\}\mathrm{m}$. The arena size was fixed and used as in Table 4.1 in the experiments.

4.1.2 Realistic Simulation

In realistic simulations, shown in Figure 4.1(b), all the actuators and sensors of Kobots were modeled using the Webots simulator [54]. No robot had access to any global information of either itself or other robots, and all the processing was done on board. Static experiments were performed using 10 simulated Kobot robots with a duration of $10\,000\,\mathrm{s}$. The arena size was scaled from the standard size to $3.55 \times 7.1\,\mathrm{m}^2$ and, similarly, the cue radius was taken as $0.5\,\mathrm{m}$. The intensity distribution of the cue

along the radial direction was also Gaussian as in the kinematic simulations. The error threshold was 4, and the experiments were repeated 5 times with random initial positions and orientations.

In the literature, landmarks in robotic scenarios were implemented in different ways, such as RFID tags [18] or QR-codes [55]. In this thesis, visual landmarks, namely, the ArUco markers [56] were chosen as the landmarks due to their simplicity and relatively low computational requirements. The OpenCV library [53] and Python were used to calculate the position and orientation of the ArUco markers. The vector $\vec{P}$ was estimated using the relative position, and orientation of the detected ArUco markers using the PnP algorithm [57]. The range of detection was realistically simulated based on the specifications of the camera used in the robots as illustrated in Figure 4.2. No additional noise was added other than the one present in the camera and the DC motors in the experiments. During realistic simulations, 2D images of ArUco markers with a size of $20 \times 20\,\mathrm{cm}^2$ were attached to the walls of the arena to represent the landmarks. Three ArUco markers were placed evenly on each long edge of the rectangular arena, making a total of six ArUco markers the same as the kinematic simulations. An ArUco markers' unique ID, its relative position, and relative orientation are estimated when detected by a robot.

4.2 Landmark-based Aggregation with Reinforcement Learning

Experiments were conducted using a kinematic-based simulator and real robots. In both setups experiments were repeated five times (five Monte-Carlo trials).

4.2.1 Kinematic Simulation

A rectangular arena with dimensions of $2.82\,\mathrm{m} \times 5.65\,\mathrm{m}$ was used. The total number of robots was chosen to be $N = 10$ in order to make the robot density low. The cue is represented as a bright circle inside the arena with a radius of $R_c = 1\,\mathrm{m}$. The intensity of the cue reduces towards its perimeter, and its distribution along its radius is considered to be a 2D Gaussian centered at the cue as shown in Figure 3.3. To test the performance of methods in non-stationary environments, location of the cue

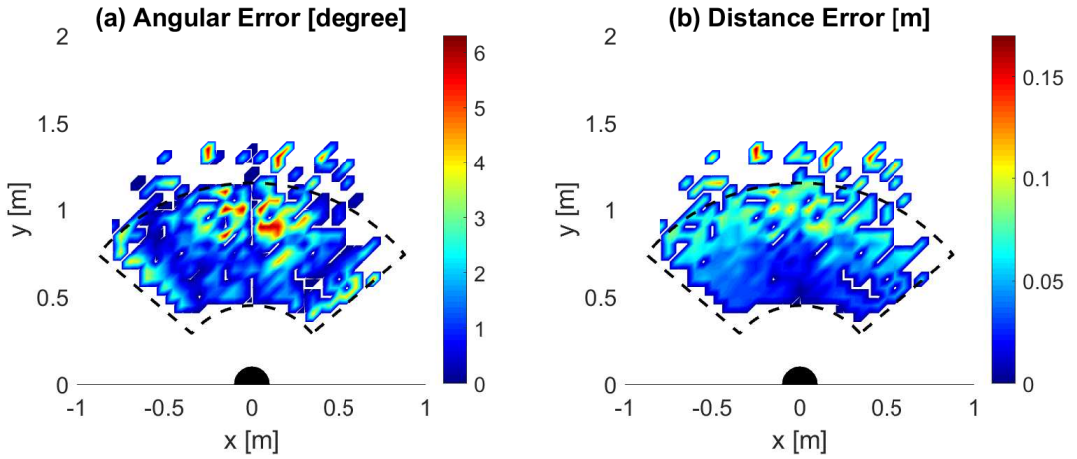


Figure 4.2: The error caused by image processing of a ArUco markers for different points of the arena. The ArUco markers (shown by the black half disk) was not detectable from the white points. The black dashed lines shows the approximate bounds of the detectable area of the ArUco markers. a) Angular error of the measured relative orientation compared to the actual orientation. b) Distance error of the measured relative distance compared to the actual distance.

was changed in a specific time, t_{change}, during the experiment from $x = 1.41\,\mathrm{m}, y = 1.41\,\mathrm{m}$ (the left-hand side of the arena) to $x = 1.41\,\mathrm{m}, y = 4.23\,\mathrm{m}$ (the right-hand side of the arena).

The duration of each experiment was chosen to be $t_{total} = 100,000\mathrm{s}$, which is long enough for all the methods to reach the steady-state. The location of the cue was changed at $t = t_{change} = 50,000\mathrm{s}$, i.e., in the middle of each experiment. For the LBA method, the error threshold parameter was chosen to be $\tau_e = 4$ since this choice has the benefit of both good adaptation characteristics for non-stationary environments and robustness to odometry noise.

In the arena, three landmarks were located with equal distances from each other on each side of the arena, making a total of six landmarks, $m = 6$, as shown in Figure 3.3. It should be noted that since landmarks represent the states of the Q-learning algorithm, increasing the number of landmarks will increase the size of the Q-table, increasing the training time. On the other hand, choosing fewer landmarks will reduce the probability of their detection; and hinder the performance of the LBA and

LBA-RL methods. By selecting the number of landmarks as $m = 6$ for our test cases, the Q-table does not become too large, and still, the performance of the LBA and LBA-RL methods are at acceptable levels. It was also assumed that robots could detect landmarks up to $0.5\,\mathrm{m}$.

For the LBA-RL algorithm, action space, $\mathcal{A}$, was designed to have $n = 44$ displacement vectors as shown in Figure 3.3. The length and angle of action vectors are discretized as $l \in \mathcal{L} = \{1.25, 2.5, 3.75, 5\}$ m and $\psi \in \ominus = \{0°, 18°, 36°, 54°, 72°, 90°, 108°, 126°, 144°, 162°, 180°\}$. It should be noted that the choice of l depends on the arena size. More vectors will span the arena better, and chances of finding actions with higher rewards will increase. However, since increasing the size of the action space will increase the Q-table size, training time will grow. On the other hand, a smaller action space size will cause faster learning, but the arena will not be spanned properly. Choosing the size of the action space as 44 is a fair trade-off between not having a large Q-table and spanning the arena sufficiently. Actually, in an aggregation problem, the action space is continuous, but solving the aggregation problem in continuous space is beyond the scope of this thesis. The parameters used in the kinematic simulation are shown in Table 4.2.

The following three experiments were performed using the kinematic-based simulator:

- **Environment Experiments:** In these experiments, the BEECLUST, LBA, and LBA-RL (with VDBE and cyclical ε schedules) methods were compared. First, this comparison was made without considering any odometry noise. Then, a noise, $\sigma_n = 15°$, was added to displacement vectors based on Eq. (3.7). Moreover, in both noisy and noiseless cases, the performance of the methods was measured in non-stationary environments. Time evolution of NAS values was plotted.
- **Noise Experiments:** In these experiments, the robustness of the methods against noise was studied. Noise was added using Eq. (3.7) with $\sigma_n \in \{0°, 5°, 15°, 30°, 45°, 60°90°, 135°, 180°\}$. For these experiments, the steady-state NAS values were plotted for the sake of clarity of the plots. The steady-state values were calculated by taking the average of the last 100 NAS values

Table 4.2: Standard values used in the LBA-RL kinematic simulation

Par.	**Description**	**Value / range**
w_a	arena width	$2.82\,\mathrm{m}$
h_a	arena height	$5.65\,\mathrm{m}$
R_c	cue radius	$1\,\mathrm{m}$
R_r	radius of a robot	$0.06\,\mathrm{m}$
R_{col}	range for robots and obstacles	$0.06\,\mathrm{m}$
I_c	measured cue intensity	$[0\ \ 255]$
w_{max}	maximum waiting time	$120\,\mathrm{s}$
t_{total}	total length of each experiment	$100\,000\,\mathrm{s}$
t_{change}	the moment when the location of the cue is changed	$50\,000\,\mathrm{s}$
N	number of robots	10
m	number of landmarks	6
τ_e	error threshold for LBA method	4
A	amplitude of cyclical waves	$\{0.25, 0.5, 0.75, 1\}$
p	period of cyclical waves	$\{50, 100, 150, 200\}$
σ_n	angular noise	$\{0°, 5°, 15°, 90°, 135°, 180°\}$
σ	inverse sensitivity of VDBE method	$\{0.01, 0.1, 1, 10\}$
γ	discount factor of LBA-RL	0
α	learning rate of LBA-RL	0.1
l	length of action vectors for LBA-RL	$\{1.25, 2.5, 3.75, 5\}\mathrm{m}$
ψ	angle of action vectors for LBA-RL	$\{18°, 36°, 54°, 72°, 90°, 108°, 126°, 144°, 162°, 180°\}$

of each run.

- **Parameter Sensitivity Experiments:** In this study set, the sensitivity of ε schedules for the LBA-RL method to different parameter values was studied. Two schedules were implemented for ε, which are VDBE and cyclical schedule. VDBE schedule has only one free parameter σ, whereas the cyclical schedule has two free parameters p and A, which are the period and amplitude of waves, respectively. As in the previous test case, steady-state NAS values were considered. For backing up the statements made in this experiment set, a random robot was selected from the swarm, and the rewards that it received and the change of its ε parameter were plotted over time.

4.2.2 Real-robot Experiment

In real-robot experiments, the latest version of Kobot robots (called Kobot) as shown in Figure 4.3 were used. Kobot differentially driven CD-sized swarm robot platform which was developed for self-organized flocking experiments. Thus, Kobots are upgraded with additional hardware and software in order to perform aggregation experiments. The main upgrades required were the capabilities added to sense the cue intensity and to detect ArUco markers.

For sensing the cue intensity, four IR sensors were placed at the bottom of the robot as the floor sensors. IR sensors were calibrated to read 0 for the lowest intensity zone (the carpet on the floor) and 255 for the highest intensity zone (the brightest zone at the center of the cue). Four readings were taken at each location, and these readings were then averaged to compensate for the tilt of the robot and for the non-uniformity of the floor/cue. Optical encoders coupled to the two DC motors were used as odometry sensors. Obstacle and robot detection was performed using the legacy range and bearing system that has eight modulated IR sensors. The range and bearing system have a range of $20\,\mathrm{cm}$, and it is able to distinguish robots from obstacles. For onboard image processing, landmark detection, and realization of aggregation methods, a Raspberry Pi 3B+ embedded computer was used as the central controller unit. Raspbian OS and ROS Melodic [58] were installed on Raspberry Pi 3B+.

Detection of ArUco markers was realized using the Raspberry Pi Camera v2.1 and the ArUco detection module of OpenCV [59]. Kobots can detect markers with a $0.05\,\mathrm{m}$ edge up to $2\,\mathrm{m}$ distance and from skew angles up to 80°. Nevertheless, detection distance was limited to $0.5\,\mathrm{m}$ and detection angle was limited to 45° to be consistent with the simulations.

During the experiments, all robots relied on onboard sensory data and computational resources, and there was no communication between the robots. Only high-level commands such as experiment start, end, and debugging messages were transferred to Kobots from the main computer. In terms of battery and power source, 2S (7.4 V Nominal) Li-Po 1300 mAh battery was used in Kobot. With this battery, Kobot could work up to two hours with a CPU load of 50% of the main controller.

For calculating the NAS values during the experiments, 2D poses of each robot were measured by using OptiTrack motion capture system having 8 USB cameras. Pose information from the camera array was transferred to the main computer running ROS Melodic for real-time NAS calculation and monitoring of the current state of the

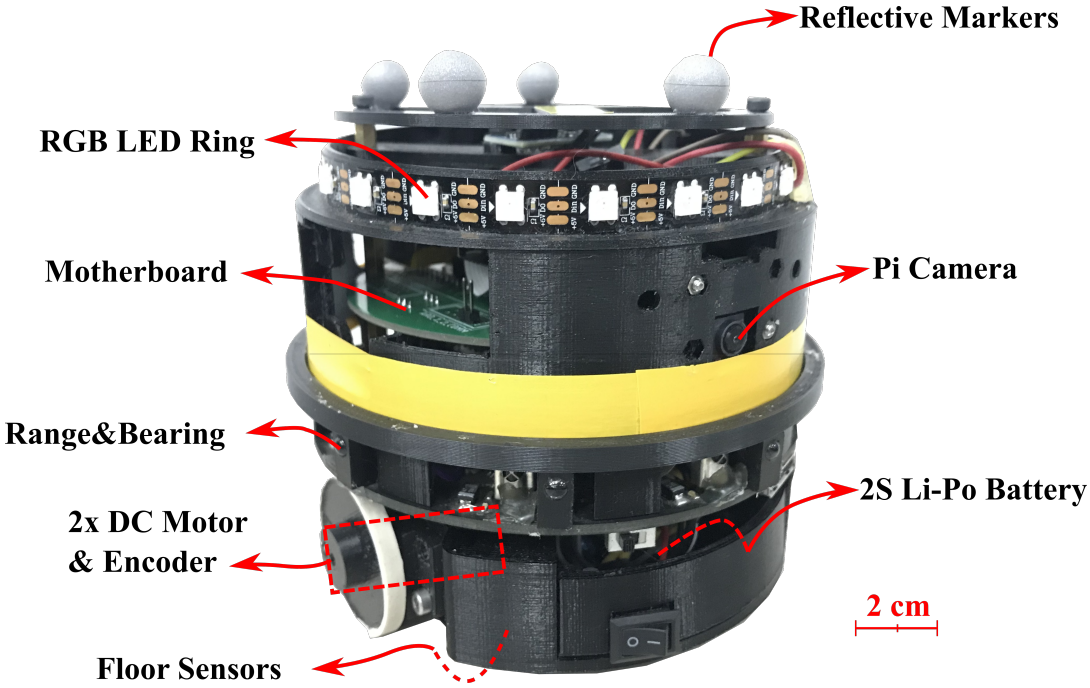


Figure 4.3: Kobot swarm robotic platform. The diameter and height of a Kobot is $12\,\mathrm{cm}$ and $11\,\mathrm{cm}$, respectively.

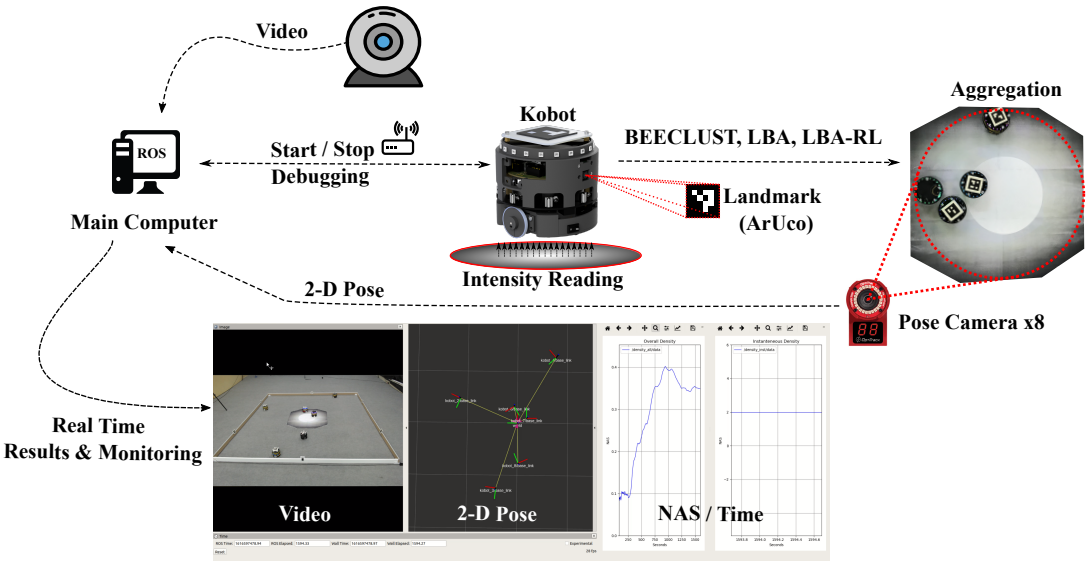


Figure 4.4: Real robot experiment setup. Position of each robot is calculated by eight pose cameras (Optitrack) located around the arena. Then, this data is sent to the main computer in order to calculate the number of robots inside the cue for derivation of NAS performance. A webcam is also located in the arena to record the experiments for documentation purposes. Kobots are utilized with on-board camera to detect the ArUco marker, i.e., landmarks.

experiments. By observing the real-time NAS value during the experiments, the time at which NAS reached a steady-state value was noted, and experiment duration was set accordingly. The setup for real robot experiments is shown in Figure 4.4.

The experimental setup consisted of a rectangular arena with six landmarks placed on the two sides of the rectangle and a circular cue with a white gradient indicating higher intensity regions as shown in Figure 4.5. Speed of robots were set to be $v = 0.14\,\mathrm{m\,s^{-1}}$. Real robot experiments were conducted in two different swarm sizes, $N = \{4, 6\}$ robots, to study the performance of the methods in two different population densities. The arena for $N = 4$ robots case was a $2.8\,\mathrm{m} \times 1.4\,\mathrm{m}$ with a cue radius of $0.3\,\mathrm{m}$. For the $N = 6$ robots case, arena was chosen to be $2.8\,\mathrm{m} \times 2.8\,\mathrm{m}$ with a cue radius of $0.45\,\mathrm{m}$. The arena for $N = 6$ robots setup is illustrated in Figure 4.5. For all three methods, maximum waiting time was set to be $w_{max} = 90\,\mathrm{s}$. The reason for choosing this waiting time is that lower waiting time will hamper the aggregation of robots, and no proper aggregation would be formed. On the other hand, time

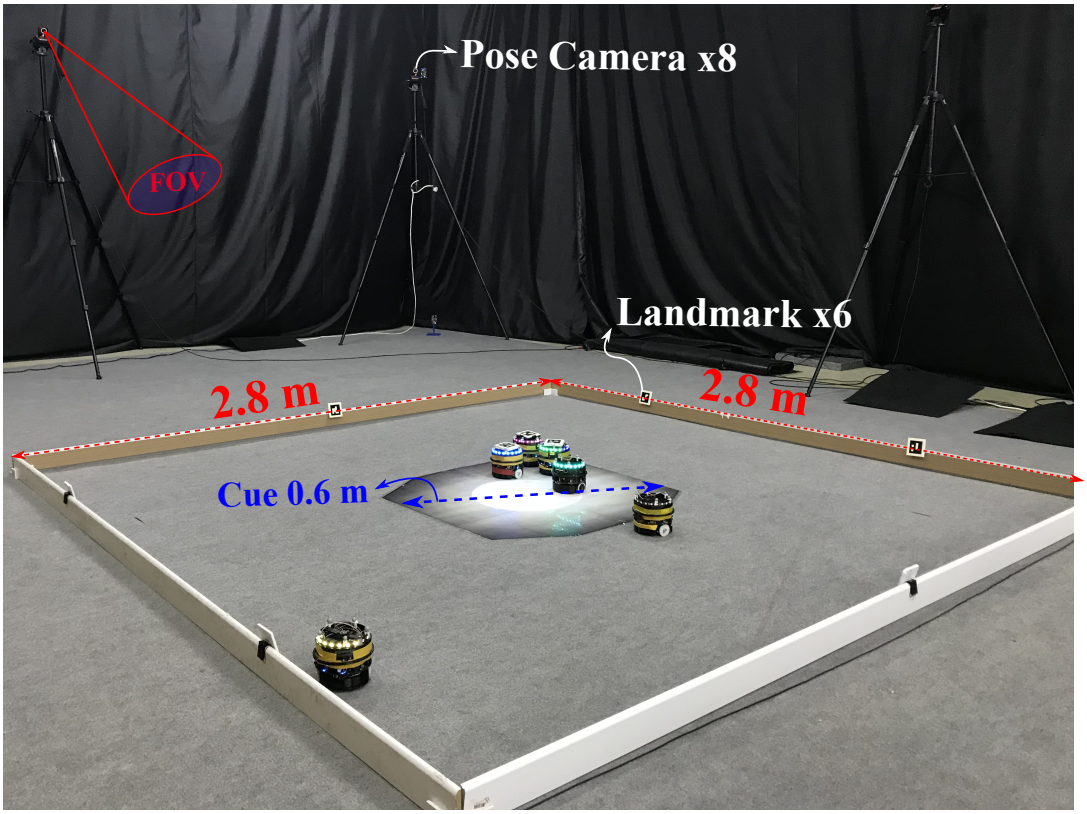


Figure 4.5: Arena setup for $N = 6$ robots experiment. A $2.8\,\mathrm{m} \times 2.8\,\mathrm{m}$ rectangular arena with six landmarks is surrounded by eight pose cameras (Optitrack) which are used for evaluating number of robots inside the cue. Robots are calibrated in a way that the grey color of carpet is read as zero cue intensity.

limitations forced by battery life of robots demanded a low waiting time. Otherwise, robots would be waiting most of the time inside cue until their battery dies. Thus, $w_{max} = 90\,\mathrm{s}$ found to be proper value for maximum waiting time. Table 4.3 shows the parameters used in the real robot experiments.

Three aggregation methods were implemented in the real robots: BEECLUST, LBA, and LBA-RL with cyclical schedule for ε. The error threshold parameter for the LBA method was chosen to be $\tau_e = 3$, and for the LBA-RL method, amplitude and period of cycles were chosen as $A = 1, p = 100$, respectively. For the real robot experiments, only static environment was studied, and no noise was added during the experiments. However, noise still existed due to natural reasons like wheel slippage and odometry errors. Despite the kinematic simulation setup where action space $\mathcal{A}$

Table 4.3: Standard values used in the real robot experiments

Par.	Description	Value / range
w_a	arena width	$2.82\,\mathrm{m}$
h_a	arena height	$\{2.82, 1.4\}\mathrm{m}$
R_c	cue radius	$\{0.3, 0.45\}\mathrm{m}$
w_{max}	maximum waiting time	$90\,\mathrm{s}$
t_{total}	total length of each experiment	$7200\,\mathrm{s}$
N	number of robots	$\{4, 6\}$
m	number of landmarks	6
τ_e	error threshold for LBA method	3
A	amplitude of cyclical waves	1
p	period of cyclical waves	100
γ	discount factor of LBA-RL	0
α	learning rate of LBA-RL	0.1
l	length of action vectors for LBA-RL	$\{1, 1.4\}\mathrm{m}$
ψ	angle of action vectors for LBA-RL	$\{36°, 60°, 90°, 120°, 144°\}$

consists of 44 displacement vectors, in real robot experiments, due to time limitations, the size of action space was reduced to 6 vectors. This reduced the learning time for the reinforcement learning algorithm considerably. The action space for the $N = 4$ robots setup is $l \in \mathcal{L} = \{1, 1.4\}\mathrm{m}$ and $\psi \in \ominus = \{36°, 90°, 144°\}$ and for the $N = 6$ robots setup is $l \in \mathcal{L} = \{1, 1.4\}\mathrm{m}$ and $\psi \in \ominus = \{60°, 90°, 120°\}$. Action space was chosen in a way that displacement vectors could span the arena properly.

A new set of kinematic-based simulations were performed with the same settings of the real-robot experiments for comparison. The only difference was that noise of $\sigma_n = 30°$ was added artificially in kinematic-based simulations to match the inherent noise in real robots. Results were reported as steady-state values of NAS.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Landmark-based Aggregation

5.1.1 Kinematic Simulation

5.1.1.1 Evaluation of the non-Adaptive LBA Method

The results of these experiments are depicted in Figure 5.1. In the first part of the experiment, with the LBA method, NAS reaches almost 0.8, whereas it is 0.3 with the BEECLUST method. In the second part of the experiment, in which the position of the cue is changed (as shown in Figure 3.2), NAS decreases dramatically, reaching almost 0 for both methods. The BEECLUST method recovers faster than the LBA method, both attaining a NAS of 0.3. As already mentioned, in order to give enough time for all the robots to utilize all the landmarks to learn the location of the cue, the cue changed at $40\,000\,\mathrm{s}$, so that all the landmarks point to the wrong location in the second half of the experiment.

With the LBA method, robots are able to use the landmarks in the environment, making them "learn" the location of the cue and aggregate rapidly on the cue around $10\,000\,\mathrm{s}$. When the position of the cue is changed at $40\,000\,\mathrm{s}$, the method cannot adapt to the change, and robots continue to move toward the "memorized" location of the cue where it does not exist anymore. This decreased the performance of the method dramatically. Since the BEECLUST method is based solely on random encounters of the robots, robots explore the environment without any preference, which makes the method have a mediocre performance, but it adapts to the changes rapidly.

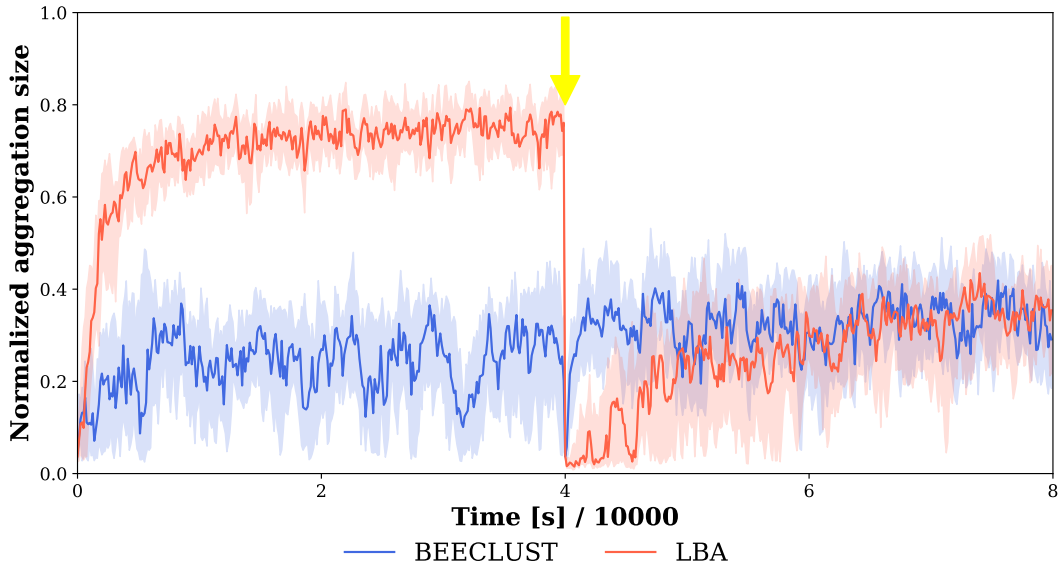


Figure 5.1: Non-adaptive LBA method experiments. The normalized aggregation size for the non-adaptive LBA and BEECLUST methods. The duration of the experiment is $80,000$s and the location of the cue changed at $40,000$s indicated by the yellow arrow. The blue and red lines are the mean performance of the non-adaptive LBA and BEECLUST methods for 20 repetitions, respectively. The shades show the 1^{st} and 3^{rd} quartiles of the results. All the other variable are taken as in Table 4.1.

In order to show the underlying differences between the LBA method and the BEECLUST method, the heat maps of the static part ($t \in [0, 40000]$s) of one of the simulations are depicted in Figure 5.2. The heat of the cue (shown as a red circle) is higher in the LBA method than in the BEECLUST method. Furthermore, the off-cue regions are colder and less dense for the proposed method; that is, robots wandered outside the cue less than the BEECLUST method since they are able to use the landmarks. This is also evident from the distribution of the temperature around the left-most top and bottom landmarks showing that robots are able to move directly toward the cue after they learn and detect the landmark. The evenly distributed temperature on the off-cue regions of the BEECLUST method is due to the stochastic nature of the method making the robots explore the environment randomly. A small detail to mention is the low presence of robots in the detectable area of landmarks. This means that when robots learn the displacement vector from a particular landmark, once the same landmark is detected, they go directly along the corresponding displacement vector

without any further exploration. These facts show that the LBA method is able to use the information available in the environment hence explore less than the BEECLUST method boosting its performance once robots learn the displacement vectors from the landmarks to the cue.

5.1.1.2 Error Threshold Experiments

In these experiments, the effect of the error threshold on the adaptability of the LBA method is investigated in a dynamic environment. Time evolution of the NAS values is depicted in Figure 5.3(a) for the LBA ($\tau_e = \{2, 10\}$), ODOCLUST and the BEECLUST methods. The steady-state NAS values for various error thresholds just before the change of the location of the cue ($20\,000\,\mathrm{s}$), after the change ($30\,000\,\mathrm{s}$) and at the end of the experiment ($80\,000\,\mathrm{s}$) are shown in Figure 5.3(b), c and d, respectively. In the first part of the experiments, before the change of cue position in Figure 5.3(a), due to the use of the landmarks, the NAS values of the LBA method rapidly reaches 0.7 then suddenly decreases to almost 0 at $20\,000\,\mathrm{s}$ due to the change of the position of the cue. After the change, robots with the error threshold equal to 2 have a faster transient response than the robots having an error threshold of 10 and adapt to the change of the cue much faster; both reaching a NAS value of 0.8 at the end of the experiment as shown in Figure 5.3(d). The BEECLUST method rapidly reaches a NAS value of 0.2 that also decreases rapidly during the change of the position of the cue. After the change, a NAS value of 0.3 is reached. The ODOCLUST method reaches a NAS value of 0.5 that decreases to almost zero during the change of the cue. After the change, NAS reaches its original value.

The error threshold changes the behavior of the LBA method drastically. In the static part of the experiment ($t \in [0\,,20\,000\,]\mathrm{s}$), the prominent factor is the noise (taken as $\sigma_n = 30°$). A higher error threshold compensates for the effect of errors due to noise increasing the performance as shown in Figure 5.3(b) and 5.3(d). However, the performance saturates for threshold values greater than three since the noise is not high enough to necessitate higher threshold values. In other words, a higher error threshold increases the robustness of the method to noise. The situation is just the opposite when the adaptability of the method is considered. Higher error threshold

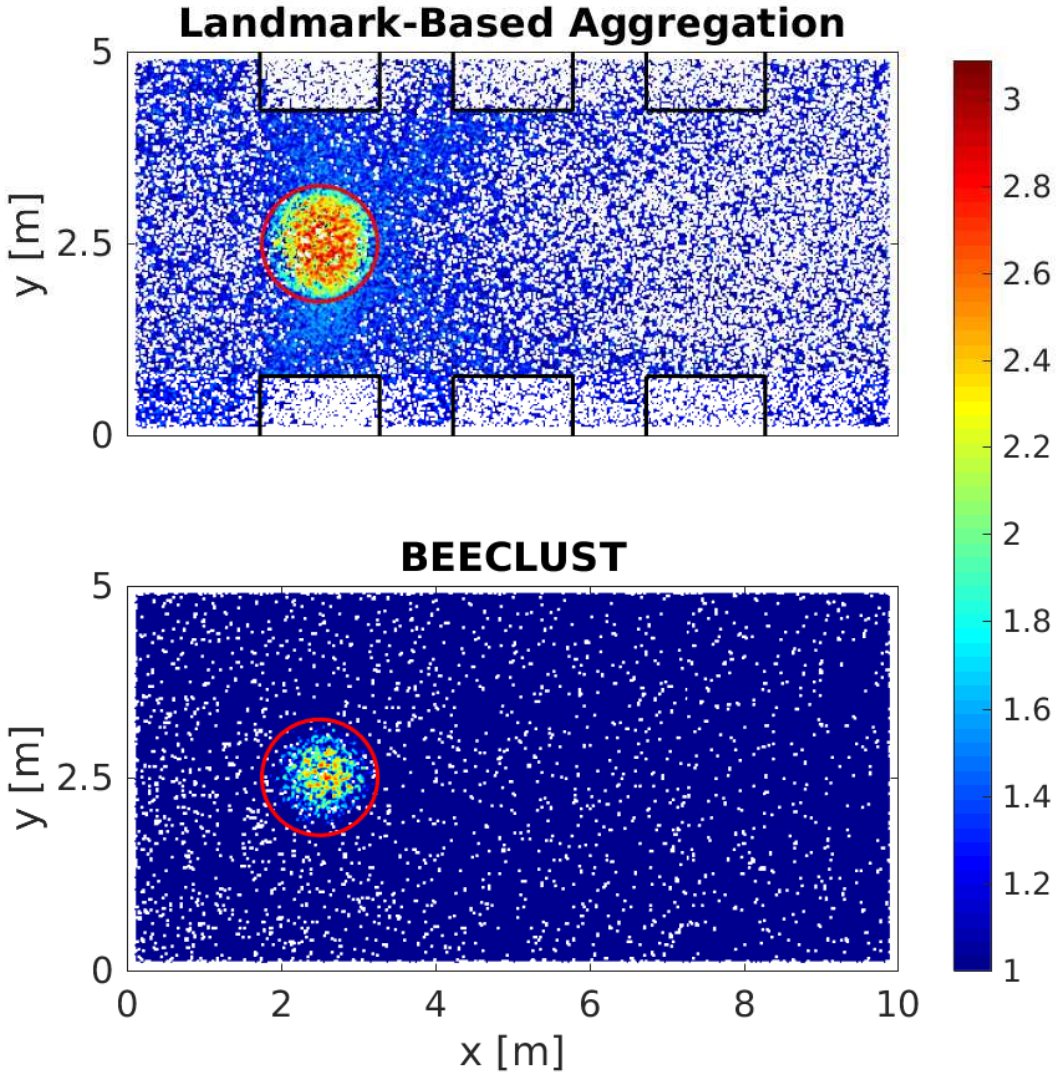


Figure 5.2: Non-adaptive LBA experiments. Heat maps of a randomly selected experiment of the non-adaptive LBA method (top) and the BEECLUST method (bottom). Positions of the robots are accumulated during $t \in [0\,, 40, 000]$s and results are color coded. The red circles represent the cue, and the black rectangles in the top figure represent the detectable regions of the corresponding landmarks.

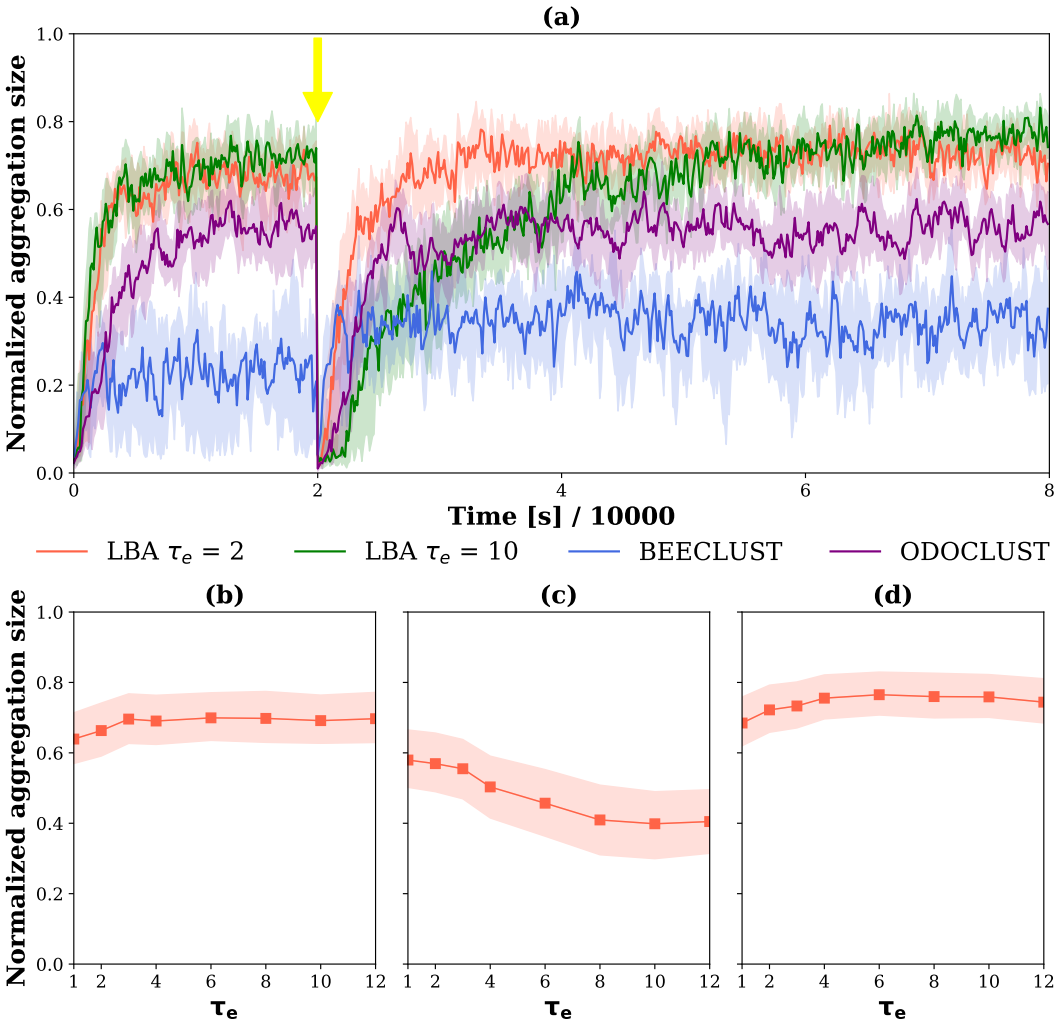


Figure 5.3: Error threshold experiments. The normalized mean aggregation size for the adaptive LBA method for: (a) $\tau_e = 2$ (red line), $\tau_e = 10$ (blue line), the BEECLUST method (green line), the ODOCLUST method (purple line). The steady-state normalized mean aggregation size for the adaptive LBA method for $\tau_e = \{1, 2, 3, 4, 6, 8, 10, 12\}$, (b) before the cue change (20,000s), (c) short time after the cue change (30,000s), (d) at the end of the experiment (80,000s). The duration of the experiment is 80,000s and the position of the cue is changed at 20,000s indicated by the yellow arrow. The noise is taken as $\sigma_n = 30°$.

values make robots stubborn, and they cannot adapt to the changes in the environment rapidly, as shown in Figure 5.3(c).

5.1.1.3 Noise Experiments

In these experiments, the effect of noise on the performance of the LBA method is analyzed in a static environment, and the steady-state results are depicted for $\tau_e = 4$ in Figure 5.4. For the LBA method, the NAS values decrease as noise increases from $0°$ to $180°$. Noise was not added to the BEECLUST method since it does not have any effect on its performance, for the robots explore the environment in a fully random manner in the BEECLUST method. The NAS values of the BEECLUST method are shown for comparison purposes, and it is around 0.2.

As discussed in the previous chapter, noise has a negative effect on the performance of the LBA method. Though the overall performance decreases, it can still be compensated by the error threshold. Increasing the error threshold as shown in Figure 5.3(b) increases the performance when the noise is constant. The advantage of the BEECLUST method is that it is totally robust against noise, albeit its low performance when compared to the LBA method. It can be deduced that when $\sigma_n = 180°$ the performance of the LBA method is almost the same as the BEECLUST method since noise suppresses all the information gained from the environment. As it can be observed in Figure 5.4 around $180°$, there is still a performance difference between the two methods. This is due to the fact that the detectable area of landmarks makes the robots not enter these regions when using the LBA method as seen in Figure 5.2, which statistically speaking, increases the probability of finding the cue hence the performance a bit.

5.1.1.4 Population Size Experiments

In population size experiments, the size of the arena is kept fixed, and the number of robots increased from 5 to 45, and the steady-state NAS values are depicted in Figure 5.5 for the LBA and BEECLUST methods. For the LBA method, the NAS values increase, reaching 0.7, then decrease to 0.6 as the number of robots increases.

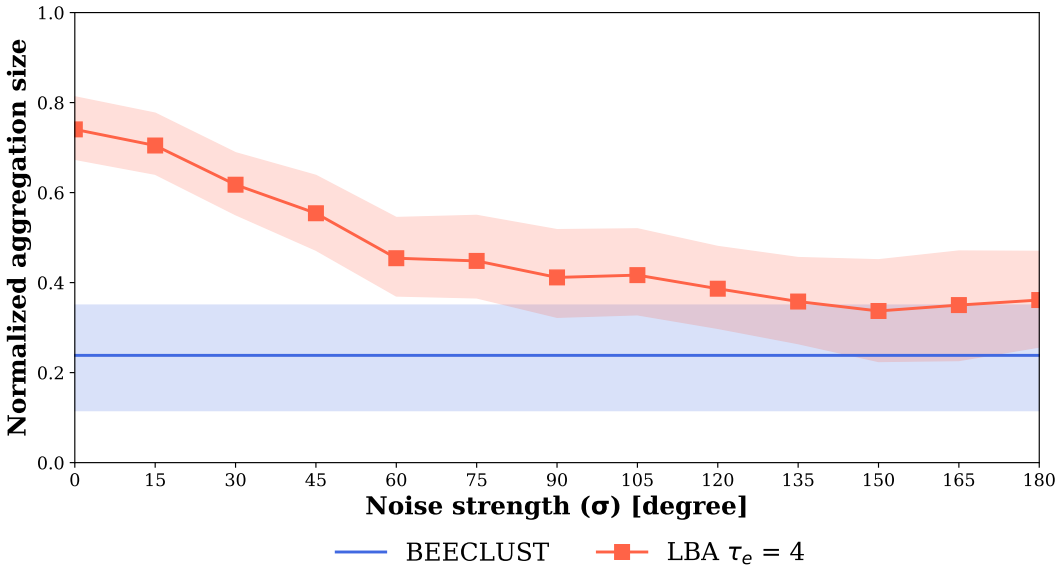


Figure 5.4: Noise experiments. The steady-state normalized mean aggregation size for the adaptive LBA (red line) and BEECLUST (blue line) methods. For the adaptive LBA method the noise is taken as: $\sigma_n = \{0°, 15°, 30°, \ldots, 180°\}$.

For the BEECLUST method, the trend is somehow different, NAS increases from a very small value, 0.1, to 0.4 where it is saturated as the number of robots increases.

When the LBA method is considered, increasing the population size (or the density of robots) increases its performance (up to 20 robots), then the performance saturates and starts to decrease slowly as the size increases. The performance increase is expected since a robot needs another robot to calculate the displacement vector (so that robot learns its way from the landmark to the cue) and to wait on the cue (so that aggregation happens) else the robot does not make intensity measurement and misses the cue as shown in the last part of Algorithm 2. So, when the number of robots increases, there is a higher chance for a robot to meet another robot increasing the learning rate and performance of the method. The reason behind the decrease in performance is more subtle; when the population size is larger than a certain value, the probability of encounter another robot off-the-cue increases. And if a robot is moving towards the cue using the total displacement vector, it would stop, measure the cue intensity (that would be 0 since it is an off-the-cue encounter), turn randomly, and move towards this random direction instead of going straight to the cue, decreasing the performance of the LBA method. However, when the BEECLUST method is

considered, increasing the population size increases the performance reaching a NAS value of 0.3. The reason for the increase in performance is the increased probability of encounters of the robots on the cue. The performance saturates around 0.3 due to the overcrowding of robots in the arena. The performance difference between the two methods is more prominent in low densities (or low population size). High robot density hinders the performance of the LBA method due to overcrowding preventing to use of environmental information.

5.1.1.5 Cue Size Experiments

In the cue size experiments, the size of the arena is fixed, yet the size of the cue is increased from 0.25 to 10, and the steady-state NAS values are depicted in Figure 5.6 both for the LBA and BEECLUST methods. When the cue size increases, the performance of the LBA method increases sharply, saturating at 1.0 at a cue radius of $5\,\mathrm{m}$. The same trend is also observed for the BEECLUST method; an increase in the cue radius increases the method performance that gets saturated at a cue radius greater than $5\,\mathrm{m}$.

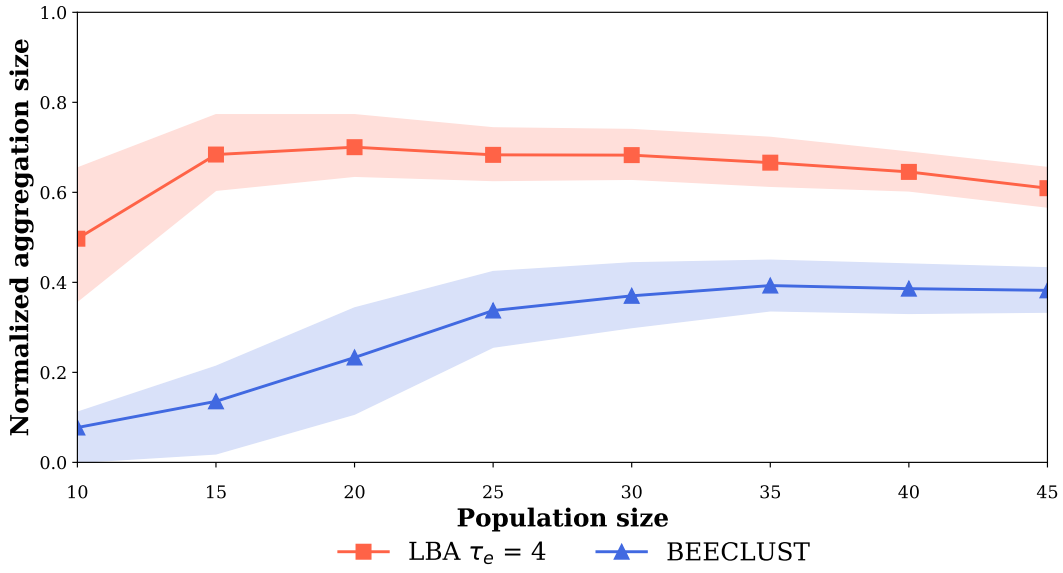


Figure 5.5: Population size experiments. The steady-state normalized mean aggregation size for the adaptive (red line) and BEECLUST (blue line) methods. The population size is taken as: $N = \{10, 15, 20, 25, 30\}$ robots.

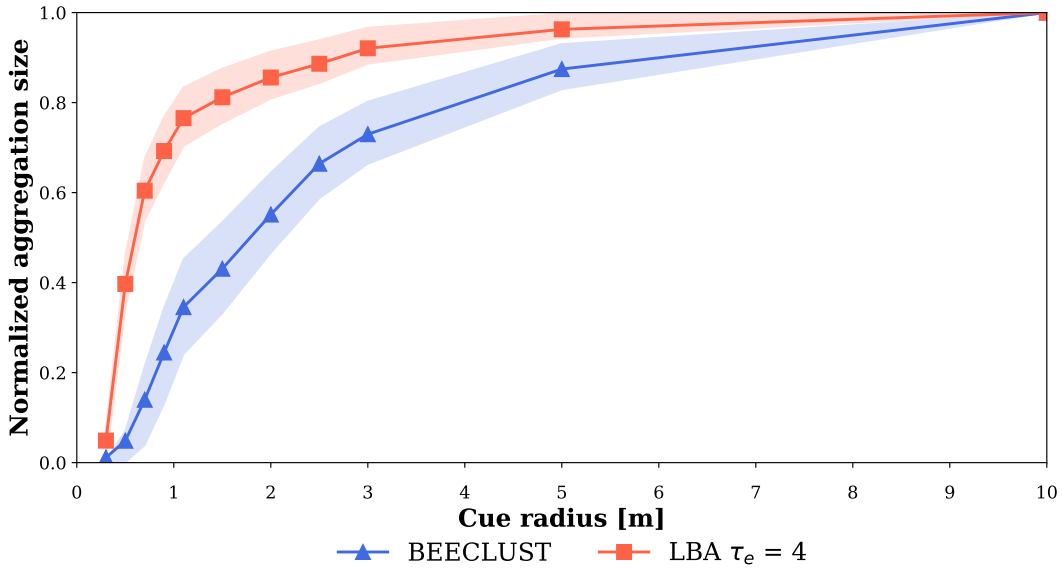


Figure 5.6: Cue size experiments. The steady-state normalized mean aggregation size for the adaptive LBA (red line) and BEECLUST (blue line) methods. The cue radius is taken as, $R_c = \{0.25, 0.5, 0.75, 1.0, 1.5, 2.0, \ldots, 5.0\}\mathrm{m}$.

When the LBA method is considered, the increase in the cue size increases the probability of finding the cue. This increases both the learning rate of total displacement vectors and the probability of finding another robot on the cue, causing a rapid increase in the performance as seen in Figure 5.6. Since the performance of the BEECLUST method solely depends on the random encounters of robots on the cue, increasing the size of the cue increases the number of robots on the cue, hence increasing the performance of the BEECLUST method. The rate of performance increase is higher for the LBA method since it also utilizes environmental information. The last point to note is that the performance difference between the two methods decreases, similar to the population size experiments, as the cue size increases. In the limiting case, the performance of the BEECLUST method catches that of the LBA method. That is an expected result since when the cue radius is $5\,\mathrm{m}$, almost $75\,\%$ of the arena is covered by the cue, and robots do not need landmarks to find the cue anymore.

5.1.2 Realistic Simulation

These simulations are performed to illustrate the applicability of the LBA method in a realistic setting using the simulated models of all actuators and sensors, including the camera, as discussed in the previous chapter. Due to heavy computational load, a systematic investigation of all parameters has not been performed using realistic simulations; instead, a representative set of values is investigated, and the results for the LBA and BEECLUST methods are depicted in Figure 5.7. Both methods show a similar trend, though the LBA method performs better than the BEECLUST method. The kinematic and realistic simulation results are very similar, showing that the kinematic simulations are reliable.

One of the main concerns in realistic simulations is noise. Since there is inherently noise in the actuation and sensing of robots in realistic simulations, the noise was not added artificially during the simulations. The noise in detecting the ArUco markers affects both the estimation of the relative orientation of the ArUco markers $\angle\vec{P}$, and the magnitude of the initial relative position vector $|\vec{P}|$. In order to investigate the noise in ArUco markers detection further, an additional experiment was performed. In this experiment, a robot was located in different locations on the arena, its camera being directed toward the ArUco markers indicated by a black semi-circle on the x-axis. Then, the gathered image by the camera was processed based on the OpenCV library, and the $\angle\vec{P}$ and $|\vec{P}|$ were estimated. The error of estimation was then calculated, and the results for the angular and distance errors were depicted as a heat map in Figure 4.2(a) and 4.2(b), respectively.

In both plots, there are regions shown in white indicating that the positions where the robot was not able to detect the QR-code. When the angular error is considered, an asymmetric pattern of positive and negative relative angles is observed. The heat map is mostly blue, indicating that the angular error is relatively small, almost below $2°$. Moreover, the detectable region has a bow shape, and the range is from $-45°$ to $+45°$. The distance error plot is half blue and half light blue, indicating that the error is mostly below $0.05\,\mathrm{m}$. The mean and standard deviation of the angular and distance errors are calculated, and the results are shown in Table 5.1.

Table 5.1: Realistic simulation errors.

Variable	Mean	Standard Deviation
$\lvert\angle\vec{P} - \angle\hat{\vec{P}}\rvert$	$0.62°$	$1.21°$
$\lvert\lvert\vec{P}\rvert - \lvert\hat{\vec{P}}\rvert\rvert$	$0.02\,\mathrm{m}$	$0.03\,\mathrm{m}$

In both of the plots, the maximum error is observed around the central axis of the detectable region that is furthest from the ArUco marker. Since the angle is calculated using the slope of the vertical lines of the rectangle on the periphery of the ArUco marker, a few pixel errors cause a big change in the slope for very small relative angles resulting in larger errors than the moderate relative angles. Finally, the undetectable points (white regions) can be categorized into three groups; a) those which are too close to the landmark, where the camera could not capture the whole ArUco marker; b) the points which are too far from the ArUco marker and due to the limited resolution of the image, the detection is not achieved; c) the regions having a larger relative angle with respect to the ArUco marker.

5.2 Landmark-based Aggregation with Reinforcement Learning

5.2.1 Kinematic Simulation

5.2.1.1 Environment Experiments

Time evolution of NAS values for the three methods without and with odometry noise are depicted in Figure 5.8. In the experiments without noise, as shown in Figure 5.8(a), all the methods, except the BEECLUST method, showed a similar steady-state performance during the first half of the experiment reaching a NAS value of 0.7. When the cue location changed, the LBA method adapted to the change faster than the LBA-RL method. The performance of the BEECLUST method was the lowest, and it was not affected by the change of the cue due to its random nature.

The LBA method showed the best transient performance and adapted quickly to the change of the cue. The LBA-RL method with the VDBE schedule was able to reach

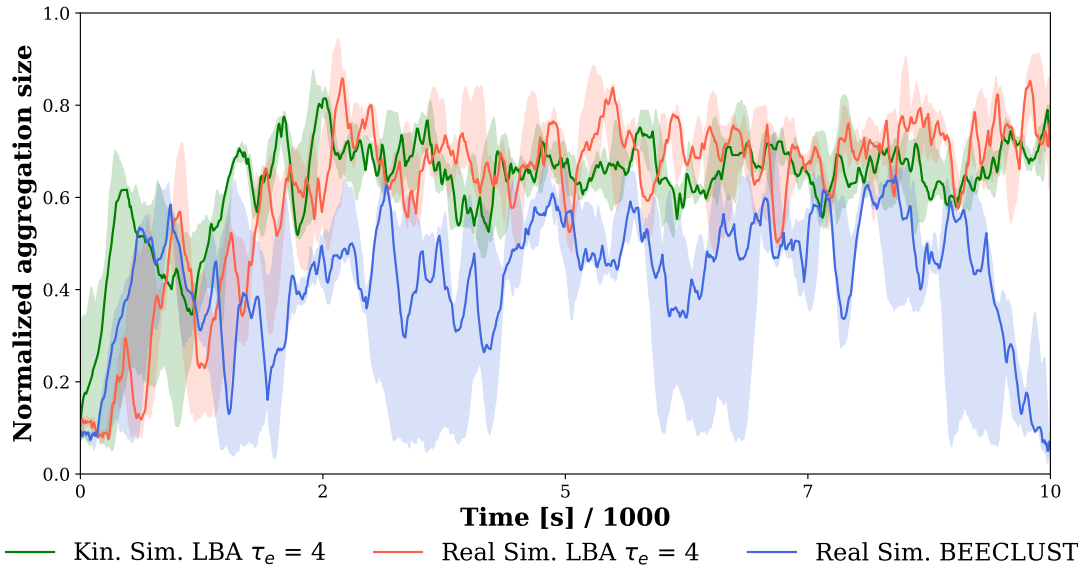


Figure 5.7: Realistic experiments. The time evolution of the normalized mean aggregation size for the adaptive LBA method with the realistic simulation (red line), with the kinematic simulation (green line) and the BEECLUST method with the realistic simulation (blue line). The population size is 10 robots. Only static experiments with a duration of $10\,000\,\mathrm{s}$ are performed. The experiments are repeated for 5 times for each setting. For the kinematic simulations, the noise is taken as: $\sigma_n = 15°$. The error threshold is taken as 4.

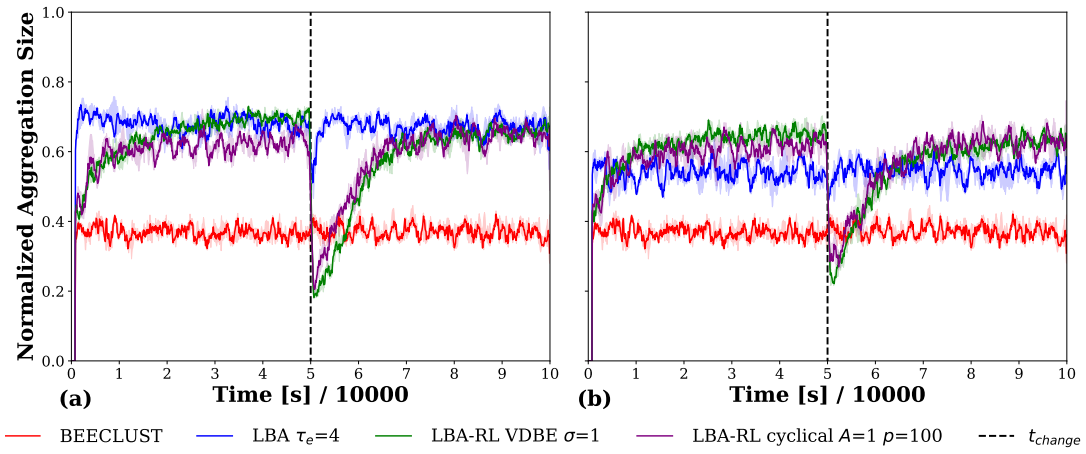


Figure 5.8: Environment Experiments. Time evolution of the NAS values of the BEECLUST method, the LBA method with $\tau_e = 4$, the LBA-RL method with the cyclical schedule, $p = 100$, and with the VDBE schedule, $\sigma = 1$ are shown. (a) The NAS values without odometry noise and (b) the NAS values with odometry noise, $\sigma_n = 15°$. Shades represent the first and third quarterlies and the solid lines are the median of 5 trials. Duration of each experiment is $t_{total} = 100,000$s. The location of the cue is changed at $t_{change} = 50,000$s, indicated by the vertical dashed line. In order to smoothen the results, moving average with a window of $[t_s, t_s + 500]$ is used.

the performance of the LBA method. The LBA-RL method with a cyclical schedule also showed a similar performance level with the LBA-RL method with the VDBE schedule. The only difference being the oscillations observed in the cyclical schedule due to the cyclical updates of ε. Moreover, the response of the LBA-RL method is slower than the LBA method, and it also adapts slower to the change of cue position. This is due to the fact that it takes time for the robots to explore and learn the environment so that they can form the Q-table. Consequently, it takes a while for a robot to discover the changes in the environment and adapt accordingly. In addition, the learning speed is controlled by the learning rate, α, and it is taken as a fixed value, $\alpha = 0.1$ in this thesis . This restricts the learning speed, hence slows down the response of the LBA-RL. In the LBA method, since no learning model is used, it works faster than the LBA-RL method.

The BEECLUST method showed the worst performance achieving a NAS value of 0.37. This is due to the fact that the robot density was low, decreasing the probability of encounter the robots; hence, the performance of the BEECLUST method decreases considerably. The good point about the BEECLUST method is that its performance is not affected by the change of cue location. The reason for that is inherent in the low performance of the method. Since robots cannot form large aggregates on the cue, changing the location of the does not affect its performance.

The results of the experiments with odometry noise, $\sigma_n = 15°$, are depicted in Figure 5.8(b). When compared to the experiments without noise, the performance of all methods except the BEECLUST method decreased. The LBA method has the most drastic decrease from a NAS value of 0.7 to 0.55, whereas the LBA-RL method showed a slight decrease from 0.7 to 0.67. This difference comes from the fact that the performance of the LBA method depends heavily on the odometry data. For calculating the total displacement vector $\vec{S}^k$, a robot needs to integrate each displacement vector, $\vec{S}_i^k$, that it traversed to reach the cue after detecting the k^{th} landmark. The length and angle of each $\vec{S}_i^k$ are computed using odometry data. Therefore, a slight amount of noise like $\sigma_n = 15°$ affects the LBA method considerably. On the other hand, the LBA-RL method also uses the odometry data to execute an action, but it does not integrate displacement vectors. Therefore, it is more robust to odometry noise. The performance of the BEECLUST method did not change since it does not

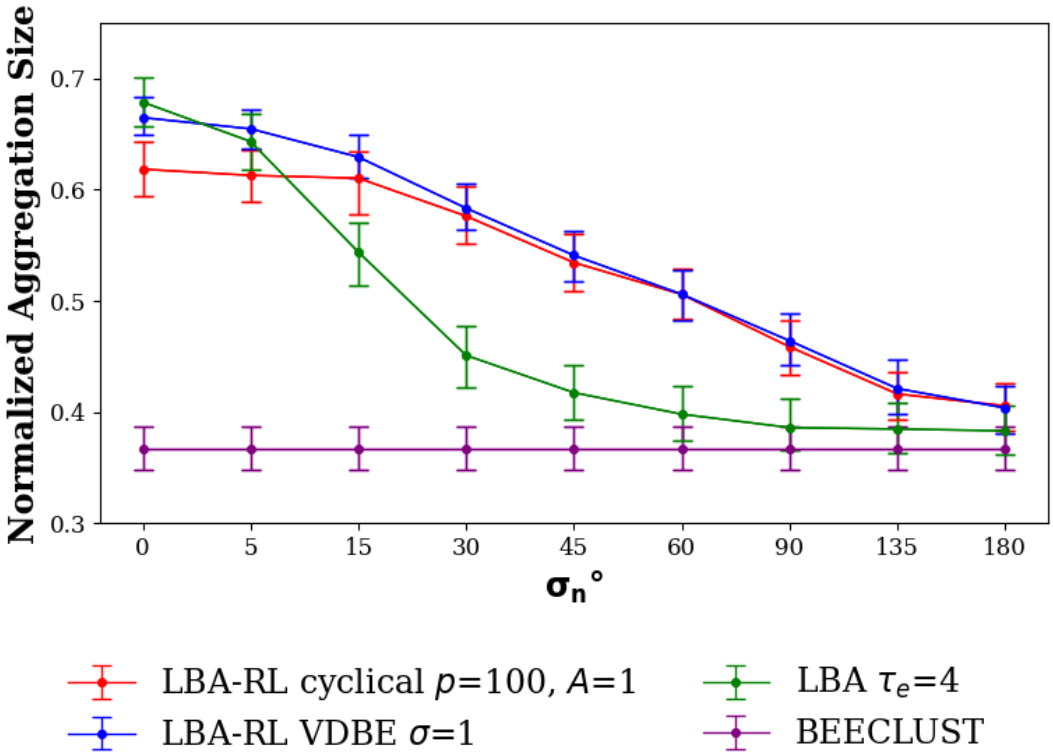


Figure 5.9: Noise experiments. The steady-state values of NAS are shown for the BEECLUST, LBA, LBA-RL with cyclical schedule, and LBA-RL with VDBE schedule methods with respect to the odometry noise, $\sigma_n \in \{0°, 5°, 15°, 30°, 45°, 60°90°, 135°, 180°\}$. Bars represent the first and third quarterlies and lines are the median of 5 trails. The experiments are static and duration of each experiment is: $t_{total} = 50,000$s. The x-axis is not drawn to scale.

use any odometry data, hence robust to odometry noise.

5.2.1.2 Noise Experiments

The results of the noise experiments are depicted in Figure 5.9. In the zero noise case, the LBA method showed the best performance having a NAS value of ≈ 0.7; however, its performance decreased dramatically noise increased. This drop in performance is expected since the LBA method depends highly on the odometer readings, as also discussed in the previous chapter. The performance of the LBA-RL method also decreased due to noise, especially in higher noise regions, but still, it is more

robust to noise than the LBA method. Since the BEECLUST method does not use the odometry readings, its performance did not change and stayed around 0.37 in all the settings. One important thing to note is when $\sigma_n = 180°$, the performance of all methods converged to BEECLUST since the information acquired from landmark is not exploitable anymore due to the intense amount of noise.

5.2.1.3 Parameter Sensitivity Experiments

The steady-state values of NAS versus the model parameters are depicted in Figure 5.10. The first and second rows represent the steady-state values before and after the change of location of the cue. The leftmost and middle columns are the plots of the period, p, and amplitude, A, parameters of the cyclic schedule, and the last column are the plots of the inverse sensitivity parameter, σ for the VDBE schedule. For the cyclical schedule, the period of waves does not affect the aggregation performance when $p \leq 10$. When $p \geq 10$, the performance increases considerably and remains around a NAS value of 0.65. Although large periods do not affect the steady-state value of the performance, they cause oscillations of NAS values, as shown in Figure 5.11. Therefore, provided that $p \geq 10$, cyclical schedules are not sensitive to the period. The same is also true for the amplitude parameter. It does not affect the performance that remains around 0.65. For both of the parameters, the results are the same before and after the change of the cue.

For the VDBE schedule, σ drastically changes the performance of aggregation. Before the change of the cue, higher σ values yielded better results. However, once the location of the cue was changed, higher σ values failed to put the robot back into the exploration state; hence the aggregation performance decreased considerably. For instance, when $\sigma = 50$, the NAS value before the change of the cue was approximately 0.7, and after the change, it dropped to 0.3, which is even worse than the performance (NAS=0.37) of the BEECLUST method. This is because robots do not return to the exploration mode; hence robots use the previously learned Q-table in the new environment. As a result, the performance becomes worse than the random choice.

In order to understand the performance difference between two schedules, the time evolution of ε for the cyclical schedule with $p = 100$ and $A = 1$ and the VDBE

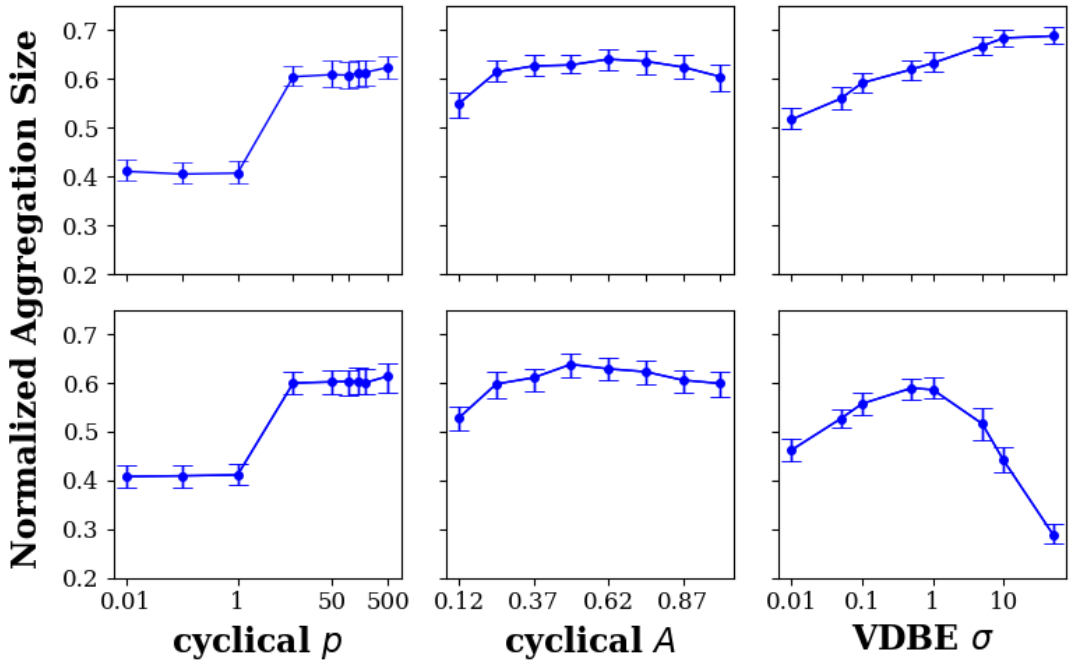


Figure 5.10: Parameter Sensitivity Experiments. The steady-state values of NAS are shown for different model parameters of LBA-RL method with cyclic schedule and LBA-RL method with VDBE schedule. Top and bottom rows show the results before and after the change of cue location, respectively. Leftmost and middle columns show the results for the period, p (A is taken as 1), and amplitude, A (p is taken as 100) parameters of the cyclic schedule, respectively. The rightmost column shows the results for the inverse sensitivity (σ) parameter of the VDBE schedule. Bars represent the first and third quarterlies and lines are the median of 5 trails. Duration of each experiment is $t_{total} = 100,000$s. The location of the cue is changed at $t_{change} = 50,000$s, indicated by the vertical dashed line. The leftmost and rightmost plots are drawn in semi-log scale.

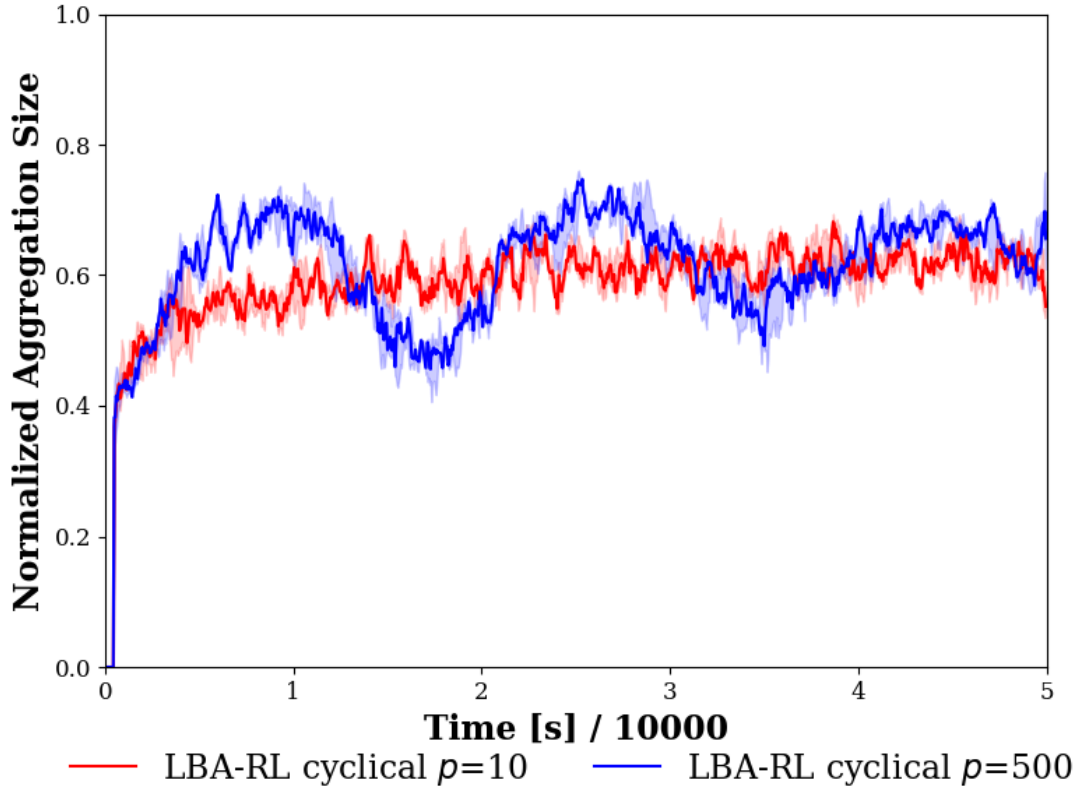


Figure 5.11: Demonstration of effect of period parameter, p, of cyclical schedule of the LBA-RL method on performance of the swarm. Shades demonstrate the first and third quarterlies and lines are the mean of 5 trials of the experiment.

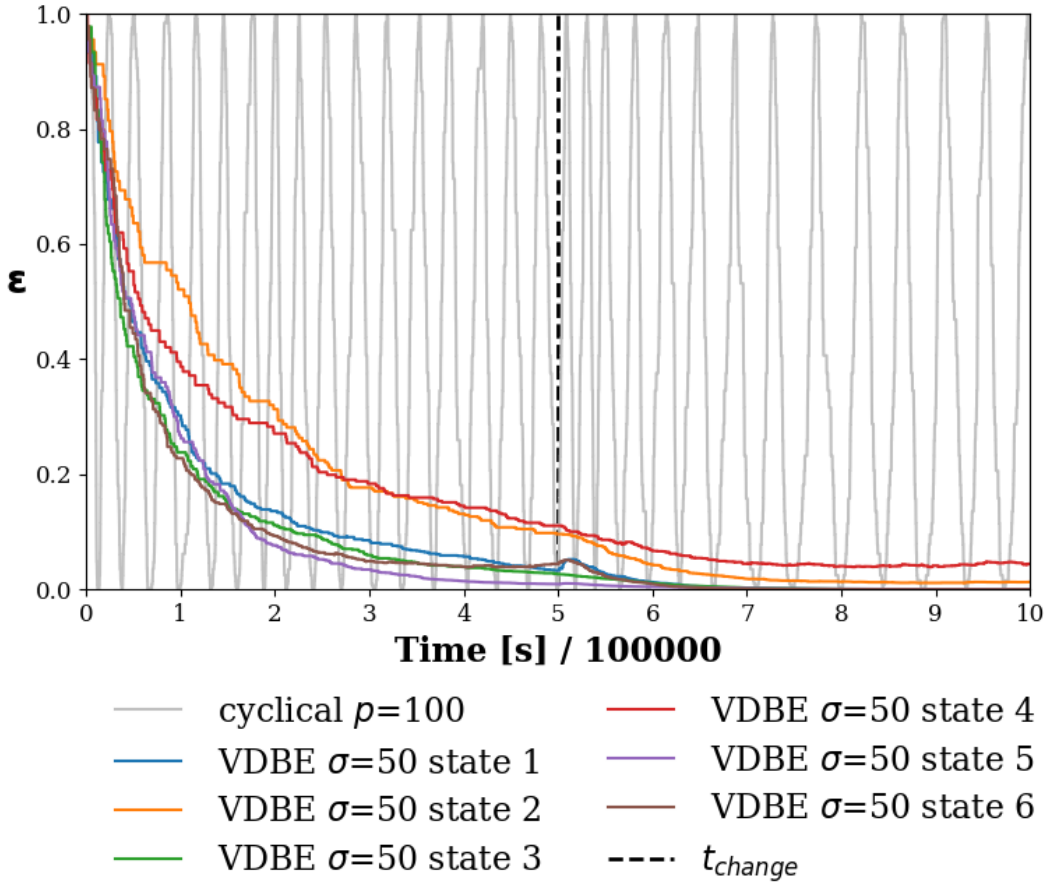


Figure 5.12: Evolution of ε during time for VDBE schedule with $\sigma = 50$ and cyclical schedule with $p = 100$ for a randomly selected robot. Occasional straight line regions in ε is due to the fact that ε is updated at every epoch, i.e., when the robot detected a landmark. So, ε is a function of epoch, not time.

schedule with $\sigma = 50$ using the data from a randomly selected robot is depicted in Figure 5.12. As expected, for the cyclical schedule, ε changes periodically based on p and A. VDBE schedule updates ε for each state separately, one line was drawn for each state making a total of six lines. ε values decrease monotonically for all the states with the VDBE schedule. This means that robots exploit more as time passes. This is beneficial when the location of the cue does not change as observed in Figure 5.10 (rightmost top plot), but if it changes as in the non-stationary experiments, the aggregation performance decreases considerably as observed in Figure 5.10 (rightmost bottom plot). This phenomenon is not observed in the cyclical schedule since robots half of the time explore and half of the time exploit.

In order to explain this phenomenon further, rewards received by a randomly selected robot for the cyclical and VDBE schedules are depicted in Figure 5.13. As expected, rewards that the VDBE schedule receives were higher than the cyclical schedule before the change of the cue. However, after the change, the VDBE schedule keeps ε low as shown in Figure 5.12 and fails to adapt to the new location of the cue. As a result, rewards were lower than the cyclical schedule after the change of the cue. For the cyclical schedule, ε oscillates; therefore, it can constantly adapt to changes in the environment.

5.2.2 Real-Robot Experiment

The BEECLUST, LBA, and LBA-RL methods were implemented using real robots. For the sake of comparison, the same experimental setup was also implement using the kinematic-based simulator. The steady-state NAS values for real robot and simulation-based experiments with 4 and 6 robots are shown in Figure 5.14(a) and Figure 5.14(b), respectively. In 4-robot experiments, the BEECLUST method has the lowest performance with a steady-state NAS value of 0.45, the LBA method has a steady-state performance of 0.58, and the LBA-RL method has the best performance reaching up to 0.62. In 6-robot experiments, a similar trend in performance has been observed. The results of kinematic-based simulation experiments are in accordance with the real robot experiments.

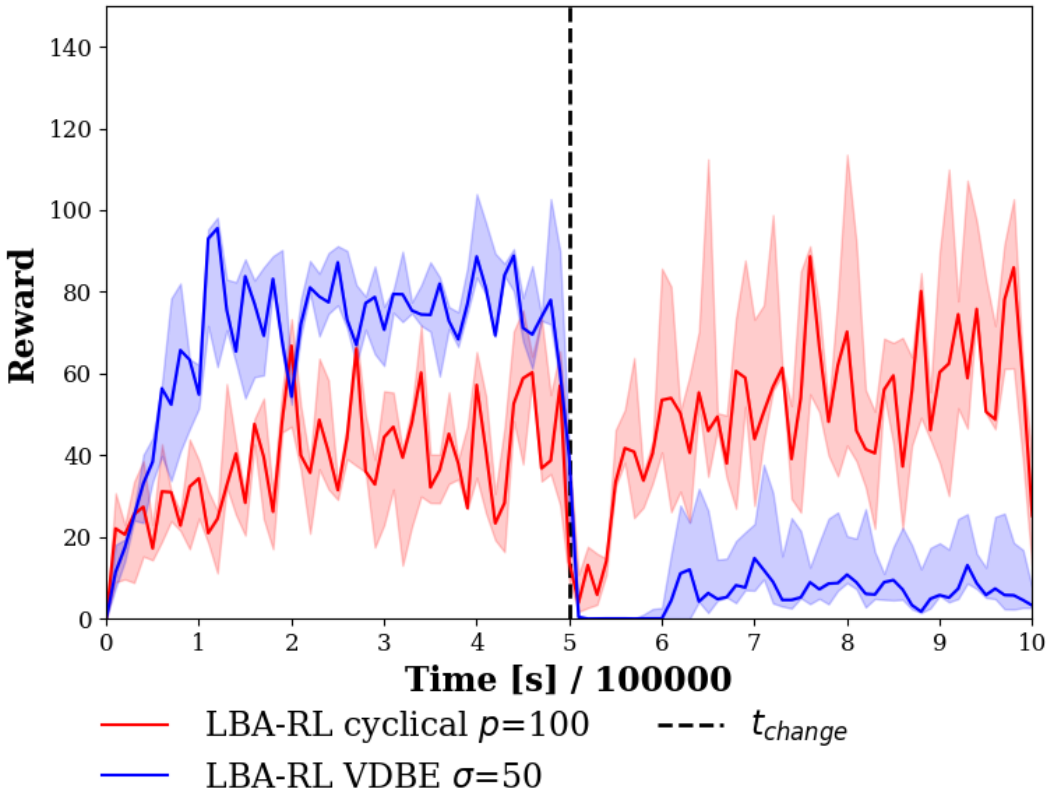


Figure 5.13: Rewards received by a randomly selected robot during time for LBA-RL method with cyclical ε schedule with a period of $p = 100$ and an amplitude of $A = 1$ and VDBE schedule with $\sigma = 50$. Shades represent the first and third quarterlies and lines are the median of 5 trails. Duration of each experiment is $t_{total} = 100,000$s. The location of the cue is changed at $t_{change} = 50,000$s, indicated by the vertical dashed line.

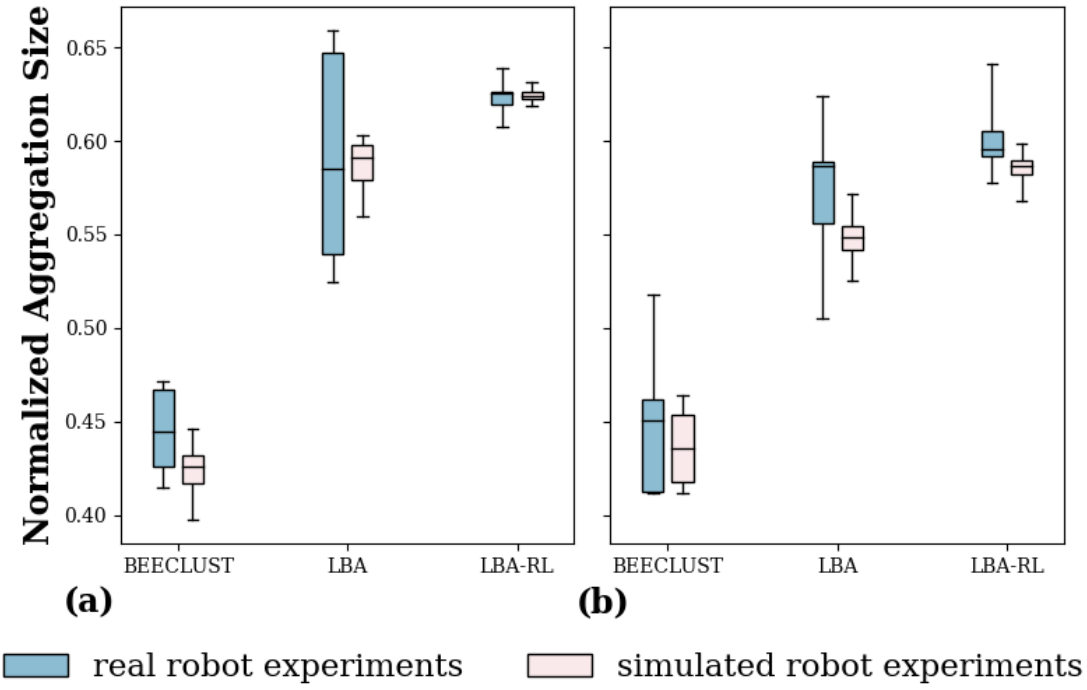


Figure 5.14: Steady-state values of NAS performance of the BEECLUST, LBA, and LBA-RL methods for real robots experiments (a) with 4 robots and (b) for 6 robots. LBA-RL method is implemented with cyclic schedule using $A = 1$ and $p = 100$. The boxes represent the median, first, and third quartiles. Whiskers are the minimum and maximum values. The pink boxes represent the results of kinematic-based simulation results and the blue boxes are real robot experiment results. Experiments are repeated five times.

5.3 Discussion

The experiments supported the idea that compared to the BEECLUST, the LBA method makes robots wander less off-the-cue and move toward the cue as soon as they meet a landmark. It was proven that exploiting the data of the landmarks significantly increases the aggregation performance when the environment is static. In a dynamic environment, however, the situation was more challenging. The dynamic feature of such environments makes the old information not valid anymore; hence, relying on outdated data is detrimental to the aggregation performance. The concept of the adaptive algorithm is introduced to cope with this challenge by using an error variable and a corresponding error tolerance threshold. By so doing, the proposed method reacts to the changes in the environment, makes the robots able to erase the wrong perceptions of the environment, and subsequently update them. By evaluating the performance of the LBA method in dynamic environments, it was revealed that the proposed method was more adaptive to the environment changes than the BEECLUST method. Thus, it was concluded that the proposed method not only outperformed the original BEECLUST method in stationary environments but also was able to adapt faster to the changing environment. Hitherto, we draw the conclusion that for a dynamic environment, non-adaptive LBA is not able to aggregate the swarm, thus considering the adaptive feature of the method is inevitable.

The study on different error threshold parameters, τ_e, demonstrated that in a noisy environment, increasing the threshold makes the system more resistant to the noise. On the contrary, the effect of the threshold on the adaptation rate is the opposite; that is, decreasing τ_e increases the agility of the method to adapt to new conditions of the environment. It is essential to mention that the effect of noise, as a demonstration of uncertainties in the environment, on the aggregation performance is destructive. The degradation of the performance versus noise is such that after a critical value of noise strength, using the information of landmarks did not make a change. Consequently, it can be concluded that for the environments with a high degree of uncertainties, it is not justifiable to employ the LBA method, and it is not reasonable to afford the cost.

Similar to another study [19], on the effect of the population density, the results bore the conclusion that the BEECLUST method lacks a proper performance in low-

density swarms. More than that, the difference between the LBA method and the BEECLUST method widened for the swarms with low-density populations. Thus, an important conclusion to be made here is the fact that in such cases adopting the LBA method significantly increases the performance. In the sample experiment, the amount of increase was so much that even a four times denser swarm of robots with the BEECLUST method could not outperform the LBA method. As one of the main conclusions of this thesis, compared to increasing the number of robots with the BEECLUST method, equipping robots with the LBA method costs less yet performs better.

The effect of cue size on the aggregation performance was also studied in the last investigation, which reflected the fact that in an environment with a low density of cue, exploiting the environmental information plays a vital role in the aggregation performance of robots. Therefore, in an environment with a low probability of finding the cue, storing data and referring to them can considerably boost the aggregation performance.

In the environment experiments (Figure 5.8), since the robot density was low, the BEECLUST method had the worst performance. For the experiments without noise, the LBA method showed the best performance. However, its performance dropped drastically in the experiments with an even slight amount of noise (Figure 5.9). The LBA-RL method was more robust to noise than the LBA method. This is also observed in real robot experiments (Figure 5.14) where odometry noise was inherently present.

The LBA-RL method uses the Q-learning technique, and ϵ-greedy policy was chosen as the policy to tackle the exploration-exploitation dilemma. In order to use the ϵ-greedy policy effectively in noisy and dynamic environments, ε must be scheduled. VDBE [46] scheduling schemes were implemented, and a new cyclical schedule was proposed by getting inspiration from [49]. The LBA-RL method with the VDBE schedule achieved the highest performance with proper tuning of the model parameters (Figure 5.9). However, its performance is very sensitive to model parameters (Figure 5.10). On the other hand, the LBA-RL method with cyclical schedule is more robust to odometry noise (Figure 5.9), less sensitive to model parameters (Fig-

ure 5.10), and performs better in non-stationary environments (Figure 5.8). Furthermore, the cyclical schedule requires less computational power and memory, suitable for simple swarm robots (Figure 5.14).

All in all, the LBA-RL method with cyclical schedule satisfies two requirements proposed in this thesis. (1) It is more robust to odometry noise, (2) it is less sensitive to model parameters, and (3) it is applicable to real robots. But, it has the demerit of dependency of its action space on the dimensions of the arena.

CHAPTER 6

CONCLUSION

In this thesis, a novel cue-based aggregation method was proposed (LBA-RL) based on Q-learning and ϵ-greedy policy with the proposed cyclical update schedule. First, landmarks were added to the environment, and robots learned the relative position of the cue with respect to the landmarks by using odometry sensors and path integration. It was shown that Landmark-Based Aggregation (LBA) outperformed the BEECLUST method in static environments, but the LBA method was not able to adapt to changes in the environment. Hence, an adaptive approach was employed to make it possible for robots to detect and remove outdated or flawed information and start learning from the beginning. This modification caused the LBA method to outperform the state-of-the-art cue-based aggregation methods (BEECLUST and ODOCLUST) in both static and dynamic environments.

Despite promising results of the LBA method, it depends heavily on odometry data which makes it susceptible to odometry sensor noise and uncertainties. Thus, reinforcement learning algorithm was employed to make the LBA method robust to uncertainties. This new approach was named as Landmark-Based Aggregation with Reinforcement Learning (LBA-RL). Through systematic analysis with the kinematic-based simulations and real robots experiments, it was shown that the proposed method shows better performance in the presence of odometry noise and environment uncertainties when compared to the BEECLUST method [9] and the LBA method [60]. A new approach to solve the exploration-exploitation dilemma was proposed to schedule ε parameter of ϵ-greedy policy by getting inspiration from [49]. The proposed cyclical schedule was compared to other state-of-the-art approaches, and it was shown that cyclical updates are less sensitive to model parameters and do not require fine-

tuning. Additionally, cyclical updates of ε make robots more robust to changes in the environment.

Nevertheless, the proposed aggregation method requires a priori information about the dimensions of the arena. Therefore, as future work, the DDPG algorithm [61] will be used to avoid requiring a priori information about dimensions of the arena. Furthermore, an adaptive approach will be considered to tune the learning speed of the algorithm. Moreover, the possibility of sharing information via a pheromone-based communication system might help the swarm to increase the performance of aggregation. Another important aspect is to analyze the number, distribution, and position of landmarks and their effects on aggregation performance as a future study. Besides, the presence of static or dynamic obstacles was not considered in this work. In future works, stationary and moving objects will be added to the scenario, and the performance of the proposed method will be evaluated under these new circumstances. Last but not least, the setup for multiple cues with different cue intensities will also be studied, and the performance of the proposed method and the BEECLUST method will be compared.

REFERENCES

[1] A. Vardy, "Aggregation in robot swarms using odometry," *Artificial Life and Robotics*, vol. 21, no. 4, pp. 443–450, 2016.

[2] D. Grünbaum and A. Okubo, "Modelling social animal aggregations," in *Frontiers in mathematical biology*, pp. 296–325, Springer, 1994.

[3] J. Krause, G. D. Ruxton, G. Ruxton, I. G. Ruxton, *et al.*, *Living in groups*. Oxford University Press, 2002.

[4] W.-J. Rappel, A. Nicol, A. Sarkissian, H. Levine, and W. F. Loomis, "Self-organized vortex state in two-dimensional dictyostelium dynamics," *Physical review letters*, vol. 83, no. 6, p. 1247, 1999.

[5] S. Camazine, J. Deneubourg, N. Franks, J. Sneyd, E. Bonabeau, and G. Theraula, *Self-organization in Biological Systems*. Princeton Studies in Complexity, Princeton University Press, 2003.

[6] J. Halloy, G. Sempo, G. Caprari, C. Rivault, M. Asadpour, F. Tâche, I. Saïd, V. Durier, S. Canonge, J. M. Amé, *et al.*, "Social integration of robots into groups of cockroaches to control self-organized choices," *Science*, vol. 318, no. 5853, pp. 1155–1158, 2007.

[7] D. D. Frank, G. C. Jouandet, P. J. Kearney, L. J. MacPherson, and M. Gallio, "Temperature representation in the drosophila brain," *Nature*, vol. 519, no. 7543, pp. 358–361, 2015.

[8] H. Heran, "Untersuchungen über den temperatursinn der honigbiene (apis mellifica) unter besonderer berücksichtigung der wahrnehmung strahlender wärme," *Zeitschrift für Vergleichende Physiologie*, vol. 34, no. 2, pp. 179–206, 1952.

[9] T. Schmickl and H. Hamann, "Beeclust: A swarm algorithm derived from honeybees," *Bio-inspired Computing and Communication Networks. CRC Press (March 2011)*, 2011.

[10] S. Kernbach, R. Thenius, O. Kernbach, and T. Schmickl, "Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system," *Adaptive Behavior*, vol. 17, no. 3, pp. 237–259, 2009.

[11] A. Sadeghi Amjadi, M. Raoufi, A. E. Turgut, G. Broughton, T. Krajník, and F. Arvin, "Cooperative pollution source exploration and cleanup with a bio-inspired swarm robot aggregation," in *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 469–481, Springer, 2020.

[12] B. Hölldobler, E. Bert Hölldobler, F. Holldobler, E. Wilson, and H. Wilson, *The Ants*. Belknap Press of Harvard University Press, 1990.

[13] M. V. Srinivasan, "Honey bees as a model for vision, perception, and cognition," *Annual review of entomology*, vol. 55, pp. 267–284, 2010.

[14] T. S. Collett, "Insect navigation en route to the goal: Multiple strategies for the use of landmarks," *Journal of Experimental Biology*, vol. 199, no. 1, pp. 227–235, 1996.

[15] J. Reinhard, M. V. Srinivasan, D. Guez, and S. W. Zhang, "Floral scents induce recall of navigational and visual memories in honeybees," *Journal of Experimental Biology*, vol. 207, no. 25, pp. 4371–4381, 2004.

[16] J. Reinhard, M. V. Srinivasan, and S. Zhang, "Scent-triggered navigation in honeybees," *Nature*, vol. 427, no. 6973, p. 411, 2004.

[17] T. S. Collett and M. Collett, "Memory use in insect visual navigation," *Nature Reviews Neuroscience*, vol. 3, no. 7, pp. 542–552, 2002.

[18] N. Lemmens and K. Tuyls, "Stigmergic landmark foraging," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 497–504, 2009.

[19] F. Arvin, A. E. Turgut, T. Krajník, and S. Yue, "Investigation of cue-based aggregation in static and dynamic environments with a mobile robot swarm," *Adaptive Behavior*, vol. 24, no. 2, pp. 102–118, 2016.

[20] D. Lee, H. Seo, and M. W. Jung, "Neural basis of reinforcement learning and decision making," *Annual review of neuroscience*, vol. 35, pp. 287–308, 2012.

[21] T. Schmickl, R. Thenius, C. Moeslinger, G. Radspieler, S. Kernbach, M. Szymanski, and K. Crailsheim, "Get in touch: Cooperative decision making based on robot-to-robot collisions," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 1, pp. 133–155, 2009.

[22] F. Arvin, K. Samsudin, A. R. Ramli, and M. Bekravi, "Imitation of honeybee aggregation with collective behavior of swarm robots," *International Journal of Computational Intelligence Systems*, vol. 4, no. 4, pp. 739–748, 2011.

[23] F. Arvin, A. E. Turgut, F. Bazyari, K. B. Arikan, N. Bellotto, and S. Yue, "Cue-based aggregation with a mobile robot swarm: A novel fuzzy-based method," *Adaptive Behavior*, vol. 22, no. 3, pp. 189–206, 2014.

[24] M. Wahby, A. Weinhold, and H. Hamann, "Revisiting beeclust: Aggregation of swarm robots with adaptiveness to different light settings," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 272–279, 2016.

[25] M. Wahby, J. Petzold, C. Eschke, T. Schmickl, and H. Hamann, "Collective change detection: Adaptivity to dynamic swarm densities and light conditions in robot swarms," in *Artificial Life Conference Proceedings*, pp. 642–649, MIT Press, 2019.

[26] M. Bodi, R. Thenius, M. Szopek, T. Schmickl, and K. Crailsheim, "Interaction of robot swarms using the honeybee-inspired control algorithm BEECLUST," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 18, no. 1, pp. 87–100, 2012.

[27] D. Kengyel, H. Hamann, P. Zahadat, G. Radspieler, F. Wotawa, and T. Schmickl, "Potential of heterogeneity in collective behaviors: A case study on heterogeneous swarms," in *International Conference on Principles and Practice of Multi-Agent Systems*, pp. 201–217, Springer, 2015.

[28] S. Garnier, M. Combe, C. Jost, and G. Theraulaz, "Do ants need to estimate the

geometrical properties of trail bifurcations to find an efficient route? a swarm robotics test bed," *PLoS Computational Biology*, vol. 9, no. 3, 2013.

[29] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone robotics," *Autonomous Robots*, vol. 11, no. 3, pp. 319–324, 2001.

[30] R. Mayet, J. Roberz, T. Schmickl, and K. Crailsheim, "Antbots: A feasible visual emulation of pheromone trails for swarm robots," in *International conference on swarm intelligence*, pp. 84–94, Springer, 2010.

[31] F. Arvin, A. E. Turgut, T. Krajník, S. Rahimi, I. E. Okay, S. Yue, S. Watson, and B. Lennox, "ΦClust: Pheromone-Based Aggregation for Robotic Swarms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4288–4294, IEEE, 2018.

[32] F. Arvin, T. Krajník, A. E. Turgut, and S. Yue, "COSΦ: artificial pheromone system for robotic swarms research," in *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 407–412, IEEE, 2015.

[33] S. Na, Y. Qiu, A. E. Turgut, J. Ulrich, T. Krajník, S. Yue, B. Lennox, and F. Arvin, "Bio-inspired artificial pheromone system for swarm robotics applications," *Adaptive Behavior*, 2020.

[34] S. Alers, B. Ranjbar-Sahraei, S. May, K. Tuyls, and G. Weiss, "Evaluation of an experimental framework for exploiting vision in swarm robotics," in *Artificial Life Conference Proceedings 13*, pp. 775–782, MIT Press, 2013.

[35] M. Schaarschmidt, A. Kuhnle, B. Ellis, K. Fricke, F. Gessert, and E. Yoneki, "Lift: Reinforcement learning in computer systems by learning from demonstrations," *arXiv preprint arXiv:1808.07903*, 2018.

[36] K. Dalamagkidis, D. Kolokotsa, K. Kalaitzakis, and G. S. Stavrakakis, "Reinforcement learning for energy conservation and comfort in buildings," *Building and environment*, vol. 42, no. 7, pp. 2686–2698, 2007.

[37] A. Charpentier, R. Elie, and C. Remlinger, "Reinforcement learning in economics and finance," *arXiv preprint arXiv:2003.10014*, 2020.

[38] C. Yu, J. Liu, and S. Nemati, "Reinforcement learning in healthcare: A survey," *arXiv preprint arXiv:1908.08796*, 2019.

[39] L. Buşoniu, R. Babuška, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 38, no. 2, pp. 156–172, 2008.

[40] S. Na, H. Niu, B. Lennox, and F. Arvin, "Universal artificial pheromone framework with deep reinforcement learning for robotic systems," in *2021 6th International Conference on Control and Robotics Engineering (ICCRE)*, 2021.

[41] X.-Y. Liu, Z. Ding, S. Borst, and A. Walid, "Deep reinforcement learning for intelligent transportation systems," *arXiv preprint arXiv:1812.00979*, 2018.

[42] M. Hüttenrauch, A. Šošić, and G. Neumann, "Guided deep reinforcement learning for swarm systems," *arXiv preprint arXiv:1709.06011*, 2017.

[43] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14413–14423, 2020.

[44] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[45] G. Chalkiadakis and C. Boutilier, "Bayesian reinforcement learning for coalition formation under uncertainty," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pp. 1090–1097, 2004.

[46] M. Tokic, "Adaptive ε-greedy exploration in reinforcement learning based on value differences," in *Annual Conference on Artificial Intelligence*, pp. 203–210, Springer, 2010.

[47] A. dos Santos Mignon and R. L. d. A. da Rocha, "An adaptive implementation of ε-greedy in reinforcement learning," *Procedia Computer Science*, vol. 109, pp. 1146–1151, 2017.

[48] M. Gimelfarb, S. Sanner, and C.-G. Lee, "ε-bmc: A bayesian ensemble approach to epsilon-greedy exploration in model-free reinforcement learning," *arXiv preprint arXiv:2007.00869*, 2020.

[49] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, IEEE, 2017.

[50] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[51] Á. Gutiérrez, A. Campo, F. Santos, F. Monasterio-Huelin, and M. Dorigo, "Social odometry: Imitation based odometry in collective robotics," *International Journal of Advanced Robotic Systems*, vol. 6, no. 2, pp. 129–136, 2009.

[52] C. J. C. H. Watkins, *Learning from delayed rewards*. PhD thesis, King's College, Oxford, 1989.

[53] D. L. Baggio, S. Emami, D. M. Escriva, K. Ievgen, N. Mahmood, J. Saragih, and R. Shilkrot, *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing, Limited, 2012.

[54] O. Michel, "Cyberbotics Ltd. webots™: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[55] S. Alers, B. Ranjbar-Sahraei, S. May, K. Tuyls, and G. Weiss, "Evaluation of an Experimental Framework for Exploiting Vision in Swarm Robotics," pp. 775–782, 2013.

[56] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[57] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[58] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[59] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and vision Computing*, vol. 76, pp. 38–47, 2018.

[60] A. Sadeghi Amjadi, M. Raoufi, and A. E. Turgut, "A self-adaptive landmark-based aggregation method for robot swarms," *Adaptive Behavior*, p. 1059712320985543, 2021.

[61] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.