BeeGround – An Open-Source Simulation Platform for Large-Scale Swarm Robotics Applications

Sean Lim[1,2], Shiyi Wang[1], Barry Lennox[1] and Farshad Arvin[1]

[1]*Swarm & Computational Intelligence Lab (SwaCIL)*
Department of Electrical and Electronic Engineering, The University of Manchester, Manchester, UK
[2] *Faculty of Engineering, Department of Bioengineering, Imperial College London, UK*
sean.lim19@imperial.ac.uk, {shiyi.wang, barry.lennox, farshad.arvin}@manchester.ac.uk

2021 7th International Conference on Automation, Robotics and Applications (ICARA) | 978-1-6654-0469-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICARA51699.2021.9376494

***Abstract*—This paper presents an open-source simulation platform developed for implementation of both homogeneous and heterogeneous robotic swarm scenarios. *BeeGround* is a fully modular simulation software that allows for a variety of experimental setups with different robotic platforms and population sizes. Users are able to define environmental conditions, e.g. size, various properties like temperature and humidity, and obstacles arrangements. The swarm controller, the individual's behaviour, is defined with a separate programming script. In this paper, we simulated honeybees aggregation mechanism as a case study to investigate the feasibility of the developed simulation platform. The results demonstrated that the developed platform is a reliable simulation software for implementing multi-agent and swarm robotics scenarios with very large population sizes, e.g. 1000 robots.**

***Index Terms*—Swarm Robotics, Multi-agent Systems, Simulation**

I. Introduction

Multi-robot systems began with groups consisting of individual robots each designed to complete specific aspects of a complex task. The idea was to break down the task into more manageable pieces, increasing efficiency and effectiveness. The problem was that, with such specific purposes built into the programming and hardware of each robot, the structure failed when a single individual broke down which then left a part of the task unfinished. To overcome this flaw, we turned to nature for solutions [1]. Insect colonies were prime examples of multi-agent systems capable of completing incredibly complex tasks in an adaptive manner (i.e. the collective does not fail until a vast majority of individuals fail). Social insects, such as bees, ants, termites, wasps, etc., are simple on an individual scale and are divided into generic roles. For example, termites have three main castes: worker, soldier, and reproductive. In the worker caste, individuals have no way of knowing what the final mound will look like, there is no universal agreement. Yet, the building continues even as workers die and they are able to complete massive structures akin to an insect metropolis. Other examples are seen in schools, flocks, and herds that move as if they are one large organism. Naturally flowing, avoiding predators and obstacles alike, never completely breaking formation or falling into total disarray. The thing that all these organisms have in common in their behaviours is that they follow a predefined shared set of rules [2]. These rules are simple enough that an individual can execute them effectively, but are carefully crafted (over millions of years of evolution) to achieve a robust desired (or emergent) behaviour. A famous example of these rules in action comes in the form of an artificial life program, Boids, by Reynolds [3] who proposed that flocks of birds followed 3 simple rules to achieve their emergent flying behaviour: *separation*, *alignment*, and *cohesion*. Separation ensured that the birds kept a set distance away from each other to avoid collision. Alignment meant that a bird will face the same general direction as all its nearest neighbours. Cohesion meant that a bird attempted to remain in the geographical centre of the cluster of its nearest neighbours. In keeping to these rules, Reynolds created life-like simulations of artificial birds flying in a flock.

To observe and develop for emergent behaviours, researchers had to take to simulations as building thousands of real-world robots would be a colossal undertaking. Over the years, numerous platforms have been developed to cater to such needs.

A. Simulation Platforms

Table I presents a list of software commonly used for robotic platforms and swarm robotics. We described some of the widely used simulation platforms in this section.

Stage [4]: one of the most popular options of the time was Stage, a standalone 2D multi-robot simulator. Commonly used as a plugin for Player, Stage struck a balance between efficiency and accuracy. By modelling only first-order motions, Stage could reduce the computational overhead, which in turn allowed it to simulate a large number of robots within a reasonable time frame (approximately 1000 robots in real-time). Many studies in swarm robotics and multi-robotics used Stage as their simulation platform.

Gazebo [5]: designed as an extension to the Player/Stage framework, Gazebo was introduced to provide highly detailed 3D environments along with rigid body dynamics. The trade-off for greater modelling of the environment, Gazebo severely limited the number of robots that could operate at any given time, roughly to the order of tens. This drawback meant that

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) projects RAIN (EP/R026084/1) and RNE (EP/P01366X/1).

Gazebo could never fully replace Stage in large-scale multi-robot simulations.

ARGoS [6]: like other swarm simulators, was designed to simulate a large number of robots. It is capable of operating in both 2D and 3D at high levels of efficiency. In this case, 10,000 e-pucks can be simulated in a 2D environment in 60% of the time it would take for a similar real-world experiment. The novelty of ARGoS comes in its ability to partition experiments into multiple sub-spaces, each overseen by a physics engine of the user's choice. Its multi-threaded architecture also allows for efficient usage of multi-core CPUs.

Webots [7]: developed by Cyberbotics Ltd.,the initially commercial Webots program provided high-fidelity robotic simulations. Similar to problems faced by other simulators, the more accurate modelling meant that the scale of swarm experiments was limited. The price barrier also made adoption of Webots less popular amongst researchers. However, as of version 2019a, Webots became fully open-source resulting in greater accessibility.

Kilombo [8]: Kilombo is a simulator designed for the low-cost swarm robot, Kilobot. Capable of simulating 100,000 kilobots in real-time, the Kilombo simulator is a must-have for any researcher utilising Kilobots in their research. However, Kilombo was designed specifically for the Kilobot, making it limited in its applications for other robotic platforms.

USARSim [9]: originally designed for Urban Search and Rescue Simulation, the Unreal Engine-based simulator evolved into a generic multi-robot simulator. The reason for its mention is that, like BeeGround, it utilises a video game engine as the base of its architecture. However, unlike BeeGround, its focus is not on large-scale swarm simulation but on small-scale multi-robot experimentation.

B. Developed Simulator in This Work

BeeGround is the developed open-source simulation platform built upon the Unity Development Engine. The goal of BeeGround was to make swarm simulation accessible for everyone. In taking advantage of Unity's physics engine and design interface, we have created a plug-and-play package that enables users to quickly create testing environments of varying sizes, obstacle placements, and swarm populations as well as program desired swarm behaviours. The freedom of design in Unity and, by extension, BeeGround allows for customisation based on the needs of the experiment. Sensors can be added or removed, means of locomotion can be altered, or the agent swapped out entirely. In addition, Unity allows for TensorFlow 2.0 integration, further expanding the use cases of BeeGround.

Throughout the remainder of this paper, we outline how BeeGround was used to test a bio-inspired swarm aggregation algorithm and showcase its capabilities and usefulness in swarm simulation.

II. BeeGround Architecture

BeeGround was designed for convenience and quick test runs. Fig. 1 reveals architecture of the developed simulation

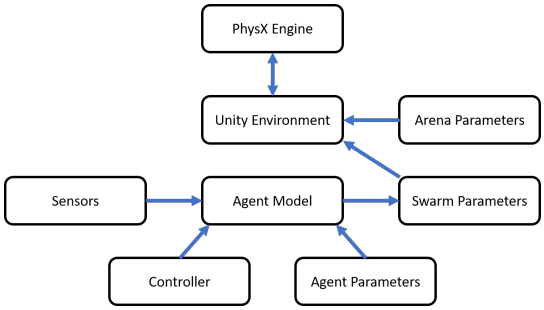


Fig. 1. Architecture of the developed BeeGround simulation software for swarm and multi-robot applications.

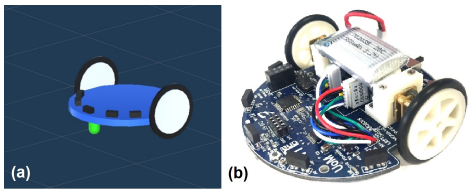


Fig. 2. (a) An abstract model of Mona with 5 infrared proximity sensors and two actuators, and (b) a Mona robot.

software and its modules. Before commencing the simulation, a user can customise the arena and swarm parameters. In arena settings, the size and the obstacle presence can be modified to cater to a variety of scenarios. For instance, in case of a cue-based bio-inspired algorithm, an additional heat map can be loaded which the agents use to reference temperature conditions within a region. The swarm parameters allow the user to specify the population size, the positions of agents within the arena, and various other kinematic constraints. In addition, ROS-Unity integration packages allow for the use of popular ROS projects in Unity with the option to publish and subscribe to topics between the two platforms. A URDF importer package also allows for existing robot definitions to be realised in the Unity Environment. In this section, we will not delve into the finer details of programming in Unity as the development engine itself has a large library of learning resources and an active community to aid in understanding. Instead, we present an overview of the components necessary in getting started with and operating BeeGround.

A. Robot Modelling

Unity Engine scripts consist of two main components that together form the basis of the robot's software: a start function and an update function. The start function emulates the initialisation phase of the robot and occurs at the beginning of the simulation. The update function behaves as a `while` loop and will continue to iterate throughout the simulation. Within this update function, the robot's behaviour is defined.

For hardware, one could define a robot by importing a standard URDF file or, alternatively, construct one from within the Unity Environment itself for quick prototyping or visualisation. For robot construction within Unity, rigid body components are added to robots which places them under

TABLE I.
LIST OF SIMULATION PLATFORMS COMMONLY USED FOR SWARM ROBOTICS RESEARCH

Platform	Ref.	OS	Level of Sim.†	Open-source	Application	No. of Robots‡	2D/3D	ROS Support
ARGoS	[6]	Linux/Mac	Realistic	Yes	Swarm	Limited	2D & 3D	Yes
Gazebo	[5]	Linux/Mac/Win	Realistic	Yes	Generic	Limited	3D	Yes
NetLogo	[10]	Linux/Win	Abstract	Yes	Generic	Large	2D	No
OpenHRP	[11]	Linux/Win	Realistic	Yes	Generic	Limited	3D	Yes
PyCX	[12]	Linux/Win	Abstract	Yes	Generic	Large	2D	No
SCRIMMAGE	[13]	Linux/Mac	Realistic	Yes	Generic	Large	3D	Yes
Stage	[4]	Linux/Mac	Realistic	Yes	Swarm	Large	2D	No
Swarm-sim	[14]	Linux/Mac/Win	Abstract	Yes	Swarm	Large	2D & 3D	No
USARSim	[9]	Linux/Mac/Win	Realistic	Yes	Generic	Limited	2D & 3D	Yes
V-rep	[15]	Linux/Mac/Win	Realistic	No	Generic	Limited	3D	Yes
Webots	[7]	Linux/Mac/Win	Realistic	Yes	Generic	Limited	3D	Yes
BeeGround	–	Linux/Mac/Win	Realistic	Yes	Swarm	1000+	2D & 3D	Yes

†*Level of Sim.:* indicates is a robot has physical properties or just a particle. ‡'Limited' means less than 50-100 robots and 'Large' means 100s of robots.

the influence of the physics engine. With this, physics such as forces, torques, and collisions can be applied within the engine's fixed update function (which occurs every 20 ms). Joints can also be implemented which allows for the creation of wheels and robotic limbs. Within the Unity framework, sensors such as cameras and range finders can be made with tools readily available in the development engine. For faster simulation, at the cost of accuracy, the physics can be ignored entirely, relying on mathematically-derived translations.

B. Arena Configuration

In the arena settings, the first step would be to define the size of the arena in standard units. BeeGround will then create a walled off arena to the desired dimensions. After which, the user can input an occupancy grid which will instantiate 1 cubic unit cube obstacles within the environment. In addition, other assets exist in Unity which allow for more complex obstacles to be crafted if so desired.

C. Swarm Parameters

The Bee agent, a simplified model of the MONA robot [16], was easily created from the ground up with customisable parameters (Fig. 2). Custom robot models can take the place of the Bee agent during swarm generation. BeeGround also allows for the swarm population and placement to be defined, allowing for some truly unique testing scenarios.

D. Simulation Parameters

For convenience in running simulations, the length of a simulation and the number of repeated trials can be defined in this section. The speed of the simulation can also be altered as it is running. This setting provides great flexibility for running long-term experiments for many repetition that is required for statistically analysing the results.

E. Bio-inspired-specific Parameters

Environmental properties, e.g. humidity, temperature, etc., play an important role in bio-inspired swarm robotics scenarios [17]. Therefore, those environmental properties must be embodied in the simulator to achieve a realistic simulation platform for swarm robotics applications. As an example, a heat map was included for the testing of the honeybees aggregation algorithm in [18]. Here, an array of temperatures was applied to the arena which the bee agents used as reference for their wait times. We can use these arrays in multiple layers to impose environmental properties with different sources and models. Also, we are able to define dynamic models of the environmental conditions that changes over time during an experiment. Furthermore, these environmental properties can interact with the robots, e.g. to implement a bio-inspired pheromone communication system [19]. The mechanics of the aggregation scenario will be further explained in the case study below. For logging in our experiments, the position of the Bee agents are recorded every second as we are looking to observe aggregation over time. However, output logs can be crafted based on the user's needs as other parameters such as velocities and rotations are made available through the Unity interface. Fig. 3 shows a snapshot from the developed simulation platform running a swarm scenario with 100 robots.

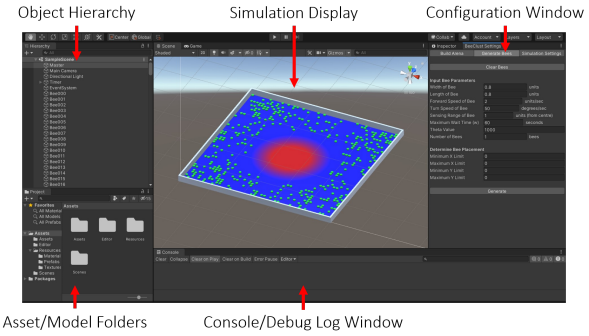


Fig. 3. BeeGround UI featuring arena instantiated with heat map with a large size swarm. *Object Hierarchy*: Overview of all objects in the environment. *Simulation Display*: Graphical representation of simulation. *Configuration Window*: Settings for BeeGround simulation including agent and swarm parameters. *Asset/Model Folders*: Library of all assets relevant to the simulation. *Console/Debug Log Window*: Tracks any errors or debugging information during simulation.

III. CASE STUDY: HONEYBEES AGGREGATION

To test the developed simulation software in its ability to implement a real robotic swarm scenario, we prepared a set of experiments on bio-inspired aggregation. In an investigation

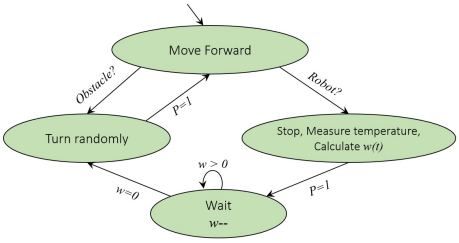
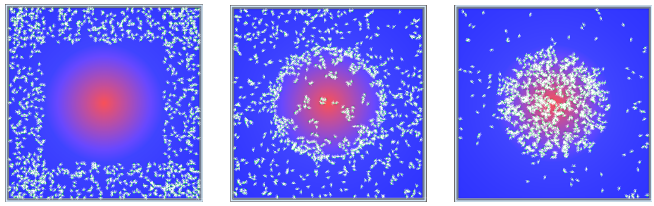


Fig. 4. Flowchart showing implemented honeybees thermotactic aggregation.



(a) t = 0 s (b) t = 600 s (c) t = 1800 s

Fig. 5. A sample simulation in BeeGround with swarm population of $N = 1000$ robots in a single cue environment setting.

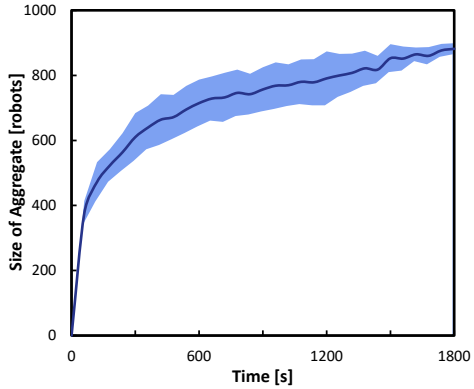


Fig. 6. Aggregation size in experiments with $N = 1000$ robots in a single cue environment. Line indicates median of the results and Blue area shows 1st and 3rd quartiles of the results.

of swarm intelligence amongst honeybees, it was observed that honeybees would form a cluster near to or at an optimal temperature, while clusters outside of this region would gradually dissolve. Schmickl et. al [20] concluded through their experiments that there were two main factors being optimised in this aggregation behaviour which led to near optimal solutions being found: temperature and social contact. Using these results, a swarm algorithm, BEECLUST, was proposed.

A. Bio-inspired Aggregation

BEECLUST, which is the algorithm adopted in this study, mimics the thermal closing behavior of young bees that cluster in an optimal area. This is an example of cue-based aggregation behaviour. The optimal area (cue) being held at temperatures ranging from 34°C to 38°C. Previous studies have shown that the BEECLUST algorithm successfully imitates the aggregation behaviour of bees in the optimal region [20], [21]. In most cases, the cue is simulated with a light source or sound source, rather than a heat source, as the implementation is simpler.

Fig. 4 illustrates the BEECLUST honeybee aggregation behaviour. BEECLUST, in and of itself, is a straightforward algorithm that the simplest of robots can execute, making it low-cost and scalable. In addition, the algorithm is crafted such that it is robust against sensors with high margins of error. The workflow of a swarm agent running BEECLUST is as follows:

1) Random exploration: Move forward until an object is detected by the proximity sensors.
2) If the object is an obstacle, turn towards a random heading and return to step 1.
3) If the object is a fellow agent, wait for a time, $w(t)$, based on the equation [17]:

$$w(t) = \frac{60S(t)^2}{S(t)^2 + 5000} \tag{1}$$

where $S(t)$ is the intensity of the cue at that particular point in the arena measured by the agent's sensor.
4) After the wait time has elapsed, turn towards a random heading and return to step 1.

B. Experiments & Results

BeeGround was used to simulate BEECLUST with a very large swarm population. N =1000 robots played the role of honeybees in a homogeneous setting meaning all the robots follow a single algorithm and physical setting. A large size arena 80×80 unit^2 was developed with a cue at the centre of the area with diameter of $d_c = 50$ unit. Each unit is equivalent to the diameter of a Mona robot. All robots moved at a speed of $v_r = 2$ unit/s. Fig. 5 shows three stages of an experiment using BeeGround. The snapshots were taken at $t \in \{0, 600, 1800\}$ s from a randomly selected experiment. Fig. 5(c) demonstrates a stable aggregation where the cue is covered by the majority of the swarm.

The experiments were repeated 5 times and observed results are shown in Fig. 6. Results showed that the swarm behaved as we expected and could find the optimal zone following the BEECLUST algorithm.

The results from aggregation in BeeGround validated the reported results of many previous studies which have used real robots [17] and different simulation platforms e.g. Stage [22], NetLogo [23], and PyCX [24].

It has significantly improved the efficiency of swarm robotics experiments with reducing complexity so we easily start a bio-inspired swarm experiment in a few simple steps on a fast 3D platform.

IV. Future Expansion

A recent addition to the Unity development engine is its integration with Tensorflow 2.0, an end-to-end machine learning platform. This machine learning package by Unity is known as ML-Agents and focuses on deep reinforcement

learning [25]. Since swarm algorithms are, in essence, optimisation problems, a swarm-focused reinforcement learning pipeline would be a powerful extension to BeeGround. Our future work will aim to explore the capabilities of reinforcement learning in the development of more efficient and robust swarm algorithms. One reinforcement learning solution of particular interest is generative adversarial imitation learning (GAIL) [26]. In GAIL, agents learn through imitation of observed behaviours. For example in investigating social insects, it would be possible to have artificial swarm agents learn from the behaviours of actual insects.

Another important improvement to the platform, which we are currently finalising, is to enable parallel simulation of several experiments. This allows us to run a swarm scenario in several settings, e.g. different populations, environmental settings, etc, simultaneously. It means that we can import some results from other settings which are in progress, or export an optimised parameter from this setting to other parallel sessions [27]. These parallel sessions are happening in nature, hence, having such a simulation platform that imitates a natural ecosystem will be of significant benefit to the swarm robotics community.

V. Conclusion

We introduced our new open-source simulation platform which has been developed based on the Unity3D engine. The simulation platform and its architecture were described in brief and a case study was carried out to evaluate the feasibility of the platform. We simulated honeybees aggregation behaviour using 1000 robots in a large size arena. It was shown that many layers of environmental properties from walls/obstacles to humidity and temperature can be modelled and simulated easily in BeeGround. Also, BeeGround supports dynamic environments where robots can interact with their environment directly, suitable for pheromone depositing and following behaviours. With BeeGround as a foundation, researchers can utilise the available tools to generate novel environments to study insect swarming behaviour or to test their own swarm algorithms.

References

[1] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, E. Bonabeau, and G. Theraula, *Self-organization in biological systems*. Princeton university press, 2003.

[2] M. Schranz, G. A. Di Caro, T. Schmickl, W. Elmenreich, F. Arvin, A. Şekercioğlu, and M. Sende, "Swarm intelligence and cyber-physical systems: Concepts, challenges and future trends," *Swarm and Evolutionary Computation*, vol. 60, p. 100762, 2021.

[3] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[4] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[5] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[6] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle *et al.*, "Argos: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[7] O. Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[8] F. Jansson, M. Hartley, M. Hinsch, I. Slavkov, N. Carranza, T. S. Olsson, R. M. Dries, J. H. Grönqvist, A. F. Marée, J. Sharpe *et al.*, "Kilombo: a kilobot simulator to enable effective research in swarm robotics," *arXiv preprint arXiv:1511.04285*, 2015.

[9] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "Usarsim: a robot simulator for research and education," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 1400–1405.

[10] S. Tisue and U. Wilensky, "Netlogo: A simple environment for modeling complexity," in *International conference on complex systems*, vol. 21. Boston, MA, 2004, pp. 16–21.

[11] F. Kanehiro, H. Hirukawa, and S. Kajita, "Openhrp: Open architecture humanoid robotics platform," *The International Journal of Robotics Research*, vol. 23, no. 2, pp. 155–165, 2004.

[12] H. Sayama, "Pycx: a python-based simulation code repository for complex systems education," *Complex Adaptive Systems Modeling*, vol. 1, no. 1, pp. 1–10, 2013.

[13] K. DeMarco, E. Squires, M. Day, and C. Pippin, "Simulating collaborative robots in a massive multi-agent game environment (scrimmage)," in *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 283–297.

[14] A. R. Cheraghi, K. Actun, S. Shahzad, and K. Graffi, "Swarm-sim: A 2d & 3d simulation core for swarm agents," in *2020 3rd International Conference on Intelligent Robotic and Control Engineering (IRCE)*. IEEE, 2020, pp. 1–10.

[15] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1321–1326.

[16] F. Arvin, J. Espinosa, B. Bird, A. West, S. Watson, and B. Lennox, "Mona: an affordable open-source mobile robot for education and research," *Journal of Intelligent & Robotic Systems*, vol. 94, no. 3-4, pp. 761–775, 2019.

[17] 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.

[18] S. Wang, A. E. Turgut, T. Schmickl, B. Lennox, and F. Arvin, "Investigation of cue-based aggregation behaviour in complex environments," *EAI International Conference on Collaborative Computing*, 2020.

[19] 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*, p. 1059712320918936, 2020.

[20] 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.

[21] 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.

[22] S. Ramroop, F. Arvin, S. Watson, J. Carrasco-Gomez, and B. Lennox, "A bio-inspired aggregation with robot swarm using real and simulated mobile robots," in *Annual conference towards autonomous robotic systems*. Springer, 2018, pp. 317–329.

[23] 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.

[24] 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4288–4294.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[26] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016, pp. 4565–4573.

[27] F. Bonnet, R. Mills, M. Szopek, S. Schönwetter-Fuchs, J. Halloy, S. Bogdan, L. Correia, F. Mondada, and T. Schmickl, "Robots mediating interactions between animals for interspecies collective behaviors," *Science Robotics*, vol. 4, no. 28, 2019.