Prototyping Sensors and Actuators for Robot Swarms in Mixed Reality

Alex Murphy[1] and Alan G. Millard[2(✉)]

[1] School of Engineering, Computing and Mathematics, University of Plymouth, Plymouth, UK
alex.murphy@students.plymouth.ac.uk
[2] Lincoln Centre for Autonomous Systems, University of Lincoln, Lincoln, UK
amillard@lincoln.ac.uk

Abstract. Swarm robotics is an approach to the coordination of large numbers of relatively simple robots with limited physical capabilities. Extending the capabilities of these robots usually requires either purchasing or prototyping new sensors or actuators (a costly and time-consuming process), often before even knowing whether these new sensors or actuators will be able to perform the necessary task effectively. This paper presents a software platform that enables researchers to prototype new sensors and actuators in the virtual world, eliminating the time and resources necessary to prototype physical hardware, and allows for experiments to be run in as many configurations as required in order to determine the efficacy of the proposed sensor/actuator.

Keywords: Swarm robotics · Virtual sensors/Actuators · Mixed reality

1 Introduction and Related Work

Swarm robotics is an approach to the coordination of large numbers of relatively simple robots with limited physical capabilities, typically designed with the space restrictions of lab experimentation in mind. Extending the capabilities of these robots usually requires either purchasing or prototyping new sensors or actuators (a costly and time-consuming process), often before even knowing whether these new sensors or actuators will be able to perform the necessary task effectively. Through necessity (i.e. due to financial/time restrictions), the swarm robotics research community has begun to tackle this problem through the use of mixed reality techniques to create virtual sensors and actuators for individual robots.

Mixed reality refers to a combination of physical and virtual objects acting as one combined version of reality [6]. This is not to be confused with augmented reality [2], which often involves overlaying virtual objects/information onto a view of a physical system. Mixed reality systems for robot swarms typically use a motion capture system in combination with bespoke software to transmit virtualised sensor information to individual robots. O'Dowd et al. [15] and Pitonakova

A. Mohammad et al. (Eds.): TAROS 2020, LNAI 12228, pp. 377–386, 2020.
https://doi.org/10.1007/978-3-030-63486-5_39

[17] have previously implemented virtual sensors for a swarm of e-puck robots [11] performing foraging tasks, using a Vicon tracking system, an external server, and Wi-Fi communication. Similarly, Turgut et al. [21] implemented a virtual heading sensor for the Kobot swarm platform with digital compasses and wireless communication, allowing robots to sense the headings their neighbours while executing collective flocking behaviours. Virtual sensors and actuators have also been implemented for Kilobot swarms through the use of external hardware such as ARK [18] and Kilogrid [1,22].

Reina et al. [19] took a different approach to creating an 'augmented reality' for a swarm of e-puck robots through the virtualisation of their sensor inputs. This was achieved by integrating an overhead camera tracking system with the ARGoS robot simulator [16] to create a mixed reality that was relayed to individual robots wirelessly. Mapping the physical world into a modified version of the ARGoS robot simulator allowed Reina et al. [19] to implement novel sensors and simulate environmental features, such as virtual pollution, which could then be passed to the robots to influence their behaviour. One experimental scenario involved simulating a polluted area and implementing a virtual pollutant sensor. The use of virtual sensors in this manner allows researchers to experiment with real robots in an environment that cannot be replicated easily in the physical world, such as hazardous areas, or to allow for the prototyping of new hardware for the robots. While not part of their research, Reina et al. [19] note that implementation of virtual actuators, such as virtual pheromones, could be done in a similar way, to provide similar experimental benefits for researchers.

Sun et al. [20] present a novel method of virtual actuation in the form of laying virtual pheromone trails behind robots to facilitate indirect inter-robot communication via stigmergy. Their ColCOSφ system integrates an overhead camera and LCD display (in place of an arena floor) with the Colias microrobot [7], which is able to detect pheromone trails rendered on the dynamic floor with colour sensors mounted to its underside. This mixed-reality approach overcomes the logistical issues associated with implementing 'pheromone' deposition in hardware, thus facilitating research that would otherwise be impractical.

Brutschy et al. [5] take a slightly different approach by abstracting actuation rather than virtualising it. Their Task Abstraction Modules (TAMs) can be placed in the environment at locations where the swarm is required to complete tasks, and individual e-puck robots can drive into them to simulate performing an action. This allows the experimenter to focus on their research question without getting caught up in the implementation of physical actuation, and is less costly than upgrading the hardware capabilities of an entire swarm.

Although virtual sensors/actuators have previously been implemented for robot swarms, there has been little investigation into the effect of different virtual sensors on swarm performance. In this paper we present the results of our experiments that vary the field-of-view of a swarm's virtual sensors, and analyse performance with respect to a simple foraging task. We also detail our mixed reality software platform that enables researchers to experiment with virtual sensors and actuators on a physical swarm of robots. This allows new sensors/actuators

Fig. 1. View of 5 e-puck robots from the overhead camera, with their virtual sensor ranges visualised in green. ArUco tags in the corners delineate the bounding box within which virtual food items are generated. (Color figure online)

to be rapidly prototyped, and their efficacy assessed, before investing time and resources in developing physical hardware. The rest of this paper describes our research methodology and experimental design, followed by a discussion and evaluation of the experimental results.

2 Methodology

In this section we describe our experimental setup, the software infrastructure implemented to carry out the experiments, and the case study foraging task.

2.1 Experimental Setup

The experimental environment comprised a small arena (600 × 840 mm) that was free of physical obstacles, as shown in Fig. 1. A Logitech BRIO[1] webcam was mounted in the ceiling above this arena, providing a tracking server with a clear real-time view of the swarm below. The server featured an eight-core Intel i7 CPU and 32 GB of RAM, running Ubuntu 16.04. This server handled all

[1] https://www.logitech.com/en-gb/product/brio.

(a) e-puck robot with Linux extension board and ArUco marker.

(b) e-puck robot detected using ArUco markers and `SwarmTracking` module.

Fig. 2. Augmented e-puck robots used for the experimental work.

sensor and actuator virtualisation, using data obtained by the overhead camera to track the location of each robot.

We used e-puck robots with a Linux extension board [8], communicating over the serial port using the advanced sercom protocol[2]. Each robot has a unique ArUco [12] marker on the top of it, as shown in Fig. 2, to enable the camera and server to track each robot individually, and virtualise the sensor and actuator data accurately. Each robot has a static IP address which can be mapped to the unique code from the ArUco tag, allowing the system to transmit individualised data to each robot.

2.2 Software Infrastructure

The software was developed as two separate applications: a tracking system, and a virtualisation system. The tracking system is a standalone swarm robot tracking system developed for this research, which uses the overhead camera to track and identify the robots using their unique ArUco tags. The virtualisation system uses the data from the tracking system in order to simulate sensor data for virtual sensors, and then send this data to the respective robot. Both pieces of software were developed in Python 2.7 using OpenCV 4.0 [4].

`SwarmTracking` [13] locates the ArUco markers on top of the robots in the camera feed, and calculates their positions in the world relative to the camera, then passes this data back to an application in a thread-safe manner. It is capable

[2] https://www.gctronic.com/doc/index.php/Advanced_sercom_protocol.

of identifying any ArUco tag that fits within the dictionary of tags specified by the user (either predefined, or custom-generated). ArUco markers provide a simple way of uniquely identifying each robot, as they encode an ID value that can be associated with a particular robot, enabling the system to appropriately track each member of the swarm. The algorithm is also built-in to OpenCV, which means it can be used with any visible-light webcam of sufficient resolution.

Figure 2b shows an e-puck robot being detected by the `SwarmTracking` module using an ArUco marker. Once detected, an outline of the ArUco tag is placed over the camera image, as well as an arrow showing the direction of movement of the robot. Direction of movement is calculated by finding the front point of the ArUco tag, and rotating this point around the centre by the angle that the tag is offset on the robot, specified by the user.

`SwarmVirtualisation` [14] is the software that virtualises sensors and actuators for swarm robots using data from a overhead camera tracking system. The software uses the `SwarmTracking` module to gather its tracking data, as well as connection with the overhead camera. As a result, the `SwarmTracking` module also passes back the frame it worked with when it calculates the position of the robots, so that the frame can be used for virtualisation and for visual feedback to users of the software. `SwarmVirtualisation` supports three distinct types of virtual objects: sensors, actuators, and environment objects. Each of these object types has sub-types that dictate the way that each individual object behaves, and the corresponding data that is generated for each camera frame. These objects are created by the users of the system, and can be written to file to ensure that the users do not have to re-create the objects each time they run the system.

Virtual sensors have three subtypes: circular, conical, and linear. These subtypes represent the different sensing methods that the system is capable of:

Circular: Defined by a range – implements a circular sensing field with the radius of the circle being the range specified, and the centre point being the centre of the robot. The sensor will detect any robots or virtual objects within the bounds of the sensing field, and returns a Boolean *True* when it detects something.

Conical: Defined by a range, field-of-view, and angular offset. The sensor implements a cone shaped sensing area originating at the centre of the robot with a length determined by the range, optionally offset by a specified angle.

Linear: Implements a line-of-sight sensor originating from the centre of the bot, parameterised by its range and width. It also takes an angular offset, calculated and used in the same manner as that of the conical sensor. This sensor will detect any robot or environment object within range and in the line of sight of this sensor, returning a Boolean *True* when it detects something.

There are two sub-types of virtual actuator: placers and grabbers. As with virtual sensors, these sub-types dictate the behaviour and attributes of the actuator.

Grabber: Picks up virtual environment objects when the robot is on top of one, provided that the robot has sufficient inventory space. The grabber picks up an item when the centre of the robot is over an environment object.

Placer: Places virtual environment objects into the world at specified intervals from the robot.

Environment objects have three types: goal, obstacle and item. These subtypes dictate how the robots interact with them:

- **Goal:** Object representing a target for the robots.
- **Obstacle:** An object to be avoided by the robots.
- **Item:** An item that can be interacted with by the robots.

Virtualisation of the sensor and actuator data is separated out into a different thread so as not to impact either the user interface, or the data collection from the `SwarmTracking` module. Each time a new frame is gathered by the `SwarmTracking` module, the simulation thread attempts to calculate whether each sensor on each bot is able to detect an object. Having this run in a separate thread enables the application to continue tracking the robots and collecting data without having to wait for these calculations to be completed.

Communication between the software and the robots is achieved by transmitting JSON[3] packets from the server to the robot. These packets contain the virtualised sensor data, transmitted to the appropriate robot only, to enable the robot to behave as if it had collected the data from an on-board sensor. The networking is handled in a separate thread to ensure that the virtualisation and data collection are not interrupted in the event of network issues.

2.3 Case Study: Foraging

This mixed reality software was used to implement virtual sensors and actuators for a swarm of 5 e-puck robots, which then carried out a foraging task – a typical task for robot swarms [3]. In our particular implementation, each robot moves around the arena at random until it detects a virtual food item, at which point it will move towards it and collect it with its virtual actuator, and then return to randomly searching for more food items. When a virtual food item is collected, another is randomly generated within the arena.

The food items for these experiments were randomly generated within the bounding box of the arena, calculated during a calibration stage when the software starts. One experiment was run for each different sensor using the same random seed, after which the random seed was changed and another experiment was run for each different type of sensor. This process was repeated until each sensor had been tested ten times. Five different virtual sensors were used for the experiments with different fields-of-view ranging from 45° to 360° (as shown in Fig. 3), as well as a control experiment with no sensors at all. The range of each sensor remained consistent throughout all experiments.

[3] https://www.json.org/json-en.html.

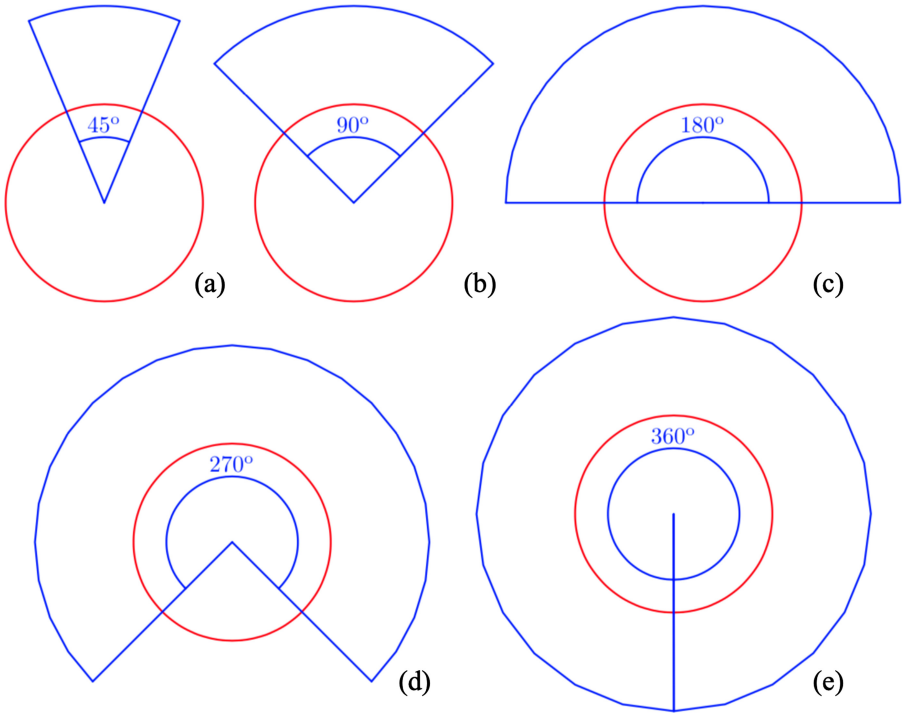

Fig. 3. Virtual sensors used in the experiments. Sensors (a), (b), (c) and (d) are conical sensors, with angles of 45°, 90°, 180° and 270° respectively, while (e) is a circular sensor (equivalent to a 360° conical sensor).

3 Experimental Results

A series of experiments were carried out to assess the efficacy of different virtual fields-of-view for the robot swarm collecting virtual food items. The number of food items collected within 10 min was used to analyse the efficacy of the virtual sensors. 10 repeat runs were completed for each of the 6 different conditions to assess the performance accounting for stochastic behaviour, representing 60 experiments in total spanning 10 h.

Figure 4 shows how effective each sensor was with respect to the total number of food items collected, compared with a control experiment where no virtual sensors were used. Most of the sensors tested show a relatively small amount of variation in the number of food items collected between each experiment, implying that the virtual sensors produce consistent results.

An unexpected result from these experiments is that the use of a 45° conical sensor performed worse than having no sensors at all in almost all cases, implying that the use of this sensor would actually hinder the performance of robots. This is because the 45° conical sensor, and to a lesser extent the 90° conical sensor, had a tendency to overshoot the angle when turning towards the food object.

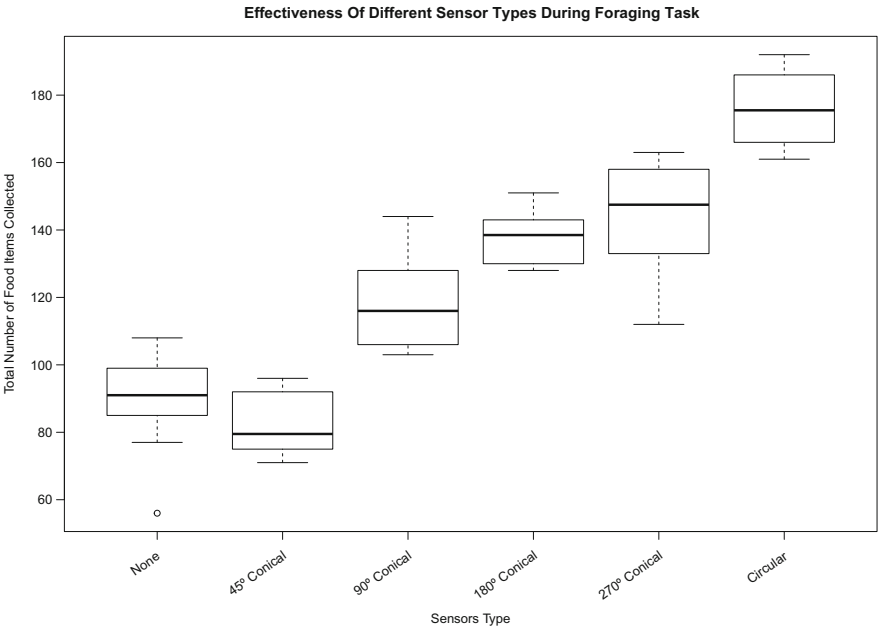

Fig. 4. Performance of each sensor type with respect to the foraging task. Each boxplot represents 10 repeat runs of each condition. The performance metric for each run is the total number of food items collected by the swarm in 10 min.

When the robot overshoots the angle, it does so by turning further than it should have done, often ending up with the food object it was attempting to turn towards now being outside of the range of the sensor. This issue mainly affects sensors with a smaller field-of-view as the larger ones are able to more effectively account for this level of error. All other sensors, outperformed the control experiment, which is to be expected, with the average performance of the sensors improving as the angle of the cone increases.

The results of these experiments show that the use of virtual sensors and actuators can successfully increase the capabilities of swarm robots. In particular, we have shown that the sensors and actuators implemented enabled a swarm of e-puck robots to consistently detect and navigate to virtual objects in real-time.

4 Conclusions and Future Work

The results of these experiments clearly show that the use of virtual sensors and actuators can successfully extend the capabilities of swarm robotics platforms. Through the use of the software developed for this project, a swarm of e-puck robots was able to successfully carry out a foraging task by identifying virtual food objects in real time, navigating to these food objects and collecting them, using a range of different virtual sensor types.

The software developed for this project is relatively limited in its scope, as the virtual sensors/actuators can only interact with virtual objects. Further work on this could include the ability to perform object/edge detection within an image, allowing the software to simulate sensors that can interact with the physical world. Extending this functionality to include detecting physical objects as well would provide a much wider range of possible uses/applications. Modelling the effect of noise would also allows for more realistic implementations of sensor hardware, as physical sensors inevitably have some level of inaccuracy, and is often significant enough to have a noticeable effect on robot performance.

In future, we intend to integrate the mixed reality functionality presented in this paper into ARDebug [10] – an augmented reality debugging system for robot swarms, and to extend the work to support the Pi-puck [9] – a Raspberry Pi extension board for the e-puck robot that is a modern replacement for the Linux extension board used for this research.

References

1. Antoun, A., et al.: Kilogrid: a modular virtualization environment for the Kilobot robot. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3809–3814. IEEE (2016)
2. Azuma, R.T.: A survey of augmented reality. Presence Teleoperators Virtual Environ. **6**(4), 355–385 (1997)
3. Bayındır, L.: A review of swarm robotics tasks. Neurocomputing **172**, 292–321 (2016)
4. Bradski, G., Kaehler, A.: OpenCV. Dr. Dobb's J. Softw. Tools **3**, (2000)
5. Brutschy, A., et al.: The tam: abstracting complex tasks in swarm robotics research. Swarm Intell. **9**(1), 1–22 (2015)
6. Hönig, W., Milanes, C., Scaria, L., Phan, T., Bolas, M., Ayanian, N.: Mixed reality for robotics. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5382–5387. IEEE (2015)
7. Hu, Cheng., Fu, Qinbing, Yue, Shigang: Colias IV: the affordable micro robot platform with bio-inspired vision. In: Giuliani, Manuel, Assaf, Tareq, Giannaccini, Maria Elena (eds.) TAROS 2018. LNCS (LNAI), vol. 10965, pp. 197–208. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96728-8_17
8. Liu, W., Winfield, A.F.: Open-hardware e-puck Linux extension board for experimental swarm robotics research. Microprocess. Microsyst. **35**(1), 60–67 (2011)
9. Millard, A.G., et al.: The Pi-puck extension board: a Raspberry Pi interface for the e-puck robot platform. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 741–748. IEEE (2017)
10. Millard, A.G., Redpath, R., Jewers, A.M., Arndt, C., Joyce, R., Hilder, J.A., McDaid, L.J., Halliday, D.M.: ARDebug: an augmented reality tool for analysing and debugging swarm robotic systems. Frontiers Robot. AI **5**, 87 (2018)
11. Mondada, F., et al.: The e-puck, a robot designed for education in engineering. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, vol. 1, pp. 59–65. IPCB: Instituto Politécnico de Castelo Branco (2009)
12. Munoz-Salinas, R.: ArUco: a minimal library for Augmented Reality applications based on OpenCV. Universidad de Córdoba (2012)

13. Murphy, A.: GitHub - SwarmTracking (2019). https://github.com/amurphy4/SwarmTracking
14. Murphy, A.: GitHub - SwarmVirtualisation (2019). https://github.com/amurphy4/SwarmVirtualisation
15. O'Dowd, P.J., Winfield, A.F.T., Studley, M.: The distributed co-evolution of an embodied simulator and controller for swarm robot behaviours. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4995–5000. IEEE (2011)
16. Pinciroli, C., et al.: ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intell. **6**(4), 271–295 (2012). https://doi.org/10.1007/s11721-012-0072-5
17. Pitonakova, L., Winfield, A., Crowder, R.: Recruitment Near Worksites Facilitates Robustness of Foraging E-Puck Swarms to Global Positioning Noise. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4276–4281. IEEE (2018)
18. Reina, A., Cope, A.J., Nikolaidis, E., Marshall, J.A., Sabo, C.: ARK: Augmented Reality for Kilobots. IEEE Robot. Autom. Lett. **2**(3), 1755–1761 (2017)
19. Reina, A., et al.: Augmented reality for robots: virtual sensing technology applied to a swarm of e-pucks. In: 2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 1–6. IEEE (2015)
20. Sun, X., Liu, T., Hu, C., Fu, Q., Yue, S.: ColCOSφ: a multiple pheromone communication system for swarm robotics and social insects research. In: 2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM), pp. 59–66. IEEE (2019)
21. Turgut, A.E., Çelikkanat, H., Gökçe, F., Şahin, E.: Self-organized flocking in mobile robot swarms. Swarm Intell. **2**(2–4), 97–120 (2008)
22. Valentini, G., et al.: Kilogrid: a novel experimental environment for the kilobot robot. Swarm Intell. **12**(3), 245–266 (2018)