ARK: Augmented Reality for Kilobots

Andreagiovanni Reina[@], Alex J. Cope[§], Eleftherios Nikolaidis[‡], James A.R. Marshall[@] and Chelsea Sabo[§]

***Abstract*—Working with large swarms of robots has challenges in calibration, sensing, tracking, and control due to the associated scalability and time requirements. Kilobots solve this through their ease of maintenance and programming, and are widely used in several research laboratories worldwide where their low cost enables large-scale swarms studies. However, the small, inexpensive nature of the Kilobots limits their range of capabilities as they are only equipped with a single sensor. In some studies, this limitation can be a source of motivation and inspiration, while in others it is an impediment. As such, we designed, implemented, and tested a novel system to communicate personalised location-and-state-based information to each robot, and receive information on each robots' state. In this way, the Kilobots can sense additional information from a virtual environment in real-time; for example, a value on a gradient, a direction towards a reference point or a pheromone trail. The Augmented Reality for Kilobots (`ARK`) system implements this in flexible base control software which allows users to define varying virtual environments within a single experiment using integrated overhead tracking and control. We showcase the different functionalities of the system through three demos involving hundreds of Kilobots. The `ARK` provides Kilobots with additional and unique capabilities through an open-source tool which can be implemented with inexpensive, off-the-shelf hardware.**

***Index Terms*—Swarms, Multi-Robot Systems, Virtual Reality and Interfaces**

I. INTRODUCTION

SINCE their development, Kilobot robots [1] have been employed in numerous swarm robotics studies, e.g., [2], [3], [4], [5], [6], [7], [8], and are the main robotic platform of various research projects, such as DiODe [9], Swarm-Organ [10], E-Swarm [11], and FloraRobotica [12]. The success of this platform is ascribed to its low cost, which allows users to perform experiments with a large number of robots at a reasonable price compared to alternative platforms, and its scaleable charging and programming features. However due to their simplicity, Kilobots are equipped with a minimal set of sensors and actuators which significantly limits the range of actions they can perform.

Manuscript received: February, 15, 2017; Accepted April, 10, 2017. This paper was recommended for publication by N. Y. Chong upon evaluation of the Associate Editor and Reviewers' comments. This work was funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 647704).

This paper has a supplementary downloadable video (41 MB) available at http://ieeexplore.ieee.org, provided by the authors. The supplementary video showcases the `ARK`'s functionalities through three demos.

The authors are with the Department of Computer Science, University of Sheffield, S1 4DP, UK ([@]email: a.reina@sheffield.ac.uk; james.marshall@sheffield.ac.uk).

[§]Contributed equally to this work.

[‡]E. Nikolaidis is also with the Alexander Technological Educational Institute of Thessaloniki, Greece.

In this paper, we present *Augmented Reality for Kilobots* (`ARK`), a system that extends the Kilobot capabilities by allowing the robot to operate in augmented reality. This system allows Kilobots access to customised information based on their location and state. They can, in turn, modify their virtual environment which can then be sensed by other robots. Through the proposed system, Kilobots can be employed in experiments that utilise a set of sensors/actuators richer than those the standard robot has (e.g., see Demo C in Section IV-C). Additionally, `ARK` makes it considerably easier and reduces the time required to operate large swarms by automating several necessary steps in setting up an experiment; e.g. positioning, motor calibration, and unique ID assignment (showcased through Demos A and B in Section IV). Finally, `ARK` provides functionalities to log and record experimental data for subsequent analysis. `ARK` is composed of three components: (i) a overhead camera tracking system that provides real-time data on robot location and state, (ii) a modified overhead emitter which broadcasts infrared (IR) signals to communicate to the Kilobots, and (iii) a base control station to coordinate the system and simulate the virtual environments. The system architecture is described in more detail in Section III. While an alternative technology for Kilobot augmented reality has been recently proposed [13], the `ARK` system differentiates itself by offering additional functionalities at a substantially lower cost which have been demonstrated in scaled robotic experiments (discussed further in Section II).

The functionalities of `ARK` are showcased through three demos presented in Section IV. In Demos A and B, we show the possibility of employing the system for automatic unique ID assignment and automatic positioning of robots at the beginning of an experiment. This is needed for automatic calibration—a function (enabled with `ARK`, but not included in this paper) that is highly desirable for experiments involving many, imprecise robotic platforms. These operations are typically tedious and time consuming when done manually. Additionally, automating the operation gives more accurate control of the robots' start positions and removes undesired biases in comparative experiments. In Demo C, we show a simple foraging scenario where 50 robots collect material from a source location and deposit it at a destination. Through this demo, we show that robots can perceive (and navigate) a virtual gradient, can modify the virtual environment by moving material from one location to another, and can autonomously decide when to change the virtual environment that they sense (either the source or the destination).

II. RELATED WORK

In various studies, the limited capabilities of the Kilobot required users to complement their experiments with the virtualisation of environmental features. This has been done through the use of additional, dedicated robots (not involved in the collective behaviour) that broadcasted constant messages about a target area [3], [5], [8] or a reference coordinate system [2]. This strategy needs to be tailored for each experimental setup, has its own limitations, and may not always be a viable solution.

As such, a more generic solution has been recently proposed in the form of the Kilogrid [13] which is an electronic device that is placed under a glass surface on which the Kilobots move. The Kilogrid, through the glass, can receive IR messages from the robots and send localised information to them. Similar to ARK, the Kilogrid allows robots to operate in augmented reality and can be used to collect experimental data (e.g. robot locations). Unfortunately, the Kilogrid is costly and time-consuming to reconstruct as it is made from bespoke components. Kilogrid is composed of cell modules with an estimated cost of one Kilobot (which has a market price of ∼£85). For a large-scale arena ($2\times2\,\mathrm{m}^2$ in size), 400 modules are necessary which would cost tens of thousands of pounds. Since the main characteristic of the Kilobots is their low cost, we present a system in agreement with this idea. Further differences consist in the communication frequency, which in the case of Kilogrid is higher, and in the use of discrete vs continuous space. While Kilogrid operates on discrete cell modules of $5 \times 5\,\mathrm{cm}^2$, the proposed ARK system tracks robots in a continuous space. This can be beneficial for precision tasks, such as motor calibration or robot placement (see Demo B in Section IV-B).

Other works have proposed systems for augmented reality for robots that present a structure very similar to ARK [14], [15], [16], [17], [18], [19], [20]. In all these works, the robots' locations are tracked through overhead cameras connected to a base control station which then delivers virtual environment information to the robots. In most works, the system is designed for a specific use case (rather than as a general purpose tool); for example, to generate virtual pheromone trails that are either projected on the ground through an overhead projector [14], [16] or displayed on an LCD screen which acts as the robots' ground [21]. Alternatively in [17], [19], robots can sense any type of virtual environment. However, the robots used in these studies are equipped with unique markers for recognition and wireless modules for direct individual communication. Unfortunately, unique markers may hide the Kilobots' LED and need to be very small and dense (due to the number and size of the robots) which hinders real-time image processing. Additionally, Kilobots do not allow for directed communication, and so they rely only on broadcast messages.

III. ARCHITECTURE

As described, the ARK is comprised of three main components which allow for a flexible and scalable augmented reality environment for Kilobots: (A) a Base Control Software (BCS), (B) communication mainly through a modified Overhead Controller (OHC), and (C) tracking via several overhead cameras. This complete system allows for the control of various Kilobot swarm sizes for both centralised and decentralised experiments. The BCS provides an open-source tool[1] by which to manage and edit all components of the system. This system architecture is shown in Figure 1.

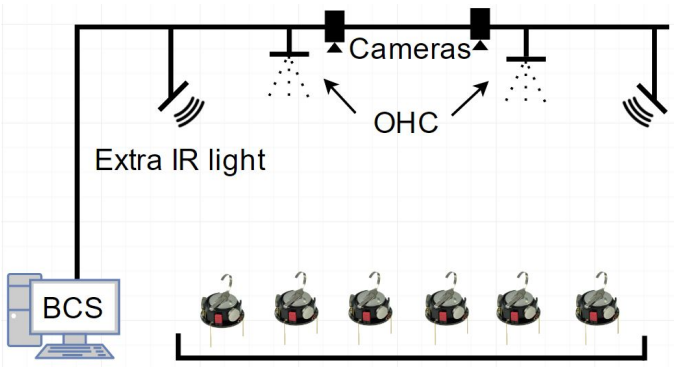

Fig. 1. ARK arena hardware architecture.

While the BCS is comprised mainly of software which can be implemented on most machines with a NVIDIA GPU, the communication and tracking require off-the-shelf, inexpensive hardware. These hardware decisions are described in further detail here along with the different stages of the processing pathway in BCS for each component.

A. Base Control Software Structure

The BCS provides a way to organise and administer the tracking, communication, virtual environments, and Kilobots themselves. Firstly, the software must obtain camera images from the four cameras covering the Kilobot arena. The system needs to correct for distortion in the images and stitch them together to form a single image of the entire arena. Secondly, the software must allow the Kilobots to be individually identified

[1]The source code is open-source and available on GitHub at https://github.com/DiODeProject/KilobotArena and the full documentation can be found at http://diode.group.shef.ac.uk/kilobots/index.php/ARK

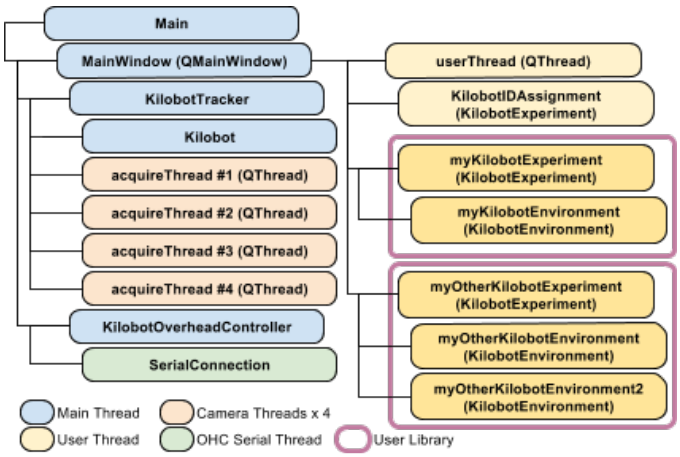

Fig. 2. BCS overview of class structure.

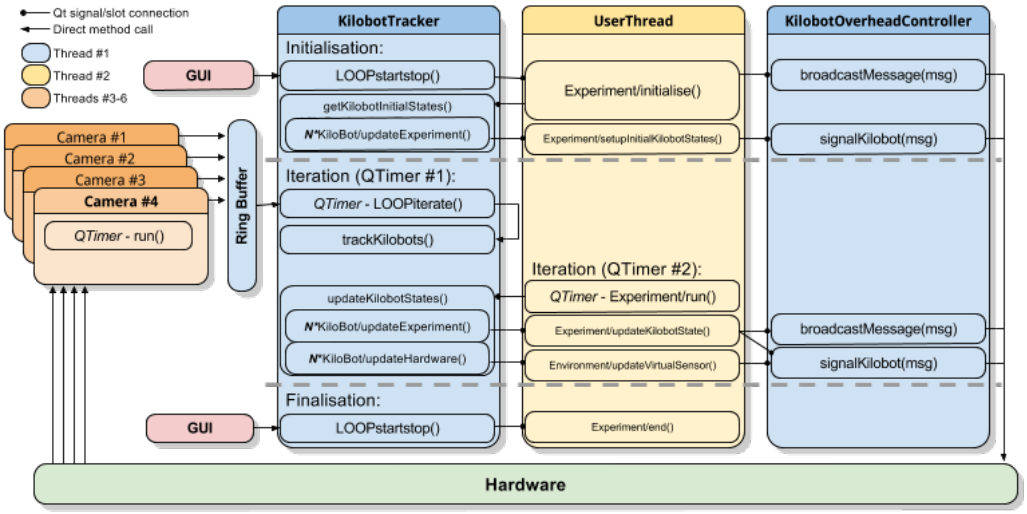

Fig. 3. BCS life cycle of an experiment.

and tracked, including both their position and LED colour. Thirdly, the software must allow the user to address unique messages to each Kilobot based on their individual state (position, orientation and LED colour), as well as broadcast identical messages to all Kilobots. Fourthly, the BCS must present a Graphical User Interface (GUI) to provide users a means to configure the system and gain feedback on progress of experiments. Finally, the BCS must execute sufficiently fast to allow for real-time operation. An overview of the BCS codebase can be seen in Figure 2 which describes the class structure of an experiment.

Two main design decisions allow real-time performance: the use of threading to allow execution across multiple CPU cores and the use of the OpenCV CUDA GPU libraries [22], [23] to perform the image processing for stitching, tracking, and LED colour identification. The GUI is created using the Qt Library Framework.

Given that the BCS supervises all the main components of the system, it is responsible for constructing and managing experiments and their virtual environments and sensors. Because Kilobots are inexpensive, simplistic robots, they are only equipped with a single sensor capable of detecting ambient light. Enabling virtual environments, and therefore virtual sensors, opens up a wide-range of feasible experiments that were not previously possible by only using Kilobot hardware. For example, the virtual sensor could act as a sort of positioning system, olfactory system capable of picking up pheromones, proximity detector, magnometer, and more. The `ARK` enables a single high-resolution virtual sensor to be implemented or multiple lower-resolution sensors. The choice is dictated by the limited number of bits available in each message (see Section III-B for further details).

`ARK` runs experiments in a separate thread to resolve issues with timing and synchronisation which arise when running real-time experiments alongside tracking and control. To facilitate simple switching of experiments, the BCS loads in experiments as external libraries (users are provided a template library that can be modified). Experiments are subclassed from kilobotExperiment where the lifecycle of an experiment is shown in Figure 3. In addition to the experiment, users can also implement a set of environments (created by subclassing kilobotEnvironment) for each experiment. Different environments allow Kilobots to respond to different virtual sensors, and Kilobots can be switched between environments throughout an experimental run as shown in Demo C. Full instructions on implementing experiments and environments can be found on our website at http://diode.group.shef.ac.uk/kilobots/index.php/ARK.

To allow experiments to run as close to real-time as possible, they are instantiated in a separate thread called userThread. As such, the interaction between the main thread and userThread is undertaken using Qt signals and slots as these can be attached in a thread safe manner. The experiment requests the locations of the Kilobots using a signal, and then each Kilobot signals back to the experiment and its environment with a copy of itself. By using a copy of the Kilobot class, the code is rendered thread-safe and the user need not worry about such complex issues. In addition, the experiment provides callbacks to set itself up and to pass back a GUI panel. This is integrated into the main GUI window and allows interactive controls to be defined by the user. Figure 4 shows the GUI interface and an example of the user GUI panel.

`ARK` automates time-consuming and mundane tasks, e.g. assigning IDs, required to operate swarms of Kilobots. Prior to tracking the Kilobots, they must first be located and then assigned IDs (either a new, unique ID or a recovered, previously assigned ID). After having located the Kilobots, we must identify their ID numbers. Recovering previous IDs is performed by sending a broadcast packet to all Kilobots containing a single ID as the payload. The Kilobot assigned that ID sets its LED colour to blue, and all other Kilobots set their LEDs to red. The tracker can then identify the correct Kilobot and note its ID before trying the next one. Once all IDs are identified, the process finishes. The identification code can be added to all Kilobot programs to avoid the need to load a separate program for identification and is designated using a

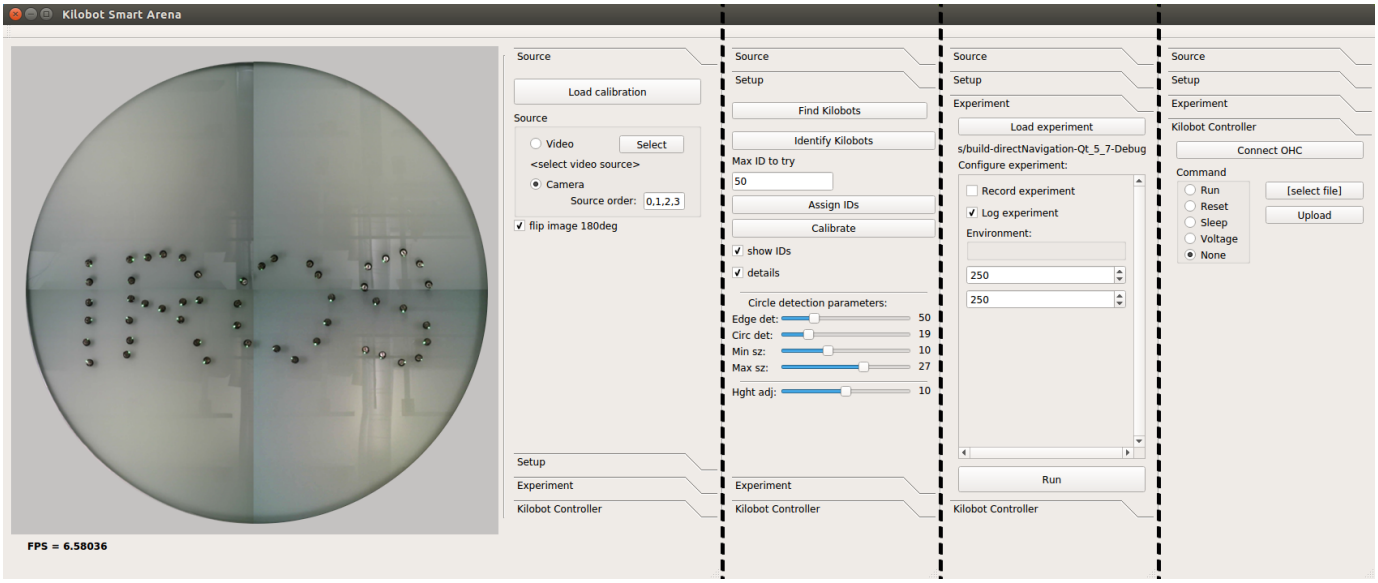

Fig. 4. BCS GUI interface. The cameras' image shows a frame of the Demos B in which ARK guides 50 robots to the desired target positions.

specific packet number type (here, we use 120). Assigning IDs requires a more complex procedure, and a specific program must be loaded onto the Kilobots to accomplish this. The ID assignment procedure is implemented as an experiment which is compiled into the main BCS program to simplify the system. The procedure is described in details in Demo A in Section IV-A.

B. Communication

Communication from the BCS to the Kilobots and from the Kilobots amongst themselves is done through Infrared (IR) signals. Kilobots are run on a smooth, very flat table which is glossy to help IR signals reflect off the surface and reach the robot light sensor. In a typical setup, the commercially-available OHCs[2] are mounted about 1 metre above the table. This results in roughly a one metre diameter region below the OHC where the signal can reach the robots. Unfortunately, the limitations of the IR LEDs on the OHC prevent the use of this system over larger areas and swarms. Further, the glossy surface and ambient room lighting can create shadows and other artefacts which degrade communication. This can also vary from point-to-point across an arena with a large surface area.

ARK presents a controller design that overcomes communication challenges under environmental uncertainty and limited coverage. The ARK OHC was designed to provide complete communication coverage of the arena despite its large size and the uncertainties in the environment. This was mainly accomplished through careful selection of IR LEDs for a re-design of a system-wide OHC and through the use of additional IR lighting [24]. The ARK OHC was designed using an Arduino pro mini, MOSFET drivers as switches, IR LEDs, and various resistors. The Arduino controls multiple (up to 6) mini-OHCs which are distributed evenly around the ARK arena. The mini-OHCs are powered by an external 12 V source that is capable of providing the required amperage. The Arduino sends the ON/OFF signal which is aided by the MOSFET drivers to ensure fast switching and high power to the LEDs. The same code[3] developed by Alex Cornejo for the OHCs can be used with the ARK OHC with only a slight modification. That is, the signal needs to be sent to the first 6 bits of PORTB on the Arduino rather than just the 2nd.

Through careful testing across various surrounding lighting conditions, surface materials, and LED parameters (wavelength, half-angle, and radiant intensity), an optimal LED was selected to use in the ARK OHC [24]. The TSAL6200 IR LED was chosen due to its performance over large distances and environmental uncertainties. The additional IR lighting source also utilizes these same IR LEDs but with a 33 Ohm resistor. The extra lighting further addresses the challenges of shadows and communication uncertainties by providing another light source directed at the arena. This allows communication even when there is no other light source in the surrounding area (e.g. the room could be pitch black) which would previously never have been possible since Kilobots IR sensors require a minimum background IR illumination for effective communication.

ARK BCS bundles messages and introduces logic that solves issues with sending messages to a large number of robots. The final stage in processing is to send information to the Kilobots. This is done by constructing packets that are sent to the OHC. Broadcast packets contain no Kilobot ID and therefore, are acted on by all Kilobots. A range of Kilobot packet types is allowed for broadcast packets (types 1 to 119) where numbers greater than 119 are reserved for system messages and 0 is reserved for signal packets. Signal packets are constructed to pass information to Kilobots with specific IDs. As signal packets are likely to be sent for a large number of Kilobots and there is a minimum duration

[2]http://www.k-team.com/mobile-robotics-products/kilobot

[3]https://github.com/acornejo/kilolib

that a packet must be sent for to ensure reliable transmission, it is necessary to bundle messages so they contain information for more than one Kilobot. As a Kilobot message contains 72bits of payload, three different Kilobots can each be sent 24bits of data in one message. These 24bits are broken down into 10bits for an ID (0→1023), four bits for a type (0→15) and 10bits for a payload (0→1023). Signal messages are constructed by delaying transmission of messages for 100 ms, and then collating all queued messages into groups of three and constructing signal packets which are then queued for transmission to the OHC at 50 ms intervals. This means that 150 Kilobots can be signalled once each in 2.5 s. During this time, a robot can move a maximum of ∼3 cm so this is suitable frequency given the slow dynamics. Even still, this limit is only hit if all robots need to receive constant updates. The communication frequency could be increased if robots only need conditional updates (e.g. conditional on location or state).

C. Tracking

Tracking of the Kilobot swarms is done through 4 overhead cameras which feed back to the BCS for processing. The number of cameras were chosen as the minimum which achieve coverage of a large arena (e.g. $2.2\times2.2\,\mathrm{m}^2$) when mounted at a specific height above the surface (here, we mounted them at 160 cm high). More cameras can be used with modifications to the ARK software. In the ARK system, we use e-CAM51_USB cameras [25] which are 71×13 mm in size, have a maximum 2592×1944 image resolution, and can achieve a maximum of 30 frames per second. However, any camera with comparable resolution and frame-rate can be used as long as the base station has the appropriate ports. Finding the Kilobots is performed using OpenCV [22], and the locations of the found Kilobots are used to instantiate Kilobot class instances.

ARK BCS parallelises processing of camera information for efficient tracking of large robotic swarms. Image acquisition from each of the four cameras covering the arena is handled in a separate worker thread. These threads use individual OpenCV CUDA streams to warp the images and resize them in preparation for stitching. This allows for concurrent execution (dependent on GPU support). The warping uses parameters determined by a separate program which performs automated calibration on the four camera images to transform them into a single view using the OpenCV panorama stitching module [22]. The final images are placed in a two-element ring buffer which is synchronised to the main stitching thread using QSemaphores. This arrangement allows the latency of acquiring images from the cameras to be hidden. Although, it must be noted that the Linux video backend is not thread-safe and only one camera can be addressed at a time. This contention is managed using a single QMutex across the threads.

The images from the camera threads are collated in the main thread and stitched together using memory copying on the GPU. Tracking can then be performed on the final full arena image. Due to the modular nature of the code, a custom tracking algorithm can easily be implemented by modifying the `trackKilobots()` method of the kilobotTracker class. However, we provide a simple and performant reference implementation using the Hough Circles algorithm (CUDA implementation) in OpenCV to locate the Kilobots. Following localization of the Kilobots, the distances between Kilobot locations can be calculated on the GPU. Then, the minimum distance for each Kilobot is found and their previous location is updated on the CPU.

IV. DEMOS

The functionality of the ARK system is showcased through three demo experiments. For all three Demos, the code for the Kilobot controller and the experiment plugins is open-source and available online[4]. At the same address and in supplementary online material of this paper (available at http://ieeexplore.ieee.org), we provide a video of the three Demos. Demos A and B display how the ARK system can be employed to automate some of the preliminary operations in typical swarm robotics experiments. Instead, Demo C shows a foraging experiment in which 50 Kilobots autonomously operate in different virtual environments with varying features.

A. Demo A

In this demo, we automatically assign a unique sequential ID to each robot $i \in N$ in a swarm of size N. The process works as follows: Each robot i selects uniformly at random a natural number $r_i \in [1, 2^d]$ which is converted to binary form. Then, ARK requests the robots to display the d binary digits one at a time through the LED colour (following the convention of red=0 and blue=1). Once ARK collects all digits, for all numbers that have been uniquely selected, it broadcasts the unique number r_i and the number i. The robot that selected r_i updates its ID to i. The process is repeated for robots that selected non unique random numbers r_i. The parameter d can be selected as a function of the swarm size N and determines the tradeoff between the speed of acquiring a binary sequence and the probability of having non unique random numbers. In the supplementary video, we show a demo with $N = 100$ robots and $d = 11$ digits in which the system is able to assign 100 IDs in a few seconds.

This process also shows how ARK can be employed to retrieve data from the kilobots (in this demo, the number r_i). Several studies may require the collection and storage of data from robots at the end of the experiment. While normally this operation is done through a wired connection, ARK allows wireless and parallel kilobot data collection.

B. Demo B

This demo displays an artistic performance of 50 robots that form words: first the word *IROS* and after the number *2017* (see Figure 4(left) and the supplementary video). This demo demonstrates how the ARK system can be used to overcome the challenge of placing robots in relatively short time, and with relatively high accuracy, a swarm of N Kilobots. The ARK system reads the N target coordinates from a text file

[4]http://diode.group.shef.ac.uk/kilobots/index.php/ARK

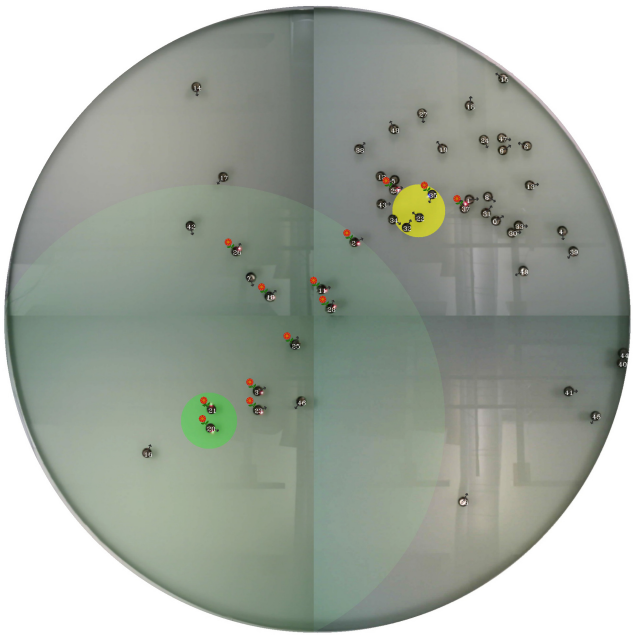

Fig. 5. A screenshot of the Demo C involving 50 foraging Kilobots. The robots sense the direction towards the source area (green circle) from a distance of 70 cm (light green shadow). The robots pick up loads from the source and carry it to the destination (yellow circle). The full video is available in the online supplementary material.

and guides each robot i towards one target location T_i through four types of motion messages: `stop`, `straight`, `left`, and `right`. Each timestep, `ARK` computes the vector v_i connecting the robot i's position to T_i. If the distance $\|v_i\| < 1$ cm, `ARK` sends to i the message `stop`; otherwise, if the angle $\theta = \angle v_i - \phi_i$ (with ϕ_i the robot's orientation), in radians, is $\theta < -0.96$ it sends `left`; if $\theta > 0.96$ it sends `right`; if $(-0.96 \leq \theta \leq 0.96)$ it sends `straight`. To minimise unnecessary communication, `ARK` sends a message only if it is different from the last sent message. The robot reacts to the four types of message as follows: if it receives `stop` interrupts its movements because it reached its destination; if it receives `straight`, `left` or `right`, it moves forward, left, or right, respectively.

C. Demo C

This demo has been designed to display a Kilobot swarm capable of sensing and acting in augmented reality. The demo is composed of varying virtual environments in order to highlight this unique feature as it offers more flexibility when designing experiments. Here, each environment has a target area: in one, it is the source area and in the other, the destination area. Source and destination areas are circular in shape with an initial diameter $d_s = 40$cm and $d_d = 5$cm, respectively. The robots can sense the direction towards the target area, and if they are within or outside the area. Robots are programmed to pick up one unit of a virtual load inside the source area, carry it to the destination, and deposit the load there. When performing actions in the virtual environments, the robot signals by lighting its LED in blue. When picking up a load from the source, the robot reduces the source's size for the rest of the robots (by reducing the area's diameter by 1 cm). Similarly when a robot deposits loads at its destination, the area increases by 1 cm.

This demo displays the possibility for robots to (i) sense a virtual environment (by directing their motion towards the target area), (ii) modify the virtual environment (by picking/depositing virtual material), and (iii) switch virtual environment (by alternating sensing of source and destination). Note that in this demo, the actuation is not passive and that the robots autonomously decide where and when to pick up or deposit the load. In this simple demo, robots switch virtual environment automatically with their pick/deposit action. The supplementary video and Figure 5 show a swarm of 50 Kilobots that retrieve nectar from a virtual flower field and deposit it in the honeycomb nest. Robots carrying a load are distinguishable by their red LED in the video.

V. CONCLUSIONS

Robotic experiments involving swarms tend to be expensive, and they present problems because the robots generally have limited sensors and actuators, the time needed to calibrate is extensive, and it is difficult to track and communicate with each individual robot. This paper presents the `ARK`, a virtualised arena, which enables inexpensive, swarm robotics research with Kilobot robots. It overcomes limitations in these low-priced platforms like limited sensing by using a virtualised environment and corresponding virtual sensors for the robots. This is enabled with a scalable overhead control system which is robust to environmental uncertainties. Additionally, the arena has an overhead tracking system that also scales well for large swarms (hundreds of Kilobots). Finally, this system is controlled by a base station where users can easily assign IDs to robots, calibrate Kilobots in parallel, setup their virtual environments, and plugin unique experiments. These methods are confirmed in various demos and show an inexpensive, open-source tool that is flexible for configuration with many, diverse experiments. `ARK` is useful both for collective behaviour studies that require sensors and actuators currently unavailable on kilobots, and for cheap prototyping to assess the impact of new sensors/actuators before hundreds of them are physically produced.

Future work involves adding even more functionalities to the `ARK` system. The example ID assignment provided is scalable but is prone to errors. Also while automatic calibration is enabled with `ARK`, this is a difficult problem by itself to solve which needs to be investigated further. Finally, we plan to implement a default procedure to collect error-free data from the robots at the end of an experiment. However, these add-ons can easily be updated and installed by users in the future.

ACKNOWLEDGEMENTS

The authors would like to acknowledge and thank Michael Port and Salah Talamali for their contributions in making this project a success.

REFERENCES

[1] M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera, and R. Nagpal, "Kilobot: A low cost robot with scalable operations designed for collective behaviors," *Robot. Auton. Syst.*, vol. 62, no. 7, pp. 966–975, 2014.

[2] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[3] G. Valentini, E. Ferrante, H. Hamann, and M. Dorigo, "Collective decision with 100 Kilobots: Speed versus accuracy in binary discrimination problems," *Auton. Agent Multi Agent Syst.*, vol. 30, no. 3, pp. 553–580, 2015.

[4] Y. K. Lopes, S. M. Trenkwalder, A. B. Leal, T. J. Dodd, and R. Groß, "Supervisory control theory applied to swarm robotics," *Swarm Intelligence*, vol. 10, no. 1, pp. 65–97, 2016.

[5] A. Reina, T. Bose, V. Trianni, and J. A. R. Marshall, "Effects of Spatiality on Value-Sensitive Decisions Made by Robot Swarms," in *Proc. Int. Symp. on Distrib. Auton. Robot. Syst. (DARS)*, ser. STAR, 2016.

[6] V. Trianni, D. De Simone, A. Reina, and A. Baronchelli, "Emergence of Consensus in a Multi-Robot Network: from Abstract Models to Empirical Validation," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 348–353, 2016.

[7] A. Becker, G. Habibi, J. Werfel, M. Rubenstein, and J. McLurkin, "Massive uniform manipulation: Controlling large populations of simple robots with a common input signal," in *Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS)*, 2013, pp. 520–527.

[8] C. Dimidov, G. Oriolo, and V. Trianni, "Random Walks in Swarm Robotics: An Experiment with Kilobots," in *Proc Int. Conf. on Swarm Intell. (ANTS)*, ser. LNCS, 2016, vol. 9882, pp. 185–196.

[9] "Distributed algorithms for optimal decision-making," http://diode.group.shef.ac.uk.

[10] "Swarm organ," http://www.swarm-organ.eu.

[11] "Engineering swarm intelligence systems," http://www.e-swarm.org.

[12] "Societies of symbiotic robot-plant bio-hybrids as social architectural artifacts," http://www.flora robotica.eu.

[13] A. Antoun, G. Valentini, E. Hocquard, B. Wiandt, V. Trianni, and M. Dorigo, "Kilogrid: a modular virtualization environment for the kilobot robot," in *Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS)*, 2016, pp. 3809–3814.

[14] K. Sugawara, T. Kazama, and T. Watanabe, "Foraging Behavior of Interacting Robots with Virtual Pheromone," in *Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS)*, 2004, pp. 3074–3079.

[15] P. J. O'Dowd, A. F. T. Winfield, and M. Studley, "The distributed co-evolution of an embodied simulator and controller for swarm robot behaviours," in *Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS)*, 2011, pp. 4995–5000.

[16] 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 Comput. Biol.*, vol. 9, no. 3, p. e1002903, 2013.

[17] A. Reina, M. Salvaro, G. Francesca, L. Garattoni, C. Pinciroli, M. Dorigo, and M. Birattari, "Augmented reality for robots: virtual sensing technology applied to a swarm of e-pucks," in *Proc. IEEE NASA/ESA Conf. Adapt. Hardware Syst. (AHS)*, 2015, p. sB_p3.

[18] F. Ghiringhelli, J. Guzzi, G. A. Di Caro, V. Caglioti, L. M. Gambardella, and A. Giusti, "Interactive Augmented Reality for understanding and analyzing multi-robot systems," in *Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS)*, 2014, pp. 1195–1201.

[19] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, "Mixed reality for robotics," in *Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS)*, 2015, pp. 5382–5387.

[20] R. Ramaithitima, M. Whitzer, S. Bhattacharya, and V. Kumar, "Automated Creation of Topological Maps in Unknown Environments Using a Swarm of Resource-Constrained Robots," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 746–753, 2016.

[21] F. Arvin, S. Yue, and C. Xiong, "Colias-Φ: An Autonomous Micro Robot for Artificial Pheromone Communication," *Int. J. Mech. Eng. and Robot. Res.*, vol. 4, no. 4, pp. 349–353, 2015.

[22] G. Bradski, "OpenCV Library," *Dr. Dobb's J. Softw. Tools*, 2000.

[23] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, pp. 40–53, 2008.

[24] E. Nikolaidis, C. Sabo, J. A. R. Marshal, and A. Reina, "Characterisation and upgrade of the communication between overhead controllers and Kilobots," White Rose Research Online, Tech. Rep., April 2017.

[25] e-con Systems India Pvt Ltd, "e-CAM51_USB Datasheet," Chennai, India, www.e-consystems.com.