# Multi-Swarm Interaction through Augmented Reality for Kilobots

Luigi Feola[1,2], Andreagiovanni Reina[3], Mohamed S. Talamali[4], and Vito Trianni[2]

***Abstract*—Research with swarm robotics systems can be complicated, time-consuming, and often expensive in terms of space and resources. The situation is even worse for studies involving multiple, possibly heterogeneous robot swarms. Augmented reality can provide an interesting solution to these problems, as demonstrated by the ARK system (Augmented Reality for Kilobots), which enhanced the experimentation possibilities with Kilobots, also relieving researchers from demanding tracking and logging activities. However, ARK is limited in mostly enabling experimentation with a single swarm. In this paper, we introduce M-ARK, a system to support studies on multi-swarm interaction. M-ARK is based on the synchronisation over a network connection of multiple ARK systems, whether real or simulated, serving a twofold purpose: (i) to study the interaction of multiple, possibly heterogeneous swarms, and (ii) to enable a gradual transition from simulation to reality. Moreover, M-ARK enables the interaction between swarms dislocated across multiple labs worldwide, encouraging scientific collaboration and advancement in multi-swarm interaction studies.**

***Index Terms*—Swarm Robotics, Multi Swarm, Heterogeneity, Kilobot, Augmented Reality**

## I. INTRODUCTION

Robot swarms are expected to revolutionise a large number of application sectors and spread into our everyday lives [1]. A *swarm* refers to a decentralised group of self-organising robots that operate in a coordinated manner without the need for a central authority or leader. The robots within a swarm can interact with each other through various means, such as by exchanging information or by physically interacting with each other through contact or proximity [2]. A homogeneous swarm consists of robots that are all identical in terms of their physical and software characteristics. This means that they all have the same capabilities and limitations, and can perform the same tasks. On the other hand, a heterogeneous swarm is made up of robots that have different physical and/or software characteristics, and can be designed to take advantage of diversity by assigning different roles to different robots based on their capabilities. This can make the swarm more efficient and robust in performing complex tasks [3].

A multi-swarm can be considered as a special case of a heterogeneous swarm. We define multi-swarm interaction (MSI) as the ability of multiple swarms of self-organising robots to interact with each other and execute increasingly complex tasks. Each swarm in a multi-swarm system may be internally homogeneous or heterogeneous, and may occupy the same or a different area in the environment. Think for instance of a swarm of flying robots interacting with a swarm of ground robots [4]. In a search and rescue scenario, the flying swarm can be deployed to quickly identify victims, while the ground swarm can bring first aid support [5]. In a multi-swarm system, coordination can emerge from the interactions between the individual robots within each swarm and the interactions between the different swarms. MSI can help achieve better overall performance, providing swarms with specialised abilities that can compensate for the limitations of other swarms, e.g., when a swarm of autonomous surface vehicles provides a swarm of underwater robots with connectivity across the water-air barrier and absolute positioning information [6]. Generally speaking, the study of the interaction among multiple robotic swarms is essential for advancing the capabilities and performance of robotic systems in various applications, from manufacturing and logistics to search and rescue and space exploration [7].

Notwithstanding the potential advantage, heterogeneous and multi-swarms are far less studied than homogeneous ones. Besides the increased complexity in designing and studying such systems, other reasons for the limited appeal are very practical ones, mostly related to increasing costs of production, experimentation and maintenance. Indeed, research with (heterogeneous) robot swarms can be overly complicated, time-consuming, and expensive in terms of space requirements, energy and labour. In particular, each research laboratory is often equipped with robot swarms that are not conceived to interact with each other, limiting the possibilities of studying heterogeneous systems.

Several tools have been developed over the years to simplify research and experimentation in swarm robotics [8], [9]. A particularly interesting approach makes use of Augmented Reality (AR) to provide controllable virtual environments for real robot swarms [10], [11], [12], [13]. In robotics,

Manuscript received: June, 9, 2023; Accepted August, 10, 2023.

This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by the Office of Naval Research Global (ONRG) through the project "Collective Decisions in Dynamic Environments" (CODE, award N62909-18-1-2093). A.R. acknowledges financial support from the Belgian F.R.S.-FNRS, of which he is Chargé de Recherches. The authors are grateful to Garry Turner for support with experiments in Sheffield.

[1]Luigi Feola is with the Department of Computer, Control, and Management Engineering Antonio Ruberti, Sapienza University of Rome, 00185 Rome, Italy (e-mail: feola@diag.uniroma1.it)

[2]Luigi Feola and Vito Trianni are with the Institute of Cognitive Sciences and Technologies (ISTC) of the National Research Council (CNR), 00185, Rome, Italy (e-mail: luigi.feola@istc.cnr.it; vito.trianni@istc.cnr.it)

[3]Andreagiovanni Reina is with IRIDIA, Université Libre de Bruxelles, 1050 Ixelles, Belgium (e-mail: andreagiovanni.reina@gmail.com); Sheffield Robotics, University of Sheffield, UK; and the Centre for the Advanced Study of Collective Behaviour, University of Konstanz, Germany

[4]Mohamed S. Talamali is with Sheffield Hallam University, UK (e-mail: s.talamali@shu.ac.uk)

Digital Object Identifier (DOI): see top of this page.

AR consists in providing robots with virtual actuators and sensors, so that they can (i) modify the virtual environment, and (ii) perceive (virtual) features in the environment beyond what their onboard sensors can do. To this end, a simulated environment is aligned with the real one, a tracking system reproduces the real robots in the virtual environment, and communication between the simulator and the robots enables the virtualisation of sensors and actuators. A very prominent example is the ARK system [10], developed to increase the experimental potential of Kilobots, a widely adopted platform in swarm robotics research [14]. Kilobots are small, inexpensive robots designed to efficiently operate within a large swarm in parallel, including powering on, charging and programming. They feature a minimalist design, being endowed with just one ambient light sensor, two vibration motors, an infrared transceiver for communication with neighbours, and a coloured LED. Set aside the morphogenesis behaviour they were designed for [15], [16], the limited capabilities of the Kilobots have constrained their experimental usage to basic behaviours, involving random walks and local communication [17], [18], [19], [20], [21]. Thanks to the ARK system, the Kilobot capabilities have been extended by superimposing a virtual environment to the physical one, hence enabling abilities such as wall avoidance, phototaxis, self-localisation, pheromone laying/sensing, and task execution [22], [23], [24], [25]. Finally, ARK allows automated logging of experimental data to run offline analysis and have a complete overview and control of the experiment. Kilobots and their ARK system can be simulated within ARGoS [26], a widely used simulator in swarm robotics research thanks to its flexibility, accuracy and efficiency [27].

To improve the ARK system and make it a suitable tool for studying MSI, we developed a multi-ARK system (M-ARK)[1]. M-ARK allows different ARK entities to be interconnected over a TCP/IP network, even if they are located in different, possibly remote places (e.g., different labs in different countries). Indeed, although ARK is designed to perform swarm experiments, it still has computational limits that can be overcome by distributing the computation over several remote machines to perform swarm and multi-swarm experiments with a potentially infinite number of robots. Thanks to augmented reality, M-ARK enables MSI by virtually placing the robots of remote ARKs inside the local ARK. This framework can facilitate collaboration between different laboratories, which own different robotic platforms, to study the emergent behaviour from the interaction of different types of swarms and reduce the costs of purchasing robots. Moreover, M-ARK makes it possible to connect real and simulated systems, by interfacing with one or multiple ARGoS instances. This is an example of mixed reality, which enables testing MSI with a very large number of robots while only part of these are deployed in a physical setting, also providing a pipeline to develop and test multi-swarm systems starting from a fully simulated to a full real-world solution.

In this work, we demonstrate how M-ARK allows to: (i) create a custom projection of the shared environment for each swarm in the multi-swarm system, (ii) render each swarm different from the others in terms of sensorimotor capabilities, and (iii) support interaction and communication across swarms. Currently, M-ARK is limited to non-physical interactions among swarms. Hence, it is relevant for MSI where the different swarms do not physically interact. That is, we consider robot swarms occupying different spaces: UAV swarms flying at different altitudes, unmanned aerial and ground vehicles, or surface and underwater robots. Also, M-ARK can easily model a resource collection task, where there may be a swarm deployed to collect objects in the environment, while another swarm do the sorting inside a depot area. Overall, the range of possible applications of M-ARK is large and relevant for advancing the study of MSI.

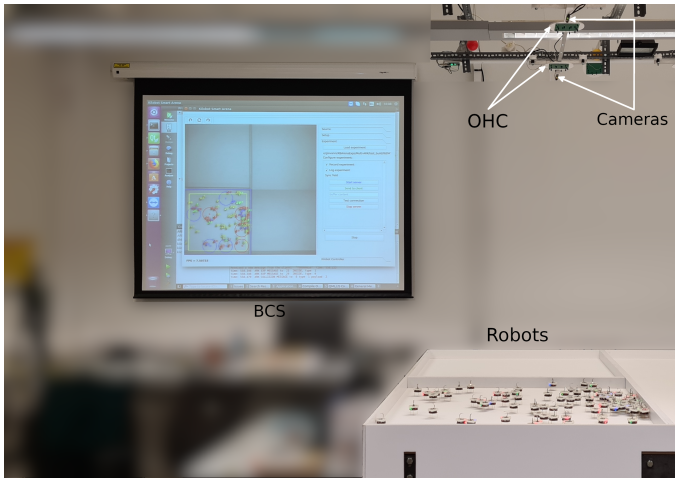


Fig. 1: Single ARK setup. Made of three main components: (i) a Base Control Station (BCS), (ii) an Overhead Controller (OHC), and (iii) a tracking system using RGB cameras.

The remainder of this paper is structured as follows. Section II describes in detail the M-ARK system analysing every single component. In Section III, the functionalities of M-ARK are showcased through two representative MSI problems: dynamic heterogeneous team formation (Section III-A) and collective resource exploitation (Section III-B). Section IV concludes the paper with discussions on the relevance of the proposed architecture for future multi-swarm robotics systems.

## II. M-ARK SYSTEM CONFIGURATION

In M-ARK, there exists a "shared environment" which is the combination of multiple "local environments", each controlled by a single ARK instance, which can be either real or simulated. As illustrated in Fig. 1, a real ARK instance is composed of three main parts: (i) an overhead camera tracking system that provides real-time data on robots' pose and state, (ii) an overhead controller (OHC) 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. A simulated ARK instance is instead a software process implemented with the ARGoS simulation APIs, which provide a means to manage the virtual environment and the communication between the simulated

[1]The M-ARK software is open source and available online at [28]

ARK control station and the robots [26]. M-ARK interconnects multiple ARK instances through TCP/IP sockets, e.g., a client-server configuration in which ARK client instances connect to the ARK server instance.

In order to start an experiment, the ARK server socket is opened on a predefined port, waiting for the ARK clients to connect. As soon as all instances are connected to each other, the experiment can begin and flows as shown in Fig. 2. The first step is the synchronisation of the environment of all ARK instances. In general, it is the server that is responsible for sending a string-type message that describes all the characteristics that the shared environment of all the ARK instances must conform to. Note that ARK instances make no difference whether they are connected to a simulated or real ARK, since the interaction between the different ARK instances occurs simply through an exchange of messages.

Once all instances are connected and synchronised, it is necessary to share how the (virtual) environment in which the robots are deployed evolves. There are two possible approaches. A time-based approach assigns to the ARK server the role of updating the ARK clients by broadcasting status messages with a fixed time period $T_p$. To this end, every client sends updated information about the local environment with a fixed period $T_c$ (usually, $T_c = T_p$ if there is no strong reason to have different updating periods). The server then combines all the information received by the ARK clients into a coherent environment state, and builds the message to synchronise all clients on the current state. The second approach is instead event-based. Update messages are exchanged between server and clients only when some relevant event occurs in the local environment, which needs to be reflected on the global environment. Note that the reception of a message from a client also represents an event that the server must consider in order to propagate relevant changes to other clients. Both methods have advantages and disadvantages. The time-based approach allows clients to finely synchronise the evolution of the shared virtual environment with the server, but may also face the problem of insufficient bandwidth and message loss. The event-based approach has a more parsimonious use of communication, but requires much more attention in the event definition and management. Also, message loss in the event-based approach must be carefully considered, because it can lead to significant deviations across local environments. A mixed solution can also be considered, in which the server follows a time-based approach while the clients follow an event-based method. This can reduce the communication load and also provide a mean to the client to verify if the events they reported to the server have been appropriately reflected in later updates from the server.

Another aspect relevant to the synchronisation of multiple ARK instances concerns the possibility of different update speeds, which can occur when simulated ARK instances are considered. When all ARK instances are constrained to evolve in real time, there is no alignment problem. However, simulated ARKs may deviate from real-time execution, which is a desirable aspect especially to speed up experiments. In such conditions, it is necessary to ensure that the time evolution of multiple ARKs is coherent. Therefore the messages exchanged between the instances can also be exploited for time synchronisation, hence constraining the time evolution to the slowest ARK process in the M-ARK system. To this end, a time-based approach is preferable.

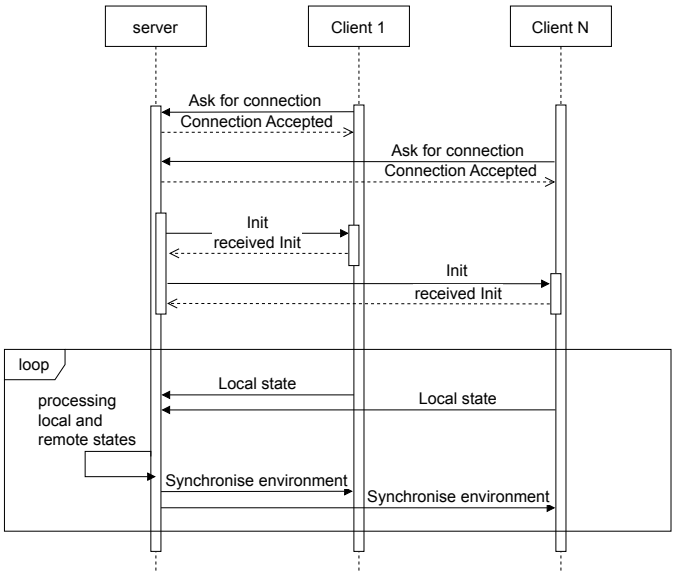


Fig. 2: Sequence diagram for M-ARK execution. First, the connection among all the ARK instances is established. Then, the server sends the *Init* message sharing with the $N$ clients the initial state of the global virtual environment. In the main loop, clients and server exchange local state and synchronisation messages to align the local environments.

In this work, we present an implementation of M-ARK following a client-server, time-based approach with $T_p = T_c = 2\,\mathrm{s}$. In Section IV, we further discuss the possible alternative implementations, also beyond a client-server approach.

## III. CASE STUDIES

### A. Dynamic Heterogeneous Team Formation (DHTF)

The first case study demonstrates the interaction between two heterogeneous swarms to solve a strictly cooperative task. The setup is inspired by the stick-pulling experiment [29], which is a representative study of team formation in swarm robotics. Effective collaborations can be achieved only when a sufficient number of robots are present in the same collaboration site. In this extension of the work done in [25], we consider the formation of heterogeneous teams, making two swarms interact through M-ARK. As in [25], collaboration sites are characterised by the complexity of the tasks they spawn, which can be easy or hard according to the requirements in terms of number of robots needed for task execution (i.e., hard tasks require more robots than easy tasks). Moreover, since we study a multi-swarm, the requirement of one collaboration site can be different across local environments, hence creating higher variability. As soon as a team composed of the correct number of robots in each of the two swarms is formed, the collaboration is established, the task is executed and the robots resume exploration in search of

other tasks. This case study demonstrates how a relevant multi-swarm problem can be easily implemented through M-ARK both in simulation and in the real world, enabling a smooth transition between the two conditions. It also shows how to create a shared environment by connecting local environments that have different features, such as the arena size or the type of tasks to be executed.

*1) Experimental Setup:* In this case study, we connect two ARK systems, both simulated and real. In the latter case, the ARK server is located at the University of Sheffield, within the Department of Computer Science (henceforth: ShARK), featuring a square experimental arena of $1\,\mathrm{m}$ side running a swarm of $N_s = 48$ robots. The ARK client is located at the Institute of Cognitive Sciences and Technologies of the Italian National Research Council in Rome (henceforth: RoARK), featuring a square arena of $0.5\,\mathrm{m}$ side running a swarm of $N_c = 12$ robots, keeping the same robot density as in the server. At the beginning of the experiment, robots are uniformly distributed in the arena. The arena is virtually divided into a $4 \times 4$ grid, and $M = 8$ collaboration sites are selected and positioned inside one of the grid cells, as shown in Fig. 3. Each collaboration site is characterised by a task type (easy or hard) that never changes. The positions of the collaboration sites within the arena is the same in both ARKs, the only differences are in: (i) the task type is randomly chosen in each ARK, i.e., a task could be easy or hard on both, or could be easy in one and hard in the other; (ii) the task's requirements (for RoARK, easy tasks require one robot, while hard tasks require two robots; for ShARK, the requirements are doubled). Once the right heterogeneous team formation is achieved within a collaboration site for both ARKs, the task is considered completed and the collaboration site becomes inactive (see the grey circles in Fig. 3), meaning that no task is spawned for $T_r = 40\,\mathrm{s}$; after this period, a task appears in the same location with the same type as before. We choose the value for $T_r$ to create a non-trivial problem, so that robots have enough time to move away from the collaboration site before it returns active.

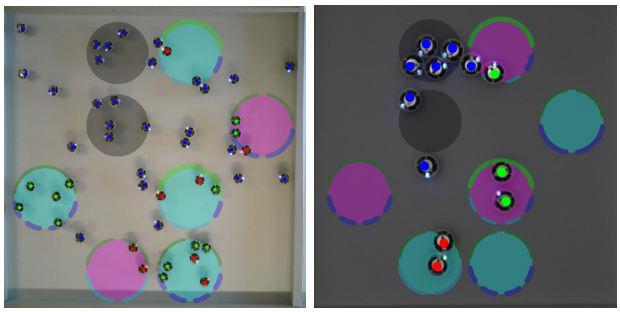

Fig. 3: DHTF experimental setup. Left: ShARK; Right: RoARK. Magenta areas represent easy tasks. Cyan areas represent hard tasks. Grey areas represent inactive collaboration sites, which become active again after $T_r\,\mathrm{s}$ from the last task execution. The arcs of circles visible over active areas show the task requirements and the current team size (light and dark blue for ShARK, light and dark green for RoARK). Colours on top of a robot illustrate its state: blue corresponds to the state Walk ($W$), green to state Dwell ($D$), and red to the state Leave ($L$). See also the supplementary video.

*2) Robot Behaviour:* The robot behaviour can be described by a finite state machine (FSM) with three states: Walk ($W$), Dwell ($D$), and Leave ($L$). Robots start in state $W$, in search of tasks to be executed, and move according to a random walk [17]. When a robot enters an active collaboration site, it switches from state $W$ to state $D$, in which it waits $T_D$ seconds motionless for other robots to join. If at some point the task requirements are satisfied on each ARK, the task gets immediately executed, and the robot switches back to state $W$. Otherwise, when the internal timeout expires, the robot switches to state $L$, in which it moves away from the collaboration site ignoring the task. When outside the collaboration area, the robot switches to state $W$ [25]. Both robot swarms have the same behaviour.

We assume that a robot detects the task type when it enters a collaboration site. Given that the task difficulty can be different for the two swarms, there are three possible combinations: (i) if the task is hard on both sides, the dwelling robot triggers a timer $T_D = 60\,\mathrm{s}$; (ii) if the task is hard on one side and easy on the other, the robot triggers a timer of $T_D = 40\,\mathrm{s}$; last, (iii) if the task is easy on both sides the robot triggers a timer $T_D = 30\,\mathrm{s}$. In this way, robots wait longer in response to different requirements for the shared environment, as the task completion depends on coordination among the two swarms.

*3) M-ARK Communication Protocol:* To enable interaction between the two swarms, the local ARKs exchange messages related to the state of the collaboration sites. Fig. 4 provides a simplified sequence diagram representing the interaction between client and server. Once the connection among the two M-ARK entities is established, ShARK (the server) sends to RoARK (the client) an *Init* message, which contains a list of the active collaboration sites, specifying the type and the requirements for both sides. This resulting message consists of $3M+1$ characters, starting with the message type identifier (*I*), followed by $M$ characters corresponding to the ID of the active sites, which are integer values. The arena is divided in $4 \times 4$ cells, and $M$ IDs are randomly selected from the set $\{1, 16\}$ by ShARK. The last $2M$ characters indicate the corresponding type, with a value of $0$ and $1$ representing respectively easy and hard tasks for both ShARK and the RoARK. In this way, robots from the two swarms can virtually sense the difficulty of the overall task. Moreover, thanks to the *Init* message the environments are synchronised to the same initial state. During the experiment execution, the two entities exchange periodic update messages every $T_p = T_c = 2\,\mathrm{s}$. RoARK sends a message to ShARK containing information about the tasks that are *Ready*, i.e., the ones that meet the requirements locally. The message is a string of $M + 1$ characters, with the first character (*R*) indicating the message type and the remaining $M$ characters representing $1$ if the corresponding task is ready and $0$ otherwise. ShARK processes this information and sends an *Accomplished* message to RoARK when the task requirements are met globally. This message is also a string of $M + 1$ characters, with the message type being indicated by the character $A$ and the remaining $M$ characters indicating whether each task has been accomplished (1) or not (0). Once

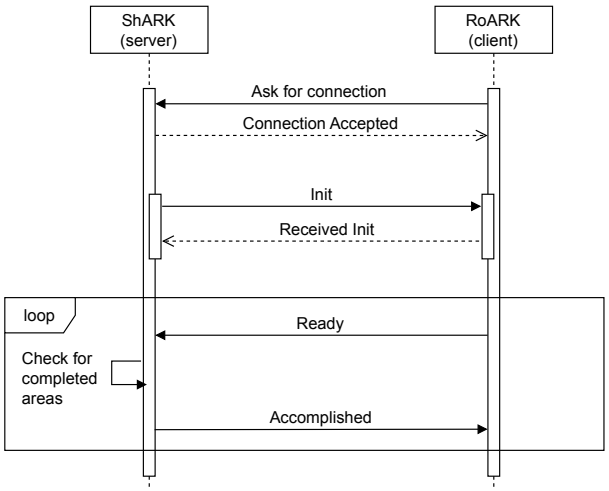


Fig. 4: DHTF sequence diagram. In the main loop, every $T_c = 2\,\mathrm{s}$ the client sends to the server its local state, specifying in a *Ready* message the tasks where requirements are locally met. The server, considering the client's and its own state, sends every $T_s = 2\,\mathrm{s}$ a message with the shared environment state to the client, specifying in a *Accomplished* message the list of tasks that are globally completed.

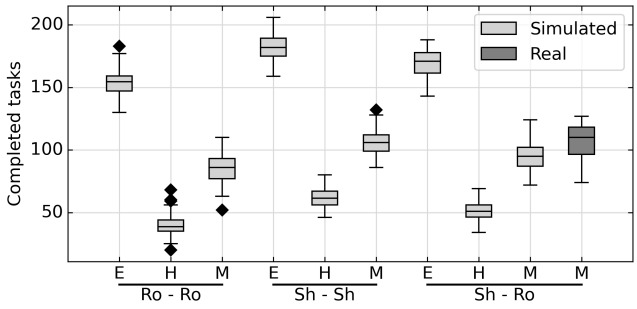


Fig. 5: DHTF results. We consider different setups: (i) RoARK-RoARK (Ro – Ro), with both environments feature a small arena and a small swarm; (ii) ShARK-ShARK (Sh – Sh), with both environments featuring a large arena and a large swarm; (iii) ShARK-RoARK (Sh – Ro), featuring mixed small and large arenas/swarms. In each condition, we consider tasks that are all easy (E), all hard (H) or randomly mixed (M). Real robots are tested in the Sh – Ro setup.

a task is accomplished, the corresponding collaboration site becomes inactive for both ShARK and RoARK. When a new task is spawned on ShARK, RoARK is notified, and the collaboration site becomes active again.

*4) Results:* We performed a large evaluation of the behaviour both in simulation and with a real M-ARK system connecting ShARK and RoARK. In simulation, we vary the size of the arenas, studying a shared environment that is either asymmetric with a small and a large arena, or symmetric with both arenas being small or large. We also vary the task difficulty with three setups: (i) all easy, (ii) all hard, (iii) randomly mixed, as shown in Fig. 3. We evaluate the performance as the number of completed tasks. In simulation, we perform 100 independent runs, while in the physical setting we perform 10 independent runs ($1800\,\mathrm{s}$ for each run). Results are shown in Fig. 5.

As expected, we notice a similar trend in the three different scenarios, with easy-easy tasks being executed in larger quantity, while hard-hard tasks are those that place the higher demands. For the mixed case, we notice that the number of completed tasks is in between of the previous two cases. Larger swarms accomplish more tasks than smaller ones, despite the wider arena. This can be justified by the fact that, while the arena and swarm size scales quadratically with the arena side, we imposed that the task requirements scale linearly instead. This choice was dictated by the observation that robots mostly stop at the border of a collaboration site, impeding entrance of other robots. Hence, the carrying capacity of a collaboration site scales linearly (with the perimeter) and not quadratically (with the area). This intuition is confirmed by the fact that, despite the swarm density remains constant across different arena sizes, the task completion performance only slightly increases. Finally, we consider the ShARK-RoARK mixed setup executed with real robots, where the performance is similar to the values obtained in simulation, hence validating the simulated M-ARK system. A video of one of the runs performed over the Sheffield-Rome labs is available as supplementary material.

### *B. Collective Resource Exploitation (CRE)*

The second case study demonstrates how to obtain a multi-swarm system with robots that (i) can move and sense the environment in different ways, and (ii) perceive and interact with each other. We investigate the coordination of two types of robot swarms, the slow ground swarm (SG) and the fast flying swarm (FF), in a resource harvesting task connected through M-ARK. The arena is divided in two regions, each containing resources to be exploited. The FF swarm needs to make a collective decision about the best region to be exploited [21], and should direct the SG swarm in the corresponding area. The SG robots can only perceive the resources locally while performing a random walk [17], and harvest the resource when passing over it. However, the SG swarm is not capable of determining the most resource-rich region on its own. Instead, it receives information from the FF swarm to move towards the selected region for harvesting. The ground robot swarm acts as the executor of the harvesting task assigned by the flying robots, utilising their internal compass to navigate through the terrain and collect the resources efficiently. The coordination between the flying and ground robot swarms results in an effective and efficient resource harvesting system.

*1) Experimental Setup:* A group of $N_G = 80$ simulated Kilobots, representing the SG swarm, is deployed within a square arena of $2\,\mathrm{m}$ side, while a group of $N_F = 20$ simulated Kilobots, representing the FF swarm, is deployed within a square arena of $0.5\,\mathrm{m}$ side. Since we are considering a shared environment, the smaller size of the arena for the FF swarm corresponds to a shorter time to cover any distance. Hence, the FF robots appear to move faster than SG robots. The shared

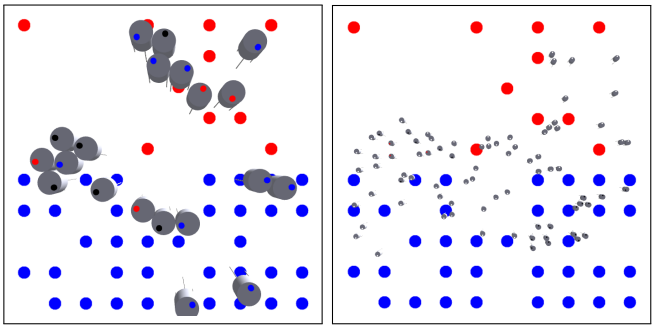

Fig. 6: CRE experimental setup. The red and the blue circles represent exploitable resources. Left: the FF swarm. The LED colour represents the robot's commitment state: black for *uncommitted* ($U$), red for *committed North* ($N$) and blue for *committed South* ($S$). Right: the SG swarm. Note that the SG arena is four times larger than the FF arena, in order to simulate a different motion speed of the robots.

environment is divided into two equal parts, north and south, and each may contain resources to be harvested by the SG robots, respectively $M_N$ resources in the north and $M_S$ in the south. The total number of exploitable resources present in each region of the arena is kept constant. This means that each time a SG robot harvests a resource, a new one appears in a different location within the same region. We consider two scenarios: one in which the number of resources in the south is greater than the resources in the north ($M_N = 10$ and $M_S = 35$ respectively, visible in Fig. 6), and the other in which the number of exploitable resources is the same ($M_N = M_S = 35$ resources).

*2) Robot Behaviour:* In this case study, we simulate a multi-swarm system in which swarms not only differ in motion speed, but also in the sensory-motor and communication abilities and in their overall behaviour. Robots belonging to the FF swarm have to make a collective decision and choose which of the two regions has the most resources. Following a well-established approach for decentralised collective decision-making [21], [24], each FF robot changes its internal state between *uncommitted* ($U$)—when no region is preferred over the other—and *committed North* ($N$) or *committed South* ($S$)—when the robot selects one of the two regions as the most profitable one. Robots always scan the arena performing a random walk [30], estimating the density of resources for both regions and broadcasting their internal state to neighbouring robots, both in the FF and SG swarms within range. The uncommitted robot can change to the state $N$ or $S$ in two ways: (i) spontaneously, with a probability proportional to the perceived density of resources in the region, or (ii) recruited by another committed robot. When in state $N$ or $S$, a robot switches to the state $U$ either (i) spontaneously, with a probability inversely proportional to the perceived density of resources, or (ii) inhibited by another robot committed to a different region. These simple rules are known to be sufficient to lead to a collective decision for one of the available alternatives [21], [24]. Finally, committed FF robots only communicate to SG robots when they are outside their preferred target region and SG robots can only receive messages from FF robots located on top of them (within a distance of $20\,\mathrm{cm}$ from the projected position on the ground).

The SG robots, instead, perform a random walk [30] to forage for resources in their current location. If an SG robot detects a communication signal from a FF robot, it rotates towards the FF robot's target region and travels straight for about 3 seconds before resuming the random walk to continue harvesting resources in the new location. This dynamic coordination strategy enables the SG robots to efficiently harvest resources in a profitable region by adapting their search based on the messages from the FF robots.

*3) M-ARK Communication Protocol:* The communication setup is similar to the one described in Section III-A3. First, the two M-ARK instances—hereafter, FF-ARK and SG-ARK—get connected, with FF-ARK acting as client and SG-ARK as server. SG-ARK sends to FF-ARK the *Active Resources* message, reporting information about the available resources in both regions, delivering a string message of $M_N + M_S$ characters, representing inactive and active areas in both regions, filled respectively with $0-1$ values. This enables FF-ARK to synchronise to changes in the shared environment resulting from resource harvesting by the SG robots and their reappearance in new positions. Also in this case, SG-ARK and FF-ARK exchange messages each $T_p = T_c = 2\,\mathrm{s}$. Conversely, FF-ARK sends to SG-ARK a message indicating the state of the FF swarm, in which the position and the commitment state of each FF robot is reported, so that SG-ARK can provide the ground robots with information about which FF robots are in communication range, and about their state. These are also string messages, composed of $5N_F$ bytes, i.e., 5 bytes per FF robots, where the first four correspond to the x and y position of an FF robot in the space and the fifth represents the commitment. This allows simulating the communication from FF to SG robots, enabling coordination in the resource harvesting task.

*4) Results:* Fig. 7 shows the results obtained over 100 independent runs in simulation, reporting the state of the multi-swarm in both the asymmetric ($M_N < M_S$) and symmetric ($M_N = M_S$) conditions. When the resource distribution is asymmetric, the FF swarm quickly converges towards consensus for the south region, with all robots converging to state $S$ (see top-left panel in Fig. 7). As a consequence, the SG swarm moves following the indications of the FF swarm towards the south region and remains there, harvesting the available resources (see the bottom-left panel). Note that, time-wise, the convergence of the SG swarm towards the south region is slower than the decision process performed by the FF swarm, both because SG robots move at a reduced speed and because they need to follow the indications of FF robots. When the resource distribution is symmetric, the decision dynamics are slower, but they always reach a consensus for either north or south (see the top-right panel in Fig. 7. Indeed, the FF swarm is either completely committed to the south region (63% of the runs), or to the north region (37% of the runs). As a consequence, also the SG swarm shows similar spatial dynamics, being distributed either in the north or in the south region, with a similar proportion as in the FF swarm

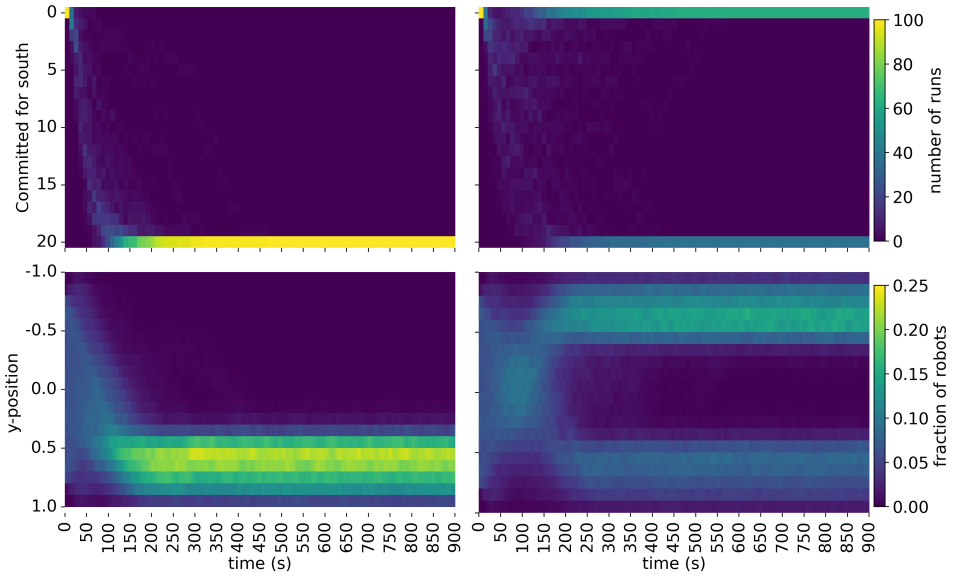


Fig. 7: CRE results. Top panels: heatmaps showing the distribution of the number of FF robots committed to the south region over time. Bottom panels: heatmaps showing the spatial distribution of SG robots in the north-south axis. Left panels: asymmetric case. Right panels: symmetric case.

consensus (see bottom-right panel). Interestingly, before the FF swarm makes a decision, the SG swarm concentrates in the middle of the arena between the two regions until it follows the consensus and moves either north or south. Finally, within this case study we also tested a hybrid simulated-physical M-ARK, to demonstrate the feasibility of running both kinds of experiments at the same time. In this case, the FF robots are implemented within the RoARK, while the SG robots are implemented in a ARGoS instance, as shown in the supplementary video.

## IV. CONCLUSIONS

M-ARK provides a versatile system to support research with multi-swarm robotics systems. The two case studies discussed in this paper provide examples of the many possibilities it offers. Through M-ARK, different swarms can interact, either indirectly through the environment or by means of situated communication messages. The former type of interaction enables stigmergy [2]. This is the easiest form of interaction that can be implemented through M-ARK, as it only requires that single ARK instances synchronise with the shared environment, as shown in Section III-A. Situated communication can also be implemented, as demonstrated in the second case study (see Section III-B), but is limited to the ability of Kilobots to upload data to the local ARK system. This limitation is specific to the Kilobot platform and can be overcome if M-ARK is expanded towards other platforms, as discussed below.

As already mentioned, physical interactions across multiple swarms are currently not possible within M-ARK. With the available technology, it may be possible to enable some rudimentary form of collision avoidance, by creating a no-walk zone in the local environment around the position where Kilobots from a different ARK are located. This can enable some form of density-dependent behaviour also in the case of MSI, which could be relevant for scalability studies given that robot density is the most limiting factor [31].

We have demonstrated how to connect remote labs enabling cross-country experiments. By supporting both simulated and real environments, M-ARK can boost research and demonstrations with physical swarms, favouring scientific collaboration and promoting MSI research. M-ARK can also be seen as a development tool, to analyse multi-swarm experiments starting with few robots and subsequently increasing the swarm size. Moreover, M-ARK reduces the simulation-reality gap enabling studying MSI by starting from fully simulated experiments running on several instances of the ARGoS simulator and ending with fully real experiments that interconnect several physical ARK instances.

Most importantly, it is possible to expand M-ARK beyond the client-server approach demonstrated here. In case of several clients, communication and synchronisation with the server may represent a bottleneck, especially if implemented with a time-based approach as demonstrated in this paper. When more than two swarms have to interact, an event-based decentralised solution is preferable. For example, a Distributed Hash Table could be exploited to support message exchange among local ARKs when events occur that may not be relevant to any swarm. Taken together, all these aspects can boost the possibilities of usage of M-ARK.

Finally, we have demonstrated that it is possible through M-ARK to study multiple swarms where robots have different sensory-motor capabilities. Even if the underlying robotic platform is always the Kilobot, differences can be created by virtualising both sensors and actuators. Moreover, differences in scale between local representations of the same environment support the implementation of different velocities for the robots, allowing for instance to simulate robots that move

faster in one of the environments, as showcased in both case studies presented here. Extending the M-ARK concept to other robotic platforms than the Kilobot can further remove barriers in the study of multi-swarm systems. Moving from M-ARK towards a *Multi-Augmented Reality for Swarms* (M-ARS) just requires a centralised local positioning system and a bidirectional communication channel between robots and the local ARS. This is feasible for several commercial platforms, including ground robots such as the e-puck [32] or even aerial robots like the crazyflie [33]. Thanks to augmented reality, swarm robotics research can reach far beyond what has been achieved to date.

## REFERENCES


[1] M. Dorigo, G. Theraulaz, and V. Trianni, "Reflections on the future of swarm robotics," *Science Robotics*, vol. 5, no. 49, p. eabe4385, 2020.

[2] V. Trianni and M. Dorigo, "Self-organisation and communication in groups of simulated and physical robots," *Biological Cybernetics*, vol. 95, no. 3, pp. 213–231, 2006.

[3] M. Dorigo et al., "Swarmanoid: A novel concept for the study of heterogeneous robotic swarms," *IEEE Robotics & Automation Magazine*, vol. 20, no. 4, pp. 60–71, 2013.

[4] W. Zhu, M. Allwright, M. K. Heinrich, S. Oğuz, A. L. Christensen, and M. Dorigo, "Formation control of UAVs and mobile robots using self-organized communication topologies," in *Swarm Intelligence: 12th International Conference, ANTS 2020*, ser. LNCS, vol. 12421. Cham: Springer, 2020, pp. 306–314.

[5] M. Rodríguez, A. Al-Kaff, A. Madridano, D. Martín, and A. de la Escalera, "Wilderness search and rescue with heterogeneous multi-robot systems," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020, pp. 110–116.

[6] A. Babić, I. Lončar, B. Arbanas, G. Vasiljević, T. Petrović, S. Bogdan, and N. Mišković, "A novel paradigm for underwater monitoring using mobile sensor networks," *Sensors*, vol. 20, no. 16, 2020.

[7] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm Robotic Behaviors and Current Applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, 2020.

[8] A. Antoun, G. Valentini, E. Hocquard, B. Wiandt, V. Trianni, and M. Dorigo, "Kilogrid: A modular virtualization environment for the Kilobot robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3809–3814.

[9] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The Robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.

[10] A. Reina, A. J. Cope, E. Nikolaidis, J. A. R. Marshall, and C. Sabo, "ARK: Augmented reality for Kilobots," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1755–1761, 2017.

[11] A. G. Millard, R. Redpath, A. M. Jewers, C. Arndt, R. Joyce, J. A. Hilder, L. J. McDaid, and D. M. Halliday, "ARDebug: an augmented reality tool for analysing and debugging swarm robotic systems," *Frontiers in Robotics and AI*, vol. 5, p. 87, 2018.

[12] W. Hönig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, "Mixed reality for robotics," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5382–5387.

[13] 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 *Proceedings of the IEEE 2015 NASA/ESA Conference of Adaptive Hardware and Systems (AHS)*. Los Alamitos, CA: IEEE Press, 2015, pp. 1–6.

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

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

[16] I. Slavkov, D. Carrillo-Zapata, N. Carranza, X. Diego, F. Jansson, J. Kaandorp, S. Hauert, and J. Sharpe, "Morphogenesis in robot swarms," *Science Robotics*, vol. 3, no. 25, p. eaau9178, 2018.

[17] C. Dimidov, G. Oriolo, and V. Trianni, "Random walks in swarm robotics: an experiment with Kilobots," in *Swarm Intelligence: 10th International Conference, ANTS 2016*, ser. Lecture Notes in Computer Sciences, vol. 9882. Cham: Springer, 2016, pp. 185–196.

[18] N. Cambier, D. Albani, V. Frémont, V. Trianni, and E. Ferrante, "Cultural evolution of probabilistic aggregation in synthetic swarms," *Applied Soft Computing*, vol. 113, p. 108010, 2021.

[19] G. Valentini, E. Ferrante, H. Hamann, and M. Dorigo, "Collective decision with 100 Kilobots: Speed versus accuracy in binary discrimination problems," *Autonomous agents and multi-agent systems*, vol. 30, pp. 553–580, 2016.

[20] 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 Robotics and Automation Letters*, vol. 1, no. 1, pp. 348–353, 2016.

[21] A. Reina, T. Bose, V. Trianni, and J. A. R. Marshall, "Effects of Spatiality on Value-Sensitive Decisions Made by Robot Swarms," in *Distributed Autonomous Robotic Systems: The 13th International Symposium*, ser. SPAR. Cham: Springer, 2018, pp. 461–473.

[22] M. S. Talamali, T. Bose, M. Haire, X. Xu, J. A. R. Marshall, and A. Reina, "Sophisticated collective foraging with minimalist agents: A swarm robotics test," *Swarm Intelligence*, vol. 14, no. 1, pp. 25–56, 2020.

[23] A. Font Llenas, M. S. Talamali, X. Xu, J. A. R. Marshall, and A. Reina, "Quality-sensitive foraging by a robot swarm through virtual pheromone trails," in *Swarm Intelligence: 11th International Conference, ANTS 2018*, ser. LNCS. Cham: Springer, 2018, vol. 11172, pp. 135–149.

[24] M. S. Talamali, A. Saha, J. A. R. Marshall, and A. Reina, "When less is more: Robot swarms adapt better to changes with constrained communication," *Science Robotics*, vol. 6, no. 56, 2021.

[25] L. Feola and V. Trianni, "Adaptive strategies for team formation in minimalist robot swarms," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4079–4085, 2022.

[26] C. Pinciroli, M. S. Talamali, A. Reina, J. A. R. Marshall, and V. Trianni, "Simulating Kilobots Within ARGoS: Models and Experimental Validation," in *Swarm Intelligence: 11th International Conference, ANTS 2018*, ser. LNCS, vol. 11172. Cham: Springer, 2018, pp. 176–187.

[27] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. D. Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271 – 295, 2012.

[28] L. Feola, A. Reina, M. S. Talamali, and V. Trianni, "Multi augmented reality for kilobots," 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8282827

[29] A. J. Ijspeert, A. Martinoli, A. Billard, and L. M. Gambardella, "Collaboration Through the Exploitation of Local Interactions in Autonomous Collective Robotics: The Stick Pulling Experiment," *Autonomous Robots*, vol. 11, no. 2, pp. 149 – 171, 2001.

[30] F. Bartumeus, M. G. D. Luz, G. Viswanathan, and J. Catalán, "Animal search strategies: A quantitative random-walk analysis," *Ecology*, vol. 86, pp. 3078–3087, 2005.

[31] H. Hamann and A. Reina, "Scalability in Computing and Robotics," *IEEE Transactions on Computers*, vol. 71, no. 6, pp. 1453–1465, 2022.

[32] J. M. Allen, R. Joyce, A. G. Millard, and I. Gray, "The Pi-puck Ecosystem: Hardware and Software Support for the e-puck and e-puck2," in *Swarm Intelligence: 12th International Conference, ANTS 2020*, ser. LNCS, vol. 12421. Cham: Springer, 2020, pp. 243–255.

[33] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA*, 2017, pp. 3299–3304.