Kilogrid: a novel experimental environment for the Kilobot robot

Gabriele Valentini[1,2] · Anthony Antoun[1] · Marco Trabattoni[1] · Bernát Wiandt[3] · Yasumasa Tamura[4] · Etienne Hocquard[5] · Vito Trianni[6] · Marco Dorigo[1]

Received: 11 August 2017 / Accepted: 4 January 2018 / Published online: 27 January 2018

Abstract We present the Kilogrid, an open-source virtualization environment and data logging manager for the Kilobot robot, Kilobot for short. The Kilogrid has been designed to extend the sensory-motor abilities of the Kilobot, to simplify the task of collecting data during experiments, and to provide researchers with a tool to fine-control the experimental setup and its parameters. Based on the design of the Kilobot and compatible with existing hardware, the Kilogrid is a modular system composed of a grid of computing nodes, or modules that provides a bidirectional communication channel between the Kilobots and a remote workstation. In this paper, we describe the hardware and software architecture of the Kilogrid system as well as its functioning to accompany its release as a new open hardware tool for the swarm robotics community. We demonstrate the capabilities of the Kilogrid using a 200-module Kilogrid, swarms of up to 100 Kilobots, and four different case studies: exploration and obstacle avoidance, site selection based on multiple gradients, plant watering, and pheromone-based foraging. Through this set of case studies, we show how the Kilogrid allows the experimenter to virtualize sensors and actuators not available to the Kilobot and to automatize the collection of data essential for the analysis of the experiments.

Keywords Kilogrid · Kilobot · Swarm robotics · Virtualization · Automated data collection · Tracking · Open source

✉ Gabriele Valentini
gvalentini@asu.edu

1 IRIDIA, Université libre de Bruxelles, Brussels, Belgium

2 Present Address: Beyond Center, School of Earth and Space Exploration, Arizona State University, Tempe, AZ, USA

3 Budapest University of Technology and Economics, Budapest, Hungary

4 School of Computing, Tokyo Institute of Technology, Tokyo, Japan

5 IRT Jules Verne, Nantes, France

6 ISTC, Consiglio Nazionale delle Ricerche, Rome, Italy

1 Introduction

Research in swarm robotics (Dorigo et al. 2014; Brambilla et al. 2013) is challenging for several reasons, ranging from conceptual problems affecting the design of distributed algorithms to experimental issues due to shortcomings of the available hardware. The latter often pose limitations to the former: due to the difficulty of running experiments with real robotic swarms, certain claims remain unverified or are supported only by *ad hoc* simulations. Experimental issues limit the possibility to showcase the value of algorithms and hold back swarm robotics from concrete application domains. In this paper, we present the *Kilogrid*, an experimental tool designed to enhance the abilities of the Kilobot robotic platform (Rubenstein et al. 2014a), making it easier to run meaningful swarm robotics experiments.

The Kilobot is a simple and inexpensive robot designed to perform large-scale swarm robotics experiments. Using Kilobots, the largest swarm robotics experiment performed to date in terms of number of robots involved—more than a thousand robots—was recently performed (Rubenstein et al. 2014b). The Kilobot has been designed keeping in mind the requirements of low-cost and sufficient functionality. The former dictates a simple hardware design with only an ambient light sensor and a pair of vibrating engines, and no moving parts such as gears or stepper motors. The latter was fulfilled through a simple but efficient infrared communication system which provides the means to rely on collective strategies to overcome the limits of the individual platform [e.g., communication with neighbors could compensate for the lack of encoders to support odometry (Rubenstein et al. 2014a)].

Additionally, particular attention was given to practical issues that limit the ability to make large-scale experiments. Simple procedures such as switching on the robots, programming them, or recharging their batteries may become cumbersome or even practically infeasible as the size of the swarm increases. Hence, to obtain *scalability*, several important design choices have been performed. Physically powering-on the Kilobots has been avoided by providing a low-power sleep mode from which the Kilobots can easily be woken up through infrared signals from an overhead controller (OHC). The OHC is key also for other actions, such as programming all the Kilobots at once, starting and halting the experiments or querying the battery voltage of the Kilobots (Rubenstein et al. 2014a). Charging the batteries of a large number of Kilobots can also be achieved simultaneously by moving all the Kilobots on a charging plate and sandwiching them with a top conductive grid, making it possible to apply a 6 VDC tension to the battery inputs (top connector and any of the Kilobot legs). Overall, the Kilobot design has solved several experimental issues that led to its adoption as a standard platform in several academic contexts, as demonstrated by successful experiments with tens or hundreds of Kilobots [e.g., collective decision making (Trianni et al. 2016; Valentini et al. 2016; Vigelius et al. 2014), collective transport (Rubenstein et al. 2013) and manipulation (Becker et al. 2013) of objects, self-assembly (Rubenstein et al. 2014b; Soorati and Hamann 2016) and disassembly (Gauci et al. 2017)].

However, the simplicity of the Kilobot as a robotic platform represents also a barrier to further progress of swarm robotics research beyond the above-mentioned initial, relatively simple experiments. Kilobots have limited sensing of the environment in which they are located, as they are provided with just a single ambient light sensor. For instance, Kilobots cannot perceive and avoid obstacles, such as objects or the arena walls (Valentini et al. 2015), which can disrupt their motion pattern and have a strong influence on the resulting collective behavior (Trianni et al. 2016; Valentini et al. 2016). Kilobots cannot perceive environmental features such as color, temperature, humidity, or gradients, meaning that several experimental activities are not possible as they would require a richer perception of the environment. Neither they can act in any other way than moving or communicating with neighbors. To overcome similar limitations, other robotic platforms are designed so that they can be extended with additional modules [e.g., there are several extension turrets for the e-puck robot to enhance vision, as well as communication and manipulation abilities (Gutiérrez et al. 2008; Liu and Winfield 2011; Mondada et al. 2009)]. However, this is not practical in terms of cost and flexibility of the approach, as an extension module would be required for each Kilobot and for any new sensory-motor modality. Moreover, Kilobots are not designed to be extended in any way through additional hardware.

A different way to provide additional sensory-motor abilities is through some form of *virtualization*: robots are provided with additional means to sense and act in their environment through a virtual or augmented reality system, which makes them perceive and act through modalities that are not available in the real world but are made possible in the augmented world. To this end, the location of each robot is usually tracked to support situated interaction (i.e., the augmented reality perceived by the robot is related to its position in the cyber-physical space) and some form of bidirectional feedback is put in place between the robot and the augmented reality management system, exploiting the available sensory and communication modalities. To provide an example, chemical communication through pheromones is typically a complex sensory modality to be implemented on robots, but a virtualization environment can easily provide such sensory system. Arvin et al. (2015a) and Garnier et al. (2013), for example, use a tracking system that detects the position of the robots, so that a trace of virtual pheromone can be recorded after their passage. Neighboring robots are informed about the presence and quantity of virtual pheromone through wireless communication or other sensory modalities [e.g., light intensity from an overhead projector, as in Garnier et al. (2013)]. In this way, the sensory-motor loop can be closed and robots can exploit a brand new interaction modality with the environment in which they move. Most importantly, the parameters regulating the behavior of the virtual sensory system (e.g., the evaporation of the virtual pheromone) are under complete control of the experimenter, which can change them at will to investigate their influence on the robot behavior.

Besides additional sensing and acting modalities, there is another experimental issue that has not been tackled by the Kilobot design: there is no way to access the internal state of the Kilobot but through a serial cable connected to the Kilobot. This is a strong limiting factor for both debugging and collecting data recorded by the Kilobot during an experiment, as this is clearly not a scalable operation. A tracking system with overhead cameras can provide data about the position of the Kilobots, while their internal state can, at least in part, be displayed through the colored light emitting diode (LED) available on the Kilobot and then extracted from the video feed. However, for any experiment requiring more than just logging positional data and a few bits of information about the Kilobots' internal state, visual tracking alone is not sufficient; instead, a bidirectional communication channel between the Kilobots and a remote workstation is necessary.

Existing virtualization environments are based on various technologies to support tracking and/or bidirectional interaction between robots and the virtualization management system. Some of them use wireless communication (Brutschy et al. 2015; Pickem et al. 2016; Reina et al. 2015), other employ RFID tags (Khaliq and Saffiotti 2015; Khaliq et al. 2014; Werfel 2012), or projected light (Arvin et al. 2015a, b; Garnier et al. 2013; Melhuish et al. 1999). Recently, an augmented reality system for the Kilobots (ARK) has been proposed in Reina et al. (2017). The ARK system is composed of a set of cameras used for tracking individual Kilobots, a modified OHC to provide Kilobots with information about the virtual environment, and a software architecture coupled to a GPU card which simulates the virtual environment and coordinates the Kilobots in the physical one. The ARK system allows the

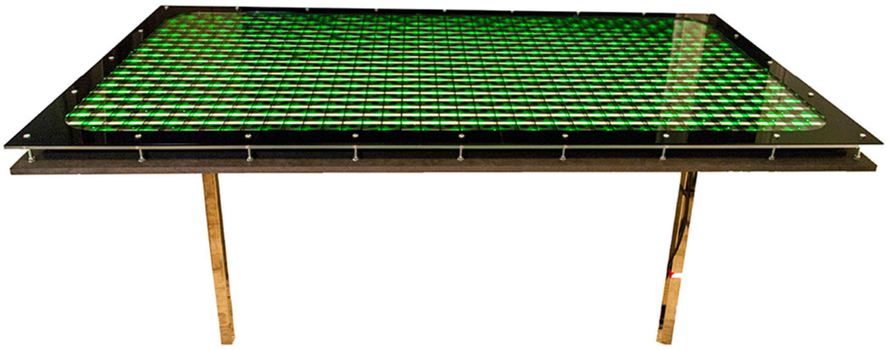


Fig. 1 Picture of a Kilogrid system composed of 200 Kilogrid modules that are positioned in 10 rows of 20 modules each

tracking of the Kilobot position and of its LED color (3 bits of information). Due to the centralized nature of the ARK system, each Kilobot can only receive up to 24 bits of information in a single message (where 10 bits give the robot identifier). Additionally, the ARK system has a low communication frequency (e.g., 150 Kilobots can receive a message each every 2.5 s) which limits the scalability of Kilobots experiments (i.e., the larger the swarm size, the lower the communication frequency). The ARK system can potentially be used to download data stored inside the Kilobot memory at the end of an experiment; however, this solution does not allow for the logging of large amounts of data because of the limited onboard memory of the Kilobots. Moreover, the ID assignment of the ARK system is prone to errors [cf. Conclusions in Reina et al. (2017)] which is a prerequisite for tracking the motion of robots. In contrast, the Kilogrid is a purely decentralized system in which virtualization is performed by each module independently. As a consequence, it allows for a high communication frequency, is scalable with the size of the swarm, and can be used to collect large amounts of data in real-time (i.e., while an experiment is being run).

The Kilogrid is the tool we have devised to overcome all the above-mentioned limitations and to provide a flexible system to support experimentation with Kilobots. As the name suggests, the Kilogrid is an experimental arena consisting of a grid of modules able to interact with Kilobots through infrared communication (see Fig. 1). Thanks to the Kilogrid, a robust bidirectional communication channel can be put in place between any Kilogrid module and the Kilobots moving above it, hence creating a flexible virtualization environment capable of simulating different sensory and acting modalities at the same time. Moreover, the Kilogrid works as a tracking system and a data logging manager, as its modules can record the position and internal state of the Kilobots during an experiment and forward them to a remote workstation. It facilitates experimentation with the Kilobot by providing an experimental arena that replaces the OHC and that allows new forms of interaction with the Kilobots. The flexibility of the Kilogrid may also pave the way to potential misuses as researchers, in contrast to the principles of swarm robotics, could provide robots with global information (see Sect. 4). Moreover, while hardware virtualization facilitates the development of control algorithms, it does not support that of new hardware solutions and could ultimately bring us closer to simulation studies. Nonetheless, the primary capabilities of the Kilobot robot, i.e., onboard computation, motion and communication, are left unaltered by the Kilogrid and its use allows researchers to experiment with a broadened set of experimental scenarios.

In this paper, we present the Kilogrid by describing its architecture and design in Sect. 2. We demonstrate its most important features with targeted experiments which are presented in Sect. 3. Finally, we discuss availability, production costs, and limitations of the Kilogrid system in Sect. 4. Section 5 concludes the paper with an outlook to novel experimental activities made possible by the Kilogrid.

2 The Kilogrid architecture

The Kilogrid provides researchers with a distributed communication system to interact with the individual robots in a swarm of Kilobots in a way that is situated in space (Mathews et al. 2015; Støy 2001). In order to do so, the Kilogrid is composed of three different parts: (i) a set of *Kilogrid modules*, (ii) a *dispatcher*, and (iii) a *KiloGUI* application. The Kilogrid modules are computationally independent units that can communicate with the Kilobots; once connected together, the modules form an experimental environment that can host a swarm of Kilobots. The dispatcher provides a bidirectional communication interface between the Kilogrid modules and the remote workstation. Finally, the KiloGUI application allows researchers to control the Kilogrid and to experiment with a swarm of Kilobots. In the rest of this section, we describe the architecture and the functioning of the Kilogrid module (Sect. 2.1), of the dispatcher (Sect. 2.2), and of the KiloGUI application (Sect. 2.3).

2.1 The Kilogrid module

The Kilogrid module consists of a printed circuit board (PCB) encased with a set of laser-cut structural components. The module can be programmed by the user to define complex functionality in the same manner as for the Kilobots. The module has a size of 10 cm $\times$ 10 cm and is divided into four *cells* of 5 cm $\times$ 5 cm. A cell is the elementary unit with which the Kilogrid interfaces with the Kilobots. Each cell can (i) communicate with the Kilobots by exchanging infrared (IR) messages and (ii) provide the experimenter with visual feedback in the form of colored light.

Functional architecture Figure 2 shows the functional diagram implemented in the PCB of the Kilogrid module. The Kilogrid module features a single microcontroller unit (MCU) that executes a user-defined control software and that interfaces this software with the low-level functionality of the module. The MCU of the Kilogrid module is an ATmega328P with 32 KB of memory running at a frequency of 8 MHz, the same as in the Kilobot (Rubenstein et al. 2014a). The MCU can control the four cells of the module independently of each other by means of a multiplexing system (i.e., two multiplexers and one LED driver).

The hardware of a cell consists of one IR transmitter, one IR receiver, and two RGB LEDs. The IR transmitter and the IR receiver are positioned in the middle of the cell. For compatibility purposes, the electronics of these devices is implemented in the exact same way as in the Kilobot. Each cell can transmit one 10-Bytes message (1 Byte for the message type and 9 Bytes for data) every 0.5 s; once a message is received, its integrity is validated using cyclic redundancy check (CRC). The received infrared signal is amplified and filtered in the exact same way as in the Kilobot, which permits to estimate the distance between a transmitting Kilobot and the cell receiving its message and vice versa. Figure 3 shows the average strength of the IR signal resulting from an experiment where 20 Kilobots were placed on top of 20 cells at different distances and orientations and let them exchange messages for 2 min. The results show a linear scaling of the signal strength as a function of the distance between the transmitter and the receiver. The two RGB LEDs are positioned at two opposite corners of the cell, and the cells are placed so that the LEDs are adjacent to the middle of

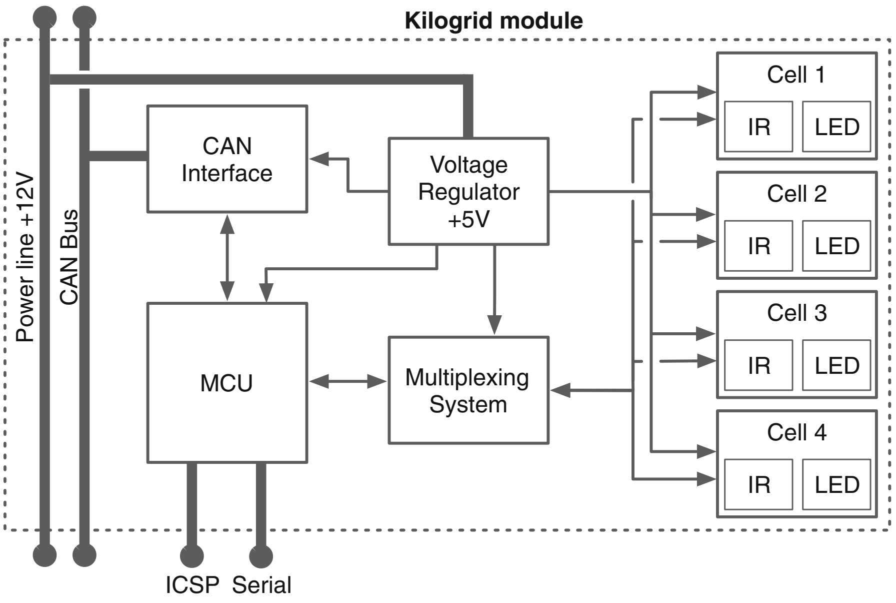


Fig. 2 Functional diagram showing the architecture of the Kilogrid module. The MCU is connected to the controller area network (CAN) bus by means of the CAN interface and to the four cells by means of a multiplexing system. Each cell is characterized by an IR transceiver and an RGB LED. The voltage regulator powers all the subsystems of the Kilogrid module. The module has an ICSP interface (in-circuit serial programming) and a serial interface to program and to debug the MCU

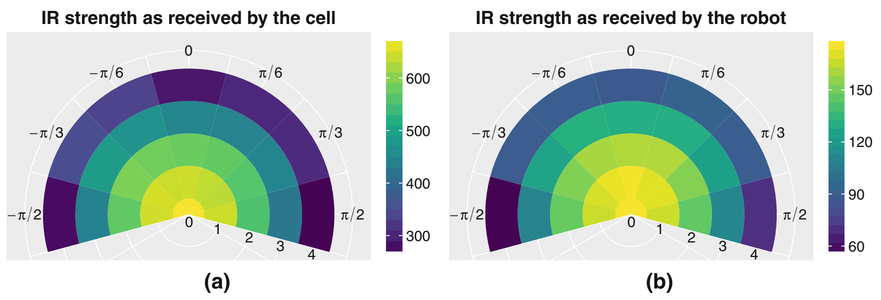


Fig. 3 Average strength of IR messages as a function of the distance $d \in \{0, 1, 2, 3, 4\}$ cm and orientation $\phi \in \{\pm\pi/2, \pm\pi/3, \pm\pi/6, 0\}$ rad between robot and cell: **a** messages sent by Kilobots and received by cells; **b** messages sent by cells and received by Kilobots (Color figure online)

the PCB sides. The two LEDs are driven in parallel by the MCU through an LED driver and always show the same color. Thanks to the presence of an LED driver, the LEDs of the cell have a resolution of 24 bits (i.e., 8 bits for each RGB component) which allows the experimenter to accurately control the color and the intensity of the emitted light.

A Kilogrid module is capable to communicate with the dispatcher as well as with any other module of the Kilogrid by means of a controller area network (CAN). In order to do so, the Kilogrid module has a CAN interface that consists of a CAN controller and a CAN transceiver connected to the MCU of the module. This CAN interface supports up to 112 Kilogrid modules on the same physical bus. In Sect. 2.2, we will explain how we designed the dispatcher in order to extend the maximum number of modules that a Kilogrid

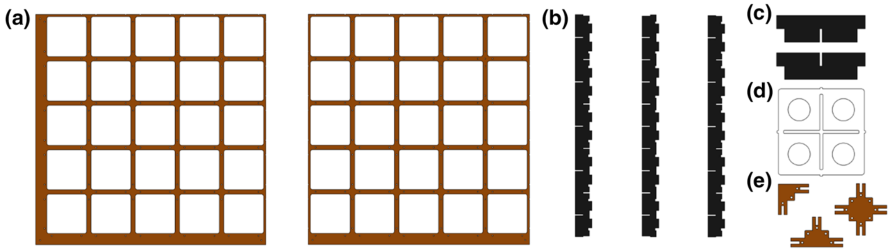


Fig. 4 Structural components of the Kilogrid: **a** support islands (corner and side versions), **b** IR barriers (female side, male sides and female middle versions, **c** internal IR barriers (male and female versions), **d** light diffuser, and **e** PCB supports (corner, side, and middle versions)

can contain. As illustrated in Fig. 2, the Kilogrid module contains a segment of the CAN bus that is routed to the CAN interface and to two different connection points placed at the opposite sides of the module. The two connection points allow the extension of the CAN bus with the corresponding segments of up to two other devices (i.e., a pair of modules, a module and the dispatcher or only one module) by means of twisted-pair cables.

All subsystems featured in the Kilogrid module are connected to an onboard voltage regulator. The voltage regulator is connected to a 12 V power line and supplies the module with 5 V. During normal operation, the Kilogrid module requires on average 185 mA and has an average power consumption of 2.22 W. As for the CAN bus, the Kilogrid module contains a segment of the 12 V power line with two connection points. The two connection points permit the 12 V power line to be extended with the corresponding segments of up to two other devices (i.e., a pair of modules, a module and an external power supply or only one module) by means of twisted-pair cables. This design solution permits to power up to 10 Kilogrid modules connected in parallel on the same 12 V power line.

The Kilogrid module can be programmed both through the ICSP interface and through the CAN interface (see Fig. 2). The ICSP interface is used to flash the firmware of the module when this has no valid running firmware (e.g., soon after production of the PCB board or due to bugs affecting a previously flashed firmware), while the CAN is the preferred programming interface during normal operation because it allows to simultaneously program all the Kilogrid modules. The firmware of the module is written using the C language and is based on an extended version of the *kilolib* library (originally developed for the Kilobot[1]), which provides a development environment familiar to Kilobots' users. Each module has a unique 10-bit identifier stored in the EEPROM memory which permits to localize its position in space (respectively, 5 bits for the row coordinate and 5 bits for the column coordinate). Additionally, the module also has a bidirectional serial port for debugging purposes.

Structural architecture The structural architecture of the Kilogrid is composed of five different components designed in different variants depending on their position (see Fig. 4). The support islands are made from wood panels and provide a first means to align other components of the Kilogrid. The IR barriers are made of black Plexiglas, divide each module into four equal cells, and prevent cells from interfering with each other. The light diffuser is made of white translucent Plexiglas and has four circular holes that allow the propagation of IR signals between a cell and the Kilobots on its top, and its primary role is to diffuse the light emitted by cell's LEDs. Finally, the PCB supports are made from wood panels and provide a second means to align the components of the Kilogrid. Differently from our initial prototype

[1] See http://www.kilobotics.com/.

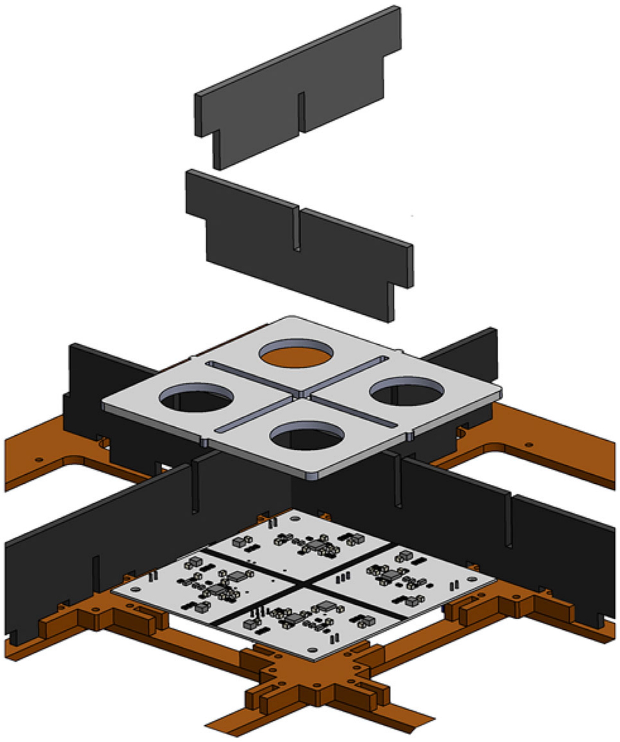


Fig. 5 Illustration of the Kilogrid assembly flow

of the Kilogrid (Antoun et al. 2016), all structural components are manufactured using 2D laser-cut technology reducing production costs associated with 3D printing technology.

Figure 5 shows the 3D exploded architecture of the Kilogrid and exemplify the assembly process. Support islands provide the first layer of the architecture and should be placed on top of a flat surface (e.g., a table). The PCB supports represent the second layer of the architecture; they are screwed to the support islands and allow to align the IR barriers and the PCB of the Kilogrid modules. After all IR barriers have been placed, the PCB are positioned on the PCB supports and secured by means of screws. The light diffuser and the internal IR barriers are positioned over each module. Finally, the entire assembly of the Kilogrid is placed under a single transparent Plexiglas surface on which the Kilobots move. The Plexiglas surface is the same that was extensively used in previous experiments without the Kilogrid and, if Kilobots are properly calibrated, it does not alter their motion properties (Valentini et al. 2015, 2016).

2.2 The dispatcher

The Kilogrid dispatcher is a device that interfaces a grid of modules connected together with a remote workstation operated by the experimenter. The dispatcher substitutes the original overhead controller designed for the Kilobot, and it incorporates all its functionalities. It can be used to issue commands to the Kilobots through the Kilogrid modules, such as to start, stop, or pause the execution of their controllers, to show the level of their batteries voltage, or to calibrate their motors. Differently from the OHC, the dispatcher provides a bidirectional communication channel between the Kilobots (located on the Kilogrid modules)

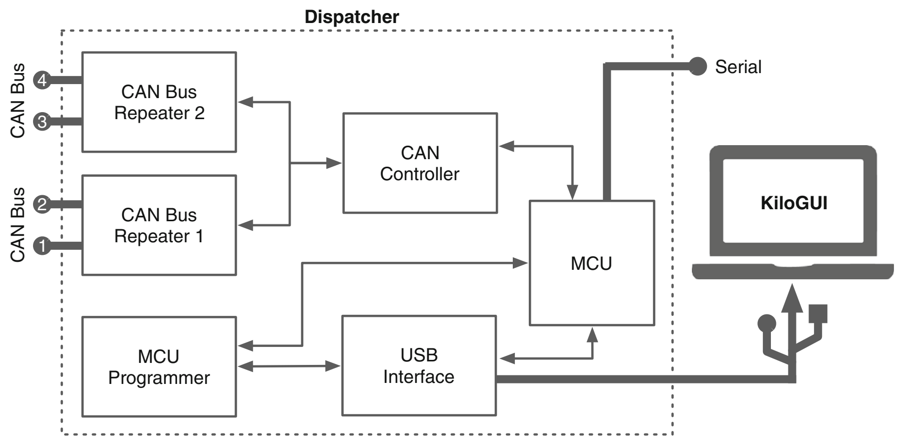


Fig. 6 Functional diagram showing the architecture of the dispatcher. The MCU is connected to the CAN bus by means of the CAN interface, to the remote workstation by means of the USB interface and to the MCU programmer. Two CAN repeaters divide the CAN bus in four lines. The MCU programmer is connected to the remote workstation by means of the USB interface. The dispatcher has a serial interface to debug the MCU

and the KiloGUI software running on a remote workstation which allows to log data during an experiment. The dispatcher consists of a PCB only and has no mechanical role in the Kilogrid system. However, to protect the PCB, it is encased between two layers of laser-cut plexiglass of 14 cm $\times$ 13 cm.

Functional architecture Figure 6 shows the functional diagram implemented in the PCB of the Kilogrid dispatcher. The core of the dispatcher is represented by its primary MCU which, similarly to both the Kilogrid module and the Kilobot, consists of an ATmega328P microcontroller. The MCU is connected to the grid of modules by means of the CAN interface and to the KiloGUI application by means of the USB interface. Its role is to transmit command issued from the KiloGUI application to the Kilogrid modules and to forward data received from the modules back to the KiloGUI application. The MCU of the dispatcher is connected to a serial port which allows to debug the control software of the dispatcher in extreme situations whereby the USB interface cannot be accessed. Similarly to the overhead controller, the dispatcher mounts an MCU programmer which is connected to the USB interface and can be used by the KiloGUI application to program the primary MCU of the dispatcher.

The CAN interface mounted by the dispatcher consists of only the CAN controller (cf. Fig. 6). Differently from the Kilogrid modules, the dispatcher is provided with two CAN repeaters rather than a single CAN transmitter. CAN repeaters physically decouple the CAN network into 4 separate buses each allowing up to 112 connected nodes. To allow for the propagation of messages between the two CAN repeaters, the speed of the CAN network is set to 250 Kbps. Each CAN message consists of 64 bits of payload available to the Kilogrid user and 51 bits for addressing (11-bit identifier), synchronization, and acknowledgment of the message. Communication within the CAN network can be peer-to-peer, multi-cast (either by row or by column), or broadcast. This design solution allows us to increase the maximum number of Kilogrid modules that can be connected to the CAN network from 112 modules to 448 modules (or, equivalently, 4.48 m^2 of experimental area). The maximum number of modules of a Kilogrid could be further increased by introducing additional CAN repeaters in the dispatcher; however, this would be at the expense of a reduction of communication speed.

Software architecture The dispatcher is used to interface the CAN network with a remote workstation and its firmware implements two types of communication mechanisms: forwarding and polling. When no experiment is being executed on the Kilogrid, communication on the CAN network can be initiated only by the KiloGUI application and is destined to either the dispatcher itself or to the modules. In the latter case, the dispatcher forwards messages received from the USB interface to the CAN network and functions as a protocol converter. Differently, when an experiment is being executed on the Kilogrid, the dispatcher coordinates the communication between the modules and the remote workstation using a polling mechanism that prevents collisions on the bus.

Each Kilogrid module stores the CAN messages it wants to transmit to the KiloGUI application into a buffer. The dispatcher periodically polls the data contained in the buffer of each module starting from the module with coordinate (0, 0) and moving through the Kilogrid by columns. Once queried by the dispatcher, a module transmits the CAN messages contained in its buffer or a message saying it has no data to transmit; at that point, the dispatcher polls the next module. The dispatcher also buffers incoming messages from the modules. When the desired number of messages is collected, the dispatcher pauses the polling of the modules, sends the data contained in the buffer to the KilogGUI application through the USB interface, and empties its buffer before resuming the polling of the modules.

In order to allow the experimenter to stop an ongoing experiment from the remote workstation, the dispatcher also regularly polls the KiloGUI application. Every five polling loops of the Kilogrid modules (i.e., after polling each modules 5 times), the dispatcher queries the KiloGUI application for commands. KiloGUI can answer by sending either a stop experiment command or a carry on experiment command. When the experiment is stopped, the dispatcher transmits a CAN message to all modules signaling the end of the experiment, after which it transmits all data remaining in its buffer to the KiloGUI application.

2.3 Extension to the KiloGUI application

KiloGUI is a simple application developed to control (i.e., program and send commands) a swarm of Kilobots by means of the overhead controller. We extended the original KiloGUI application in order to interface the experimenter with the Kilogrid while keeping the possibility to use the OHC. All KiloGUI functionalities previously developed for the OHC have been ported to the Kilogrid system, such as the possibility to send specific commands to the Kilobots (e.g., run, pause, and start an experiment, show battery voltage, enter sleep or charging modes) and calibrate their motors.

Using the KiloGUI application, the experimenter can load a new firmware to both the Kilobots and the modules of the Kilogrid. In order to allow the experimenter to easily set different behaviors to different cells as a function of their spatial positions, the modules of the Kilogrid receive from KiloGUI specific parameter configurations after being programmed. Parameters for each module are specified in an xml file which is parsed by the KiloGUI application. During the execution of an experiment, KiloGUI records data sent by the modules through the dispatcher (e.g., the position and internal state of the Kilobots) into a data file which is available to the user for offline parsing. The received data are also simultaneously visualized within the KiloGUI application providing a means to easily debug Kilobots and modules' controllers. Additionally, we implemented a timer mechanism within the KiloGUI that allows the experimenter to set a maximum experiment time after which the experiment is automatically terminated.

3 Experiments

We demonstrate the potential of the Kilogrid by means of a system composed of 200 modules for a total of 800 cells and an area of 100 cm $\times$ 200 cm (see Fig. 1). In our experiments, we use swarms of Kilobots with size ranging between 50 and 100 units. We use a series of four different robotic experiments to illustrate how the Kilogrid can be used to virtualize sensors and actuators and to collect the data necessary to analyze experimental results. Video recordings of the Kilobot experiments are available in the supplementary material.[2] In our experiments, the Kilobots make use of a pair of low-level motion routines, respectively, random walk and gradient following, whose functioning is described below.

Random walk During the execution of the random walk routine, the Kilobot alternates between a rotation phase and a phase of straight motion. The Kilobot moves forward for a stochastic amount of time that is sampled from a Gaussian distribution with mean $\mu_{\mathrm{f}}^{\mathrm{rw}}$ and standard deviation $\sigma_{\mathrm{f}}^{\mathrm{rw}}$. Successively, the Kilobot turns on the spot in a random direction (clockwise and counterclockwise rotations have the same probability) for a stochastic amount of time sampled from a folded Gaussian distribution with mean $\mu_{\mathrm{r}}^{\mathrm{rw}}$ and standard deviation $\sigma_{\mathrm{r}}^{\mathrm{rw}}$.

Gradient following The Kilogrid system can be used to virtualize a gradient by programming cells to broadcast numeric values representing a discretized landscape. During the execution of the gradient following routine the Kilobot first moves forward for a constant amount of time $T_{\mathrm{sense}}^{\mathrm{gf}}$ during which the Kilobot senses the gradient and stores the maximum and minimum values perceived. Then, the Kilobot compares the most recently collected values of the gradient with those collected during the previous sensing period. When descending the gradient (ascending), the Kilobot compares the current minimum (maximum) with the previous minimum (maximum) and, in the case in which the former is larger (smaller) than the latter, the Kilobot initiates a turn. The direction of the rotation is chosen randomly (clockwise and counterclockwise rotation have the same probability) and kept the same during successive turns unless the Kilobot is moving in the wrong direction for more than $T_{\mathrm{stuck}}^{\mathrm{gf}}$ seconds. The Kilobot rotates for a stochastic amount of time that is sampled from a folded Gaussian distribution with mean $\mu_{\mathrm{r}}^{\mathrm{gf}}$ and standard deviation $\sigma_{\mathrm{r}}^{\mathrm{gf}}$. After this period of time, the Kilobot begins a new sensing period while moving forward.

3.1 Exploration and obstacle avoidance

As mentioned above, the Kilobot lacks the sensors necessary to perceive the presence of passive obstacles (Rubenstein et al. 2014a). When experimenting with large number of Kilobots, the borders of the arena confining the swarm (Becker et al. 2013) represent one of the primary obstacles hindering the robots' motion (another being the robots themselves). Depending on the specific experiment, other (possibly scattered) obstacles (Rubenstein et al. 2013) or virtual borders delimiting different areas (Valentini et al. 2016) may be part of the considered scenario. The Kilogrid can be used to virtualize a proximity sensor so that the Kilobot becomes capable of coping with this and similar experimental conditions. This is done by programming selected cells to convey the necessary information to the Kilobots. We exemplify this feature of the Kilogrid by performing two experiments in which the Kilobots' task is to explore an environment while avoiding obstacles: in the first experiment the

[2] See http://iridia.ulb.ac.be/supp/IridiaSupp2017-001/.

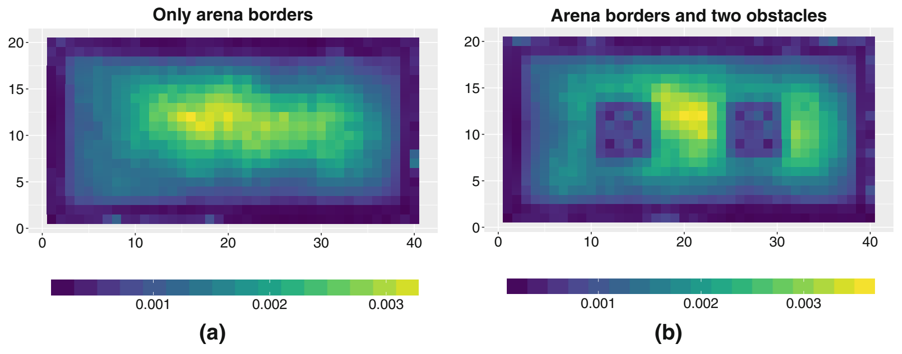


Fig. 7 Spatial probability distribution of a swarm of 100 Kilobots performing an exploration and obstacle avoidance behavior with **a** only the arena borders as obstacles and **b** including two square obstacles in the middle of the arena (Color figure online)

only obstacles are the arena borders, while the second experiment also includes two circular obstacles in the middle of the arena.

Module Controller In this experiment the Kilogrid is programmed to perform two functionalities: broadcast a gradient landscape and track the Kilobots positions. All cells of the Kilogrid are programmed to broadcast a numeric value. Cells in the proximity of the arena and obstacle borders broadcast the value 2 and cells adjacent to these cells broadcast the value 1, while the remaining cells in the middle of the arena broadcast the value 0. Simultaneously, the cells of the Kilogrid can receive messages from the Kilobots. All messages received are forwarded to the remote workstation and used to track the Kilobots' positions over time.

Kilobot controller A Kilobot begins the execution of an experiment by performing a random walk with parameters $\mu_{\mathrm{f}}^{\mathrm{rw}} = 10$ s and $\sigma_{\mathrm{f}}^{\mathrm{rw}} = 1$ s for the forward motion and $\mu_{\mathrm{r}}^{\mathrm{rw}} = 0.5$ s and $\sigma_{\mathrm{r}}^{\mathrm{rw}} = 3$ s for the rotation motion. As soon as the Kilobot perceives the proximity of an obstacle, it enters the gradient following routine and seeks to escape the perceived obstacle (parameters: $T_{\mathrm{sense}}^{\mathrm{gf}} = 5$ s, $T_{\mathrm{stuck}}^{\mathrm{gf}} = 20$ s, $\mu_{\mathrm{r}}^{\mathrm{gf}} = 3$ s, $\sigma_{\mathrm{r}}^{\mathrm{gf}} = 0.5$ s). The Kilobot resumes performing a random walk when the perceived gradient equals 0. During the entire execution of the experiment, the Kilobot signals its spatial location to the Kilogrid by broadcasting its Kilobot identifier once every 2 s. This communication frequency is set by default, can be varied by the user, and is used in all our case studies.

Experiments We used a swarm of 100 Kilobots and performed five independent runs with a duration of 60 min for each of the two experimental setups. Figure 7 shows the results of our robot experiments. When the only obstacles obstructing the motion of the Kilobots consist of the arena borders (see Fig. 7a), the swarm distributes approximatively evenly in the internal portion of the arena avoiding the regions of the environment in the proximity of the borders. When adding two circular obstacles in the arena (see Fig. 7b), the swarm is still able to successfully avoid the obstacles but this time its spatial distribution is concentrated in the center of the arena with similar but lower densities at the sides.

3.2 Site selection based on multiple gradients

As shown in the previous experiment, the Kilogrid can be used to let the Kilobots perceive their environment. This feature can be used by the experimenter to create complex experimental setups composed of different spatial regions and to guide the Kilobots between them. In this section, we consider a site selection scenario where a swarm of Kilobots is required to choose one of two available sites. We use the Kilogrid to partition the environment into a central nest with two sites at its side. This scenario is the same considered in Valentini et al. (2016), where Kilobots relied on an external light source and a phototaxis behavior to navigate the environment. We show here how the Kilogrid can be used to virtualize different spatial regions and to guide the motion of Kilobots between different regions by means of a combination of multiple gradients. An advantage of this approach over previous studies is that, by using the Kilogrid, the experimenter has complete control of all experimental conditions and related parameters.

Module controller In this experiment the modules of the Kilogrid provide the Kilobots with information about the type of regions in the environment (i.e., the nest and the red and blue sites), the quality of the sites, and a set of navigation gradients. The type of a region is sensed by the Kilobots and affects their behavior. Modules representing the sites also provide information about the site's quality and abstract the sensing of noisy measurements: each module broadcasts a random value that is normally distributed with mean $\rho_{\text{red}} = 1.0$ and $\rho_{\text{blue}} = 0.9$, respectively, for the red and blue site, and variance $\sigma^2 = 0.1$. Sampling of noisy qualities is performed independently by each module every second. Additionally, modules broadcast four different gradients that are used by the Kilobots for navigation: a gradient for obstacle avoidance (as performed in the previous experiment), two linear gradients to move toward each of the two sites, and a valley-like gradient to move from either of the two sites to the nest. The Kilogrid is also used to monitor the development of the experiment and to collect data necessary for the following analysis.

Kilobot controller A Kilobot executes the finite-state machine described in Valentini et al. (2015, 2016). It repeatedly alternates a period of dissemination, in which it broadcasts locally its opinion while performing a random walk (parameters: $\mu_{\text{f}}^{\text{rw}} = 25$ s, $\sigma_{\text{f}}^{\text{rw}} = 1.5$ s, $\mu_{\text{r}}^{\text{rw}} = 0.5$ s, and $\sigma_{\text{r}}^{\text{rw}} = 3$ s), and a period of exploration, in which it evaluates the quality of a certain site, again, while performing a random walk (parameters: $\mu_{\text{f}}^{\text{rw}} = 15$ s, $\sigma_{\text{f}}^{\text{rw}} = 1.5$ s, $\mu_{\text{r}}^{\text{rw}} = 0.5$ s, and $\sigma_{\text{r}}^{\text{rw}} = 3$ s). The duration of the dissemination state is proportional to the quality of the site associated to the Kilobot's current opinion. The dissemination state is executed within the nest. During this period, the Kilobot collects the opinions of its neighbors in a FIFO memory which contains 4 opinions. Before transitioning to the exploration state, the Kilobot switches opinion according to which opinion is most frequent in the set defined by the union of the FIFO memory and its current opinion (i.e., it applies the majority rule). In order to navigate between the different regions of the environment and to avoid colliding with the arena walls, the Kilobot execute a gradient following behavior using one or more gradients at the same time (parameters: $T_{\text{sense}}^{\text{gf}} = 5$ s, $T_{\text{stuck}}^{\text{gf}} = 30$ s, $\mu_{\text{r}}^{\text{gf}} = 3$ s, $\sigma_{\text{r}}^{\text{gf}} = 0.5$ s). By doing so, the Kilobot can, for example, avoid the arena walls while moving from the nest to the red site (i.e., obstacle avoidance gradient and linear gradient toward the red site) or ensure that it remains within a specific region of the environment (i.e., obstacle avoidance and valley-like gradients to stay in the nest). Once in a site, the Kilobot collects noisy quality values broadcast by the cells of the Kilogrid and averages these values to estimate the site quality.

Experiments We used a swarm of 100 Kilobots and performed 10 independent runs with a duration of 120 min each. Initially, Kilobots are randomly distributed in the environment, begin the execution of the experiment in the exploration state, and 50 of them favor the red site while 50 favor the blue site. As shown in Fig. 8a, in all 10 experiments the swarm is able to make a collective decision for the best site (i.e., the red site) in approximately 7000 s. Figure 8b shows the spatial distribution of Kilobots computed over the 10 runs. The presence

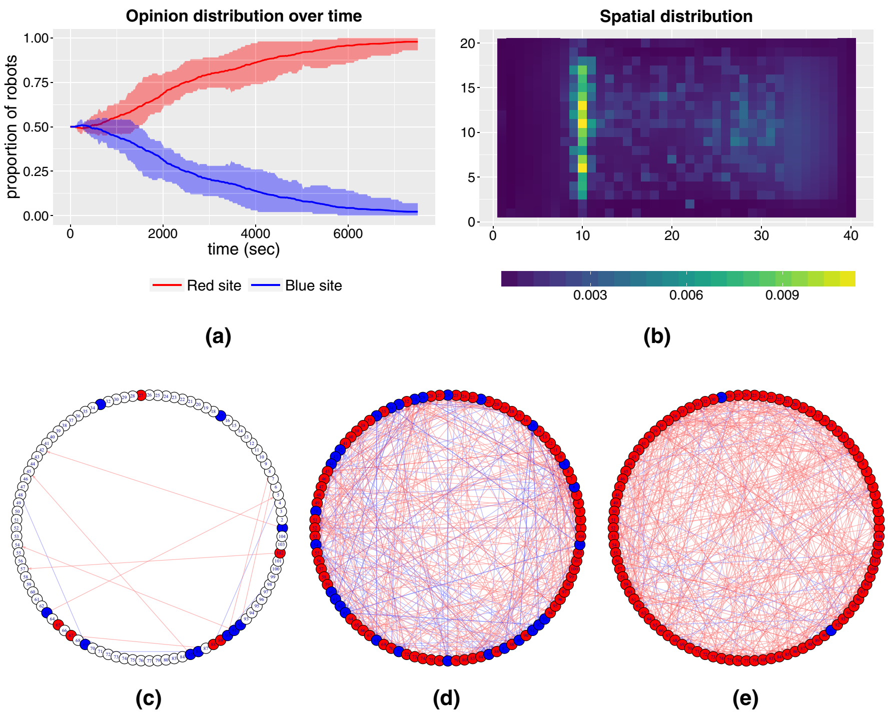


Fig. 8 Site selection using multiple gradients with 100 Kilobots: **a** opinion distribution over time; **b** spatial distribution of Kilobots in the environment; and interaction network at time **c** 100 s, **d** 2500 s and **e** 7000 s. The color of the nodes in the network represents the current favored site of a Kilobot (i.e., red site and blue site), and that of an edge represents the last perceived opinion of a neighbor. Nodes colored in white represent Kilobots for which tracking messages were not yet available at time 100 s (Color figure online)

of different regions in the environment (i.e., sites and nest) can be recognized by the two vertical lines of cells with higher density which correspond to the point in the environment were Kilobots trigger the gradient following behavior turning on the spot to remain in the nest or in one of the two sites. The density of Kilobots in the nest is approximately uniform as well as in both the sites. The red site (i.e., right side of Fig. 8b) has an higher density of Kilobots than the blue site which results from the swarm making a collective decision for that site. Finally, Fig. 8c–e shows the network of interaction between Kilobots at the beginning (100 s), in the middle (2500 s), and toward the end of the decision-making process (7000 s). The network of interaction provides us with information about which Kilobot is interacting with which other Kilobot and the type of information these Kilobots are sharing (i.e., the actual and the last perceived opinions, respectively, the color of the nodes and that of the edges). It is worth noting that this type of data can be collected from an experiment thanks to the duplex communication channel provided by the Kilogrid while cannot be collected using simpler virtualization system relying on cameras for tracking (e.g., the ARK system described in Sect. 1).

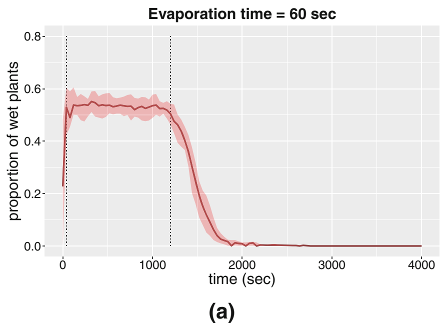
3.3 Plant watering

Through the use of the Kilogrid, the ability of the Kilobot to interact with its environment are not limited to sensory perceptions but also include the possibility to modify the environment by means of virtual actuators. When a certain physical interaction between a Kilobot and a region of the environment is required by the experimenter, the cells of the Kilogrid can be programmed to change their internal state upon the reception of a dedicated message from the Kilobot. We illustrate this feature of the Kilogrid by considering a plant watering scenario in which a swarm of Kilobots is required to water the plants of a virtual garden. In this scenario, we use the Kilogrid to virtualize the deposition of water over each plant by individual Kilobots as well as the evaporation of the deposited water. Each cell of the Kilogrid represents a plant and, at all times, the internal state of a cell is either *dry* or *wet*.

Module controller The modules of the Kilogrid perform three functionalities: virtualize the presence of a plant with its water deposition and evaporation mechanisms, signal the proximity to the arena borders to the Kilobots, and log the data required for our following analysis. Each cell of the Kilogrid is programmed to broadcast its current water state (i.e., dry or wet) together with its identifier (i.e., spatial coordinates) and, in the case of wet cells, the identifier of the last Kilobot who watered that cell. Upon reception of a message from a Kilobot, the module compares the identifier contained in the message with that of the cell receiving the message and, if these two identifiers coincide, the state of the cell is changed from dry to wet for a duration of T_{evap} seconds. After this period of time, the state of the cell is reverted back to dry. Cells in the proximity of the arena borders additionally broadcast a numeric value representing a gradient as done in previous experiments (Sects. 3.1 and 3.2). Finally, every time that a cell changes its internal state (i.e., from wet to dry or vice versa), the respective module signals the occurrence of the relative event to the remote workstation which logs this information and allows us to analyze the performance of the system.

Kilobot controller Each Kilobot is programmed to carry a limited amount of water. At the beginning of the experiment, a Kilobot performs a random walk behavior for a stochastic amount of time uniformly distributed in (0; 15] s (parameters: $\mu_{\mathrm{f}}^{\mathrm{rw}} = 20$ s, $\sigma_{\mathrm{f}}^{\mathrm{rw}} = 1.5$ s, $\mu_{\mathrm{r}}^{\mathrm{rw}} = 0.5$ s, $\sigma_{\mathrm{r}}^{\mathrm{rw}} = 3$ s). During this period of time, the Kilobot ignores possible messages received by the cells of the Kilogrid. This mechanism breaks the synchrony among the Kilobots, reduces the collisions of messages between different Kilobots and the same cell, and therefore eases the diffusion of Kilobots in space. After this period of time the Kilobot continues to move according to a random walk; however, it also looks out for dry plants. Upon the perception of a dry plant, the Kilobot (henceforth the focal robot) stops moving and broadcasts a message signaling the deposition of a unit of water. This message, which contains the coordinates of the plant to be watered and the Kilobot identifier, is broadcast for 5 s. During this period of time, the Kilobot also senses the state of the plant. If the cell virtualizing the plant begins to broadcast a wet message, and if the Kilobot identifier contained in this message coincides with that of the focal robot, the Kilobot decreases its internal quantity of water by one unit; otherwise, it resumes performing a random walk. Additionally, if the Kilobot perceives the arena borders it executes, if necessary, the same obstacle avoidance behavior described in Sect. 3.1.

Experiments We used a swarm of 100 Kilobots and performed two series of robot experiments where we varied the duration of the evaporation time (respectively, $T_{\mathrm{evap}} = 60$ s and $T_{\mathrm{evap}} = 120$ s). For each series of robot experiments, we performed five independent repetitions and collected information about the proportion of watered plants over time. At the beginning of an experiment, each Kilobot is provided with 100 units of water; the experiments terminate when all Kilobots have exhausted their water reservoir. Figure 9 shows the results of our robot experiments. In both experimental setups, the swarm of Kilobots converges (in about 50 s) to an approximately constant proportion of wet plants. When $T_{\mathrm{evap}} = 60$ (see Fig. 9a)



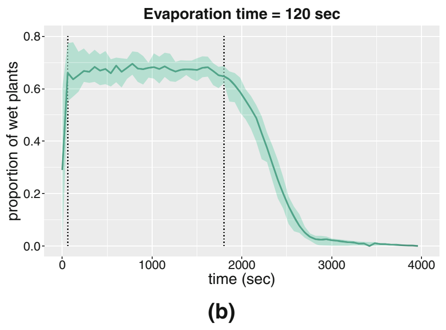


Fig. 9 Proportion of wet plants over time with a swarm of 100 Kilobots performing a plant watering behavior. Results for **a** an evaporation time of $T_{\text{evap}} = 60$ s, and **b** for an evaporation time of $T_{\text{evap}} = 120$ s. Shaded areas show the minimum and maximum over five independent runs, and vertical lines show periods of approximately constant performance

the swarm is able to keep watered approximately 53% of the dry plants for a duration of 1200 s while when $T_{\text{evap}} = 120$ (see Fig. 9b) the performance improves with 67% wet plants for a duration of 1800 s.

3.4 Pheromone-based foraging

A particularly popular approach in the literature of swarm robotics is the use of pheromone communication (Arvin et al. 2015a; Garnier et al. 2013; Payton et al. 2001). Inspired by the foraging behavior of certain species of ants (Goss et al. 1989), pheromones communication is a type of stigmergic behavior (Beckers et al. 1994) in which the environment is used as the medium carrying the information to be shared with other members of the swarm similarly to a message blackboard. We demonstrate how the Kilogrid can be used to virtualize indirect communication mechanisms such as those required for stigmergy by considering a foraging scenario. Kilobots are initially located in a central area called nest and can choose to forage from two sites (red and blue sites). Resources at the two foraging sites have the same quality but differ in their cost in terms of the traveling distance back and forth to the nest (Valentini et al. 2017). Kilobots looking for foraging sites can sense and deposit virtual pheromones from and into the cells of the Kilogrid and use this information to guide their navigation between the nest and the foraging sites.

Module controller The modules of the Kilogrid perform three functionalities: inform the Kilobots of the region type, broadcast a navigation gradient, and virtualize the deposition and evaporation of pheromones. Each cell in the Kilogrid belongs to a particular region such as the nest, the red and blue sites, or is part of the open environment that the Kilobots need to explore. As in the previous experiments, the cells of the Kilogrid are used to inform the Kilobots about the proximity of the arena borders. Additionally, cells broadcast a radial navigation gradient that is used by the Kilobots to move toward or away from the nest. In this experiments, the Kilobots can deposit virtual pheromones. The virtual pheromone is represented by an integer value stored within the module. Each cell broadcasts its current pheromone value, can receive pheromone deposited by the Kilobots and increment its current pheromone value, and implements an evaporation mechanisms whereby pheromone decreases over time at a constant rate of one unit every 60 s. In order to monitor the evolution of the experiment, each module collects tracking messages from the Kilobots moving on top of them and forwards these messages as well as messages about the amount of pheromones to the KiloGUI application.

Kilobot controller At the beginning of the experiments, Kilobots search for a site by performing a random walk (parameters: $\mu_{\text{f}}^{\text{rw}} = 15$ s, $\sigma_{\text{f}}^{\text{rw}} = 0.5$ s, $\mu_{\text{r}}^{\text{rw}} = 0.5$ s, $\sigma_{\text{r}}^{\text{rw}} = 3$ s). The goal of this behavior is to move away the Kilobots from the nest, while they are looking for a site to forage from. As soon as a Kilobot finds a site, it explores the site for a duration of 60 s, emulating the collection of resources. After this period of time, the Kilobot returns to the nest by following the radial gradient while releasing pheromones. Once back to the nest, the Kilobot performs a random walk for a duration of 60 s which emulates the deposition of resources. Meanwhile, the Kilobot observes the distribution of pheromones within the nest and, at the end of this period, it uses this information to decide its new direction of motion. Whenever a Kilobot looking for a foraging site perceives the presence of pheromone, it stops performing the random walk and begins following a gradient formed by a combination of pheromones and radial navigation gradient. This behavior biases the motion of Kilobots toward a foraging site. In the case in which a Kilobot following a pheromone trail looses track of the trail, it resumes performing a random walk while looking for a foraging site.

Experiments We used a swarm of 50 Kilobots and performed five independent runs with a duration of 90 min each. The environment is characterized by a nest positioned in the center and two sites, the red site and the blue site, positioned on the left and the right side of the nest, respectively. Kilobots are initially located in the nest. The blue site is located 1.75 times farther from the nest than the red site, has a higher cost, and represents therefore a suboptimal foraging choice. Figure 10a shows the distribution of pheromones deposited in the environment which provides an indication of both the paths taken by Kilobots back and forth the nest as well as their collective choice for the red site (left side of Fig. 10a). Similarly, the spatial distribution of the Kilobots in the swarm shown in Fig. 10b has a higher density in the middle between the nest and the red site. Figure 10c confirms the collective preference for the swarm to forage from the red site by depicting the number of round trips between the nest and each of the two sites. Finally, Fig. 10d provides the details of the microscopic state of each Kilobot. Each horizontal line gives the microscopic state of one robot during an experiment: the yellow line represents a robot searching the environment for a foraging site; the red and blue lines represent, respectively, a robot moving from a site (respectively, red and blue sites) toward the nest. From this figure, it is possible to observe that each Kilobot requires a different amount of time to complete a round trip between the nest and a site as a results of both Kilobot collisions and, at times, poorly calibrated motors (represented by a few red and blue lines longer than average).

4 Discussion

Open hardware The Kilogrid is an open-source experimental environment available under a Creative Commons—Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0) license. The source and design files necessary to build the Kilogrid modules and the dispatcher, the documentation required to develop software for the Kilogrid, as well as the source code of the demos described in this paper can be accessed online from the Kilogrid website.[3]

[3] http://iridia.ulb.ac.be/kilogrid/.

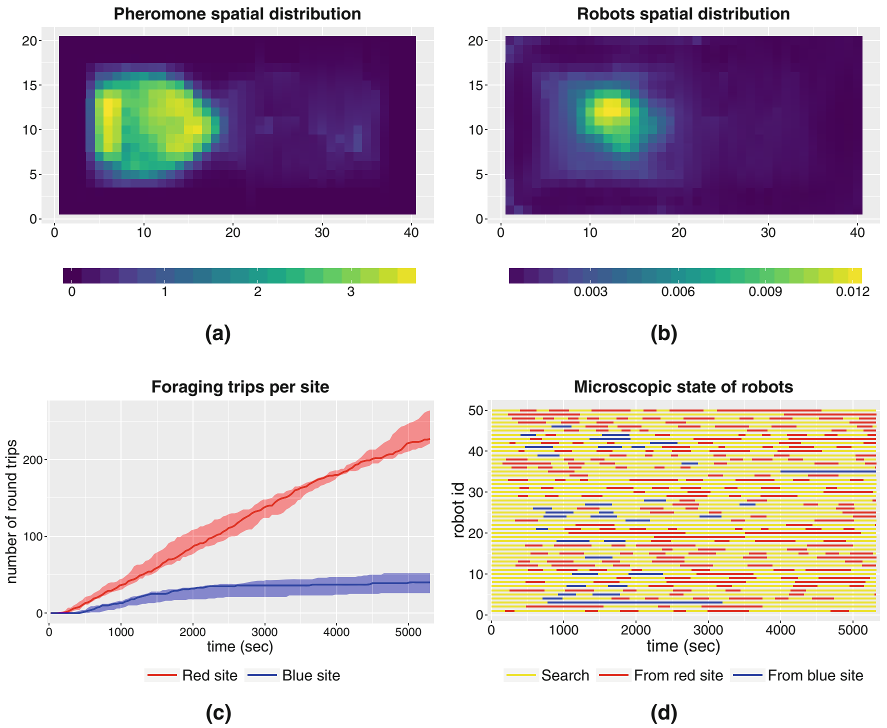


Fig. 10 Pheromone-based foraging with a swarm of 50 Kilobots: **a** cumulative spatial distribution of deposited pheromones (logarithmic scale); **b** spatial distribution of Kilobots in the environment; **c** number of foraging trips for each site; and **d** microscopic state of Kilobots (horizontal lines) during one experiment (yellow, search the environment for a foraging site; red and blue, return to the nest, respectively, from the red site and from the blue site) (Color figure online)

Table 1 Details of production costs for a Kilogrid of 200 modules

Category	Quantity	Total cost
Assembled SMD board for modules	200	€13467.34
Assembled SMD board for dispatcher	1	€372.09
Plexiglas material	–	€733.43
Wood material	–	€89.93
Cabling material	–	€200
Power supply	2	€75.24
Total cost		€14938.02
Total cost per module		€74.69

Production costs Table 1 provides the details of the production costs for a Kilogrid composed of 200 modules.[4] The largest share of the production costs for the Kilogrid is represented by the surface-mount device (SMD) boards for the modules and for the dispatcher. The cost for the SMD boards includes all components, the electronic boards, assembly of the

[4] Costs do not include taxes.

components, and the serigraphy of labels on both sides of the boards, and depends on the chosen manufacturer. In order to build the physical structure necessary to host the Kilogrid modules, we used a combination of laser-cut Plexiglas and medium-density fiberboard (MDF) wood. Finally, to power the Kilogrid and to connect together its modules we used 2 power supplies and cabling material. The overall cost of the Kilogrid presented and discussed in this paper amounts to €14938.02, for a cost per module of €74.69. In comparison, a swarm of 100 Kilobots, which is the typical size of a swarm used with the described Kilogrid, costs approximately €10042.08, that is, €100.42 per Kilobot.[5] The unit price of a Kilogrid module has therefore a cost of approximately 75% of that of a Kilobot. Notice, however, that, contrary to Kilobots purchased from K-Team Corporation, the Kilogrid used in this paper is a prototype and its cost does not take advantage from scale production.

Limitations The Kilogrid, as its name suggests, has a discretized nature with a resolution of 5 cm by 5 cm that is determined by the size of its cells. The virtualization of sensors and actuators is therefore bounded by this resolution. Additionally, the communication protocol between a cell and a Kilobot, which is the same as the communication protocol between a pair of Kilobots, does not guarantee the delivery of messages and allows for the exchange of only 2 messages per second. A consequence of these limitations is, for example, that the Kilogrid cannot provide a Kilobot with its orientation based on the information of a single cell (e.g., if the Kilobot is stuck against a border of the arena). To obtain a coarse-grained virtualization of an analogous bearing sensor, a Kilobot would need to integrate the coordinates of each perceived cell over time. Alternatively, the system could be extended to obtain a more precise virtual bearing sensor with the addition of an overhead camera coupled to the Kilogrid by means of the KiloGUI application.

Risk of abusing the Kilogrid The capabilities of the Kilogrid system have the potential to be misused by the experimenter. Indeed, the Kilogrid could be exploited to provide Kilobots with unrealistic information, for example, global information that is only available from an external observer or information from the virtualization of unrealizable hardware, which could lead to overestimate the performance of the considered swarm strategy. Nonetheless, the Kilogrid can potentially be used to mitigate this same issue because it provides the swarm robotics community with a standardized experimental environment for a commercially available robot. When researchers make their controllers publicly available, the Kilogrid can be used to enhance the reproducibility of the results and to facilitate the comparison of alternative strategies.

Kilogrid versus ARK, a pairwise comparison The Kilogrid system is not the only virtualization environment available for the Kilobot. As described in Sect. 1, the ARK system (Reina et al. 2017) has been released the year following our first publication about the Kilogrid (Antoun et al. 2016). Despite being both virtualization systems, they have been designed with different objectives in mind (see Table 2 for a summary). The Kilogrid is a spatially distributed environment where virtualization is implemented by means of localized messages conveyed by individual cells to the robots which in turn process this information onboard. In contrast, the ARK system is centralized, and it relies on global information extracted from visual tracking of the robots position and state and conveys off-board processed directives (e.g., direction of motion) by means of centralized broadcast messages. The Kilogrid is a modular system, and the production of most of its components can be outsourced (i.e., PCB boards, wooden and plexiglass parts). The assembly of the Kilogrid requires approximately

[5] The price of a Kilobot is computed from the price listed by K-Team Corporation and does not include taxes nor the cost of the OHC and of the Kilobot charger.

Table 2 Pairwise comparison between the Kilogrid system and the ARK system

Feature	Kilogrid	ARK
Virtualization	Spatially distributed	Centralized
	Onboard	Off-board
Design	Modular	Monolithic
IR communication	Bidirectional	Unidirectional
	Kilogrid $\leftrightarrow$ Kilobot	ARK $\rightarrow$ Kilobot
IR communication bandwidth (system $\rightarrow$ Kilobot)	72 bits every 0.5 s per cell	24 bits every 2.5 s
IR communication bandwidth (Kilobot $\rightarrow$ system)	72 bits every 2 s per cell	Not available
Scalability	Scalable	Not scalable
Cost	€74.69/module	€630
	€6733.67/m^2	
Granularity	Discrete (5 cm by 5 cm)	Continuous
Robot orientation	Not available	Available
Automatic calibration	Not available	Available

2 working days and, thanks to its modularity, it is based on the repetition of a few assembly steps. In contrast, the ARK system is monolithic, being composed of cameras and of an extended version of the original Kilobot OHC. The Kilogrid offers a spatially distributed bidirectional communication channel where each cell can communicate information to the robots with a bandwidth of 72 bits every 0.5 s and receive information from the robots with a bandwidth of 72 bits every 2 s.[6] In contrast, the ARK system has an effective bandwidth that depends on the number of robots, i.e., the bandwidth per robot decreases as the number of robots increases. For a swarm of 150 robots, it can provide only 24 bits every 2.5 s of which 10 bits are dedicated to the robot identifier which is not required by the locally broadcast Kilogrid messages. A consequence of this design choice is that, as opposed to the Kilogrid, the ARK system is not scalable. The cost of the Kilogrid is of approximately €75 per module and €6730 per square meter—approximately the same as the cost of the Kilobots operating on the same surface. The full cost of the entire ARK system is not publicly available; however, the extended OHC costs approximately €125 and the four cameras have an estimated price of €225. In addition to these costs, a user of the ARK system needs to buy a CUDA-enabled GPU card (approximately €280). The Kilogrid is discrete in nature with a resolution of 5 cm by 5 cm, while the ARK system is continuous. Although not implemented yet, the Kilogrid can be used to approximate a discrete orientation of each robot (8 possible directions) by having modules interpolate the position of the robots over time. In contrast, the ARK system can detect the orientation of a robot from individual camera frames and relies on interpolation between frames to improve accuracy. Finally, the Kilogrid does not provide automatic calibration of the Kilobots motors and this needs therefore to be performed manually. In contrast, the ARK system provides this feature. However, the authors of the ARK system acknowledge the difficulty of this activity [cf. Conclusions in Antoun et al. (2016)] and do not provide details about its implementation, usage, and performance such as calibration time or quality of calibrated motion.

[6] If communication between Kilobots is not required, that from a Kilobot to a cell can be augmented up to 72 bits every 0.5 s.

5 Conclusions

This paper accompanies the open-source release of the Kilogrid, a modular and distributed virtualization and data logging system designed to enhance the quality of experiments with the Kilobot robot. The Kilogrid can be used to virtualize sensors and actuators and to systematize the collection of data during an experiment. We described the design and the functioning of the Kilogrid system and demonstrated its capabilities by means of four different robot experiments: exploration and obstacle avoidance, site selection based on multiple gradients, plant watering, and pheromone-based foraging. By doing so, we showed not only how the task of collecting the results of robot experiments with the Kilobots is greatly simplified, but also how certain types of analysis are made possible by means of a distributed, duplex communication channel (see Fig. 8c–e). Finally, we discussed the production cost, the open-source availability, and the limitations and the possible misuses of the Kilogrid system.

Acknowledgements This work was supported by the European Research Council through the ERC Advanced Grant "E-SWARM: Engineering Swarm Intelligence Systems" (contract 246939) to Marco Dorigo. Marco Dorigo acknowledges support from the Belgian F.R.S.-FNRS, for which he is a Research Director.

References

Antoun, A., Valentini, G., Hocquard, E., Wiandt, B., Trianni, V., & Dorigo, M. (2016). 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 Press.

Arvin, F., Krajník, T., Turgut, A. E., & Yue, S. (2015a). COSϕ: Artificial pheromone system for robotic swarms research. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 407–412).

Arvin, F., Xiong, C., & Yue, S. (2015b). Colias-ϕ: An autonomous micro robot for artificial pheromone communication. *International Journal of Mechanical Engineering and Robotics Research*, *4*(4), 349–353.

Becker, A., Habibi, G., Werfel, J., Rubenstein, M., & McLurkin, J. (2013). Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In *2013 IEEE/RSJ international conference on intelligent robots and systems* (pp. 520–527).

Beckers, R., Holland, O.E., & Deneubourg, J. L. (1994). From local actions to global tasks: Stigmergy and collective robotics. In *Artificial life IV* (pp. 181–189). MIT Press.

Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, *7*(1), 1–41.

Brutschy, A., Garattoni, L., Brambilla, M., Francesca, G., Pini, G., Dorigo, M., et al. (2015). The TAM: Abstracting complex tasks in swarm robotics research. *Swarm Intelligence*, *9*(1), 1–22.

Dorigo, M., Birattari, M., & Brambilla, M. (2014). Swarm robotics. *Scholarpedia*, *9*(1), 1463.

Garnier, S., Combe, M., Jost, C., & Theraulaz, G. (2013). Do ants need to estimate the geometrical properties of trail bifurcations to find an efficient route? A swarm robotics test bed. *PLoS Computational Biology*, *9*(3), 1–12.

Gauci, M., Radhika, N., & Rubenstein, M. (2017). *Distributed autonomous robotic systems: The 13th international symposium*. Springer, chap Programmable self-disassembly for shape formation in large-scale robot collectives (**in press**).

Goss, S., Aron, S., Deneubourg, J. L., & Pasteels, J. M. (1989). Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, *76*(12), 579–581.

Gutiérrez, Á., Campo, A., Dorigo, M., Amor, D., Magdalena, L., & Monasterio-Huelin, F. (2008). An open localization and local communication embodied sensor. *Sensors*, *8*(11), 7545–7563.

Khaliq, A. A., Di Rocco, M., & Saffiotti, A. (2014). Stigmergic algorithms for multiple minimalistic robots on an RFID floor. *Swarm Intelligence*, *8*(3), 199–225.

Khaliq, A. A., & Saffiotti, A. (2015). Stigmergy at work: Planning and navigation for a service robot on an RFID floor. In IEEE *International Conference on Robotics and Automation, ICRA 2015*, (pp. 1085–1092). IEEE Press.

Liu, W., & Winfield, A. F. (2011). Open-hardware e-puck linux extension board for experimental swarm robotics research. *Microprocessors and Microsystems*, *35*(1), 60–67.

Mathews, N., Valentini, G., Christensen, A. L., O'Grady, R., Brutschy, A., & Dorigo, M. (2015). Spatially targeted communication in decentralized multirobot systems. *Autonomous Robots*, *38*(4), 439–457.

Melhuish, C., Welsby, J., & Edwards, C. (1999). Using templates for defensive wall building with autonomous mobile ant-like robots. In *Proceedings of towards intelligent mobile robots (TIMR99)*, Vol. 99.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., & Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In P. J. S. Gonçalves, P. J. D. Torres & C. M. O. Alves (Eds.), *Proceedings of the 9th conference on autonomous robot systems and competitions* (Vol. 1, pp. 59–65). IPCB: Instituto Politécnico de Castelo Branco.

Payton, D., Daily, M., Estowski, R., Howard, M., & Lee, C. (2001). Pheromone robotics. *Autonomous Robots*, *11*(3), 319–324.

Pickem, D., Wang, L., Glotfelter, P., Diaz-Mercado, Y., Mote, M., Ames, A.D., Feron, E., & Egerstedt, M. (2016). Safe, remote-access swarm robotics research on the robotarium. arXiv:1604.00640.

Reina, A., Cope, A. J., Nikolaidis, E., Marshall, J. A. R., & Sabo, C. (2017). ARK: Augmented reality for kilobots. *IEEE Robotics and Automation Letters*, *2*(3), 1755–1761.

Reina, A., Salvaro, M., Francesca, G., Garattoni, L., Pinciroli, C., Dorigo, M., & Birattari, M. (2015). 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 Press.

Rubenstein, M., Ahler, C., Hoff, N., Cabrera, A., & Nagpal, R. (2014a). Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robotics and Autonomous Systems*, *62*(7), 966–975.

Rubenstein, M., Cabrera, A., Werfel, J., Habibi, G., McLurkin, J., & Nagpal, R. (2013). Collective transport of complex objects by simple robots: Theory and experiments. In T. Ito, C. Jonker, M. Gini & O. Shehory (Eds.), *Proceedings of the 12th international conference on autonomous agents and multiagent systems, IFAAMAS, AAMAS '13* (pp. 47–54).

Rubenstein, M., Cornejo, A., & Nagpal, R. (2014b). Programmable self-assembly in a thousand-robot swarm. *Science*, *345*(6198), 795–799.

Soorati, M. D., & Hamann, H. (2016). Robot self-assembly as adaptive growth process: Collective selection of seed position and self-organizing tree-structures. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 5745–5750).

Støy, K. (2001). Using situated communication in distributed autonomous mobile robotics. In *Proceedings of the 7th scandinavian conference on artificial intelligence, SCAI'01* (Vol. 1, pp. 44–52). IOS Press.

Trianni, V., De Simone, D., Reina, A., & Baronchelli, A. (2016). Emergence of consensus in a multi-robot network: From abstract models to empirical validation. *IEEE Robotics and Automation Letters*, *1*(1), 348–353.

Valentini, G., Ferrante, E., & Dorigo, M. (2017). The best-of-n problem in robot swarms: Formalization, state of the art, and novel perspectives. *Frontiers in Robotics and AI*, *4*, 9.

Valentini, G., Ferrante, E., Hamann, H., & Dorigo, M. (2016). Collective decision with 100 Kilobots: Speed versus accuracy in binary discrimination problems. *Autonomous Agents and Multi-Agent Systems*, *30*(3), 553–580.

Valentini, G., Hamann, H., & Dorigo, M. (2015). Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off. In R. Bordini, E. Elkind, G. Weiss & P. Yolum (Eds.), *Proceedings of the 14th international conference on autonomous agents and multiagent systems, IFAAMAS, AAMAS '15* (pp. 1305–1314).

Vigelius, M., Meyer, B., & Pascoe, G. (2014). Multiscale modelling and analysis of collective decision making in swarm robotics. *PLoS ONE*, *9*(11), e111542.

Werfel, J. (2012). Collective construction with robot swarms. In R. Doursat, H. Sayama, & O. Michel (Eds.), *Morphogenetic engineering: Toward programmable complex systems* (pp. 115–140). Berlin: Springer.