Simulating Kilobots Within ARGoS: Models and Experimental Validation

Carlo Pinciroli[1](✉), Mohamed S. Talamali[2], Andreagiovanni Reina[2], James A. R. Marshall[2], and Vito Trianni[3]

[1] Robotics Engineering, Worcester Polytechnic Institute, Worcester, MA, USA
cpinciroli@wpi.edu

[2] Department of Computer Science, University of Sheffield, Sheffield, UK
{mstalamali1,a.reina,james.marshall}@sheffield.ac.uk

[3] ISTC, National Research Council, Rome, Italy
vito.trianni@istc.cnr.it

Abstract. The Kilobot is a popular platform for swarm robotics research due to its low cost and ease of manufacturing. Despite this, the effort to bootstrap the design of new behaviours and the time necessary to develop and debug new behaviours is considerable. To make this process less burdensome, high-performing and flexible simulation tools are important. In this paper, we present a plugin for the ARGoS simulator designed to simplify and accelerate experimentation with Kilobots. First, the plugin supports cross-compiling against the real robot platform, removing the need to translate algorithms across different languages. Second, it is highly configurable to match the real robot behaviour. Third, it is fast and allows running simulations with several hundreds of Kilobots in a fraction of real time. We present the design choices that drove our work and report on experiments with physical robots performed to validate simulated behaviours.

1 Introduction

Simulators are key tools for swarm robotics research. Many studies were performed mainly (when not exclusively) in simulation [3,7]. Swarm simulations tend to be "minimalistic", in that they include only few relevant features of the robots. Often, robots are modeled as abstract particle-like agents. For instance, a simulator often used for swarm robotics research is MASON [14], specifically developed for multi-agent systems research and not tailored to represent any specific robotic platform. This kind of simulations are useful to prove the validity of decentralised coordination algorithms, but fall short when physical interactions need to be taken into account, e.g., to simulate pulling and pushing forces among robots [23]. Simulations in this case need to move beyond simple kinematics and include the full dynamics of modern rigid-body simulation engines [16,17,19].

When a reference robotic hardware is available, simulations need to account for all its components, including sensors, actuators and communication devices. These can make simulations very costly in terms of computational requirements,

© Springer Nature Switzerland AG 2018
M. Dorigo et al. (Eds.): ANTS 2018, LNCS 11172, pp. 176–187, 2018.
https://doi.org/10.1007/978-3-030-00533-7_14

placing a tradeoff between the accuracy of the simulation and its speed [16]. However, even with accurate simulations, the "reality gap" cannot be completely filled [11]. This is especially apparent when using automatic design techniques that might exploit the idiosyncrasies of the simulated experimental setup [7,8]. To support swarm robotics experimentation and minimise errors when moving to real-world experiments, simulations not only need to implement techniques that reduce the reality gap (e.g., sensor sampling [15] and introduction and configurability of noise [10]), but should also provide cross-compiling solutions to directly reuse the control software developed for simulation also with the real robot hardware. In this way, any issue related to translating the algorithm from the one to the other platform can be removed.

Among the simulators developed for swarm robotics research, ARGoS [17] offers a number of desirable features. ARGoS has a modular design conceived to allow the user to select which aspects of the simulation should be assigned more computational resources (thus increasing accuracy) and which aspects can be simulated coarsely (thus improving scalability). For example, ARGoS supports the use of different types of physics engine—from simple kinematics to fully 3D dynamics. In addition, ARGoS is designed for parallel computation, an important feature to fully exploit the computational power of modern multi-core machines. Several robotic platforms, both custom and off-the-shelf, are currently available in ARGoS either as part of the core package or as extensions.

One robotic platform currently having momentum in swarm robotics is the Kilobot [20,22]. The Kilobot design is driven by the need for low cost (to allow for production of a thousand robots), small size (to fit the spaces of a typical research lab), robustness (to reduce faults), and ease of use [20]. Meeting these design goals meant sacrificing important features for swarm robotics research, such as accurate environmental sensing and remote access to the robot state. To compensate for missing features, devices such as the *Kilogrid* [24] and the *ARK* virtualisation environment [18] have been proposed, greatly expanding the realm of experimental activities attainable with Kilobots.

Despite the success of the Kilobot as an experimental platform, a fast and accurate simulation environment is still important to enable fast design cycles and educational activities based on the Kilobot. In the recent past, several simulators have appeared that include the Kilobot platform. Among these, it is worth mentioning V-REP [19], KBSim [9], and Kilombo [12]. V-REP is a generic, modular framework originally designed for complex 3D simulations of robotic arms and mobile robots. V-REP simulations tend to be very accurate, at the cost of long run-times when hundreds of robots are involved in the simulation. KBSim and Kilombo are designed to provide fast simulations for large numbers of robots. Kilombo, in particular, can perform simulations involving thousands of Kilobots hundreds of times faster than real time. Both KBSim and Kilombo, however, achieve scalability by drastically simplifying the motion and communication models of the robots, and proper validation of these models is currently unavailable. Another important aspect is that all these platform impose limitations when one is to transfer code developed in simulation onto real platforms.

V-REP and KBSim do not target the Kilobot API, and thus require a complete rewrite of the code. This is likely to introduce bugs and it makes convincing model validation hard to achieve. Kilombo, on the other hand, achieves direct interoperability with physical Kilobots by modifying the original Kilobot API and imposing limitations on how the code can be written. Specifically, users are not allowed to use global variables and must resort to conditional compilation techniques to transfer the code successfully. Modifying the Kilobot API also means that future improvements and bug fixes must be ported to the Kilombo version, if compatibility is to be preserved.

In this paper, we present a plugin for the ARGoS simulator [17] that offers accurate models, scalability sufficient to run one-thousand-robot experiments in real-time, and full compatibility with the original Kilobot API. We present the Kilobot and the reference behaviour in Sect. 2, along with experiments conducted to determine the real-world behaviour and extract features to be implemented in simulation. In Sect. 3, we validate the simulations against the real-world behaviour in representative experimental conditions. We demonstrate that ARGoS closely predicts the Kilobot behaviour, and offers sufficient efficiency to run large-scale simulations in a fraction of real-time. Section 4 concludes the paper.

2 Kilobot: Reference Behaviour and Simulation Models

The Kilobot is a small autonomous robot with a circular shape (diameter: 3.3 cm) standing on three rigid legs, which moves thanks to two vibrational motors and a slip-stick locomotion principle [20]. The robot is provided with infrared transceivers for communication and range detection of neighbours, as well as with an ambient light sensor and a coloured LED for displaying the robot state. In the following, we detail the simulation design for the various components.

2.1 Body Model

The Kilobot is simulated as a small cylinder with the same radius as the real robot, resting on three thin cylinders representing the legs. When the simulation is performed with ARGoS's 2D physics engine, the robot body is the circular projection of the main board on the plane. The Kilobot locomotion is provided by the two vibrational motors, which implement a classical differential-drive model. The forward and rotational speeds are determined by the difference between the velocity v_ℓ and v_r, resulting respectively from the left and right motor activation.

Although the motors can be finely controlled, the normal practice with Kilobots is to use only 3 motion modes: straight motion, clockwise and counterclockwise rotation. We calibrated the parameters of the ARGoS model so as to result in a forward speed of $1\,{}^{\text{cm}}/_{\text{s}}$ and a rotation speed of $45\,{}^{\circ}/_{\text{s}}$, corresponding to the nominal speed of real Kilobots [20].

2.2 Noise and Inter-individual Variations

Real Kilobots have strong inter-individual variations, which are due to the simple design and the slip-stick locomotion system. This system is strongly dependent on small variations in the position of the motors and the bending of the legs. To obtain acceptable locomotion, calibration of individual robots is necessary to find good values for the motor parameters that provide straight motion and rotations at the desired speed. Most researchers rely on manual calibration, but this process is cumbersome and dependent on the surface on which the Kilobot maneuver. The ARK platform [18] enables the parallel calibration of tens of robots. While ARK shortens calibration time, it cannot guarantee error-free precision due to the intrinsically high noise of Kilobot motion. As a result, through either manual or automatic calibration, Kilobots hardly move straight or rotate at the nominal speed, and present strong inter-individual variations.

To capture this, the Kilobot model has a noise component for the motion of a single robot. Given the nominal left (ℓ) and right (r) speeds v_i (with $i \in \{\ell, r\}$) of the differential drive model, the actual speed $\hat{v}_i$ of each wheel is computed as $\hat{v}_i = f_i(v_i + b_i)$, where f_i is a per-step actuation noise and b_i is a per-robot bias added to the nominal speed v_i. Both f_i and b_i are Gaussian-distributed random parameters, with mean and standard deviation defined in the experiment configuration file. For each robot, the bias b_i, with $i \in \{\ell, r\}$, are drawn from the specified Gaussian distributions at the beginning of the experiment. Instead, the actuation noise f_i is drawn at each time-step.

The ARGoS default values are set from measurements performed on a sample of 120 real Kilobots. The nominal (noise-free) speed is set to a forward speed of $v_i \simeq 1\,\mathrm{cm/s}$ and a rotation speed of $\sim 45\,°/\mathrm{s}$ (i.e. $v_\ell \simeq 2\,\mathrm{cm/s}$, $v_r = 0$ for right rotation and viceversa for left rotation). To determine the distribution of the noise observed in reality, we conducted experiments on 120 different Kilobots (in batches of 6) that have been previously calibrated (60 manually and 60 automatically—we could not notice any remarkable difference). Robots were asked to move straight for 1 min. Through ARK, we recorded the trajectory of each robot, we derived the robot displacement every 10 s, and through a differential drive model we computed the left and right speeds $\hat{v}_i$ (with $i \in \{\ell, r\}$). Through $\hat{v}_i$, we could compute the bias $b_i^t = \hat{v}_i - v_i$ (ignoring white noise, i.e. $f_i = 1$) for each 10 s motion trajectory (thus $t \in \{1, ..., 6\}$ for our 60 s experiments). Finally, we computed the average bias $b_i = \sum b_i^t / 6$ for each robot and we report the distributions of biases (for both left/right velocities $i \in \{\ell, r\}$) of the 120 tested Kilobots in Fig. 1(a). From this distribution, we computed the mean $\mu_b = 0.015\,\mathrm{mm/s}$ and the standard deviation $\sigma_b = 1.86\,\mathrm{mm/s}$ which we use as default values in ARGoS. Figure 1(b) shows a comparison between the mean square displacement (MSD) of the 120 Kilobots and the simulated robots. Noise-free simulations show the robots moving (as expected) at the nominal speed of $\sim 1\,\mathrm{cm/s}$, whereas the default noise values show that the simulated robots have motion dynamics remarkably similar to reality.

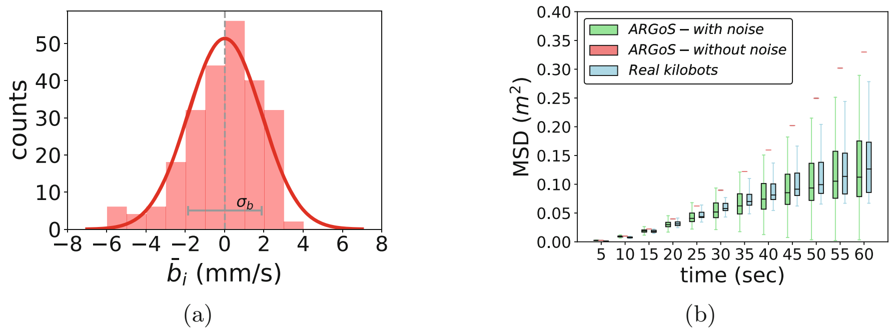


Fig. 1. (a) Distribution of the bias in straight motion estimated from measurements over 120 different Kilobots which have been previously calibrated (60 manually, 60 automatically). The bars considers 12 bins in the range $[-6, 6]$ mm. The solid red line shows the approximated Gaussian distribution $\mathcal{N}_b(\mu_b, \sigma_b)$ (with mean μ_b and standard deviation σ_b) used in the default configuration of ARGoS. (b) Comparison between simulation (600 robots) and reality (120 robots) in term of MSD when robots are asked to go straight for 1 min. The default noise values of ARGoS give an accurate match between reality and simulation.

2.3 Robot-Robot Communication

Communication among the Kilobots is implemented exploiting the infrared transceiver positioned under the robot body, which sends a modulated infrared signal that bounces on the ground and can be perceived by neighbours within a distance of about 15 cm. The communication protocol implemented in the most recent firmware (see https://www.kilobotics.com) is Carrier Sense Multiple Access with Collision Detection (CSMA-CD) with exponential backoff, meaning that upon detection of the occupied channel, message sending is delayed within an exponentially increasing range of time slots. To avoid interferences, the maximum transmission frequency is set to 2 Hz. Nevertheless, Kilobots may find the channel busy when transmitting, and collisions can still occur. To evaluate the impact of concurrent communication, we performed an experiment with 25 Kilobots packed in a 5×5 square formation—their bodies touching each other—and attempting to transmit messages at maximum rate. Messages contained just the ID of the sender, so that robots getting a message could update the number of messages received from each other robot. At the same time, robots stored also the number of messages successfully transmitted. We performed 10 independent runs, and found that the transmission probability—i.e., the ratio between successfully transmitted messages between the packed robots and solitary robots—was close to maximum (0.992 ± 0.002) and independent from the position of the Kilobot in the 5×5 formation. On the reception side, we observed that the probability of receiving a messages depended strongly on the position of the receiving robot (see Fig. 2). Generally speaking, robots in the periphery where affected less by interferences than robots in the center (see the left panel in Fig. 2 showing the reception probability of each robot with respect to all other

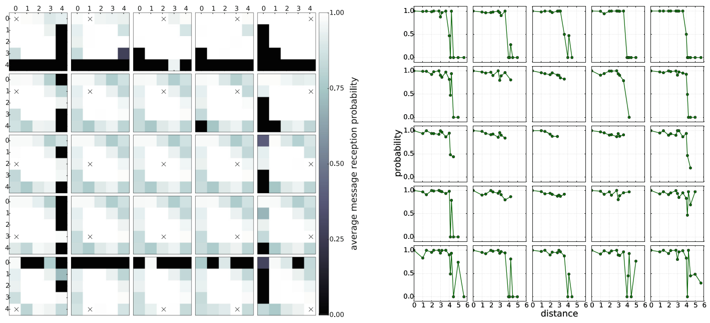


Fig. 2. Communication interference measured on 25 Kilobots packed in a 5×5 grid and concurrently sending messages to each other. Left: average message reception probability for each robot in the grid from any other robot. For each panel, the position of the receiving robot on the grid is indicated by a $\times$. Right: average message reception probability with respect to distance, in body lengths, for every Kilobot in the grid.

robots). Additionally, the decrease of the probability of reception is more pronounced for center robots, indicating a stronger effect from interferences (see the right panel in Fig. 2). Indeed, collisions are more probable in the center, where robots may receive at the same time messages sent by robots at the periphery that do not sense each other.

According to these results, we have implemented the Kilobot communication limiting the maximum transmission frequency to 2 Hz, including a configurable small error on the transmission side, and modelling message collision on the receiver side with a tuneable probability when two robots happen to be concurrently transmitting.

2.4 Light Sensor

Real Kilobots are provided with a photodiode sensor to detect the ambient light. In the commercial version[1], this sensor is placed looking upward on the robot body. Simulating the light sensors is highly dependent on the type and position of the ambient light. ARGoS natively offers light sensor models and can simulate light sources with a tuneable intensity. The sensor readings decrease quadratically with the distance from the lights. In the Kilobot plugin, we rescaled the readings in the range typical of the real robot.

2.5 ARK Simulation

The Augmented Reality for Kilobots system (ARK) [18] overcomes the limitations of Kilobots by providing a flexible set of virtual environments, with which

[1] Kilobots are open-hardware and in Europe are produced and sold by K-Team Corporation (see https://www.k-team.com).

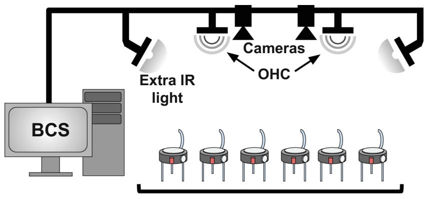


Fig. 3. Graphical representation of the ARK architecture. See more details in [18].

Kilobots can interact through a variety of user-configurable sensors and actuators. ARK comprises a base station interfaced with an array of cameras, and an array of overhead controllers (OHCs) for transmitting IR messages to Kilobots (see the system architecture in Fig. 3). Communication with the Kilobots during an experiment is obtained via broadcast of addressed messages, with each broadcast packet containing three *ARK messages* (3-bytes long) for different Kilobots. Kilobot addresses (10 bits) are uniquely assigned by ARK during a pre-experimental phase. For sensor readings, the other 14 bits of data can be assigned to multiple virtual sensors as desired by the user. Location-specific information from the virtual environment can be determined for a Kilobot by its physical position at the time the message is to be sent, which is determined through robust tracking of each ID-assigned Kilobot over the duration of an experiment. Kilobots communicate virtual actuator commands via signalling through RGB LEDs, which are received via the base station's camera array and translated into operations on the virtual environment. Each augmented reality experiment can be composed of more than one virtual environments. Each environment has user-defined structure and spatio-temporal dynamics, as exemplified in [6]. As well as enabling richer experimental paradigms, ARK's features also lend it to automatic motor calibration and other house-keeping features [18].

ARK is integrated with ARGoS through the ARK Loop Function (ALF), the simulated counterpart of the ARK's base control station. The ALF is executed every ARGoS time-step and is in charge of simulating the virtual environments and of sending IR ARK messages to the simulated Kilobots. To facilitate the transfer from simulation to reality, the ALF uses the same method names and structure of its real counterpart. Similarly to the ARK's base control station, the ALF has real-time access to the state of the simulated Kilobots, i.e., their position, orientation, and LED colour. This information can be used by the user to code the functioning of the virtual actuators and sensors. The virtual actuators update the virtual environments, and virtual sensor readings are computed using the Kilobot's state, then transmitted to the robot. The ALF automatically codes the 3-byte ARK messages within standard 9-byte Kilobot messages in the same way ARK does. Therefore the Kilobot control software needs to decode the ARK messages in ARGoS in the same way it does in reality. This implementation choice is particularly helpful because it allows for the use of identical code in simulation and reality. ALF gives the user the possibility to limit the communication to a maximum frequency of 60 ARK messages per second (to

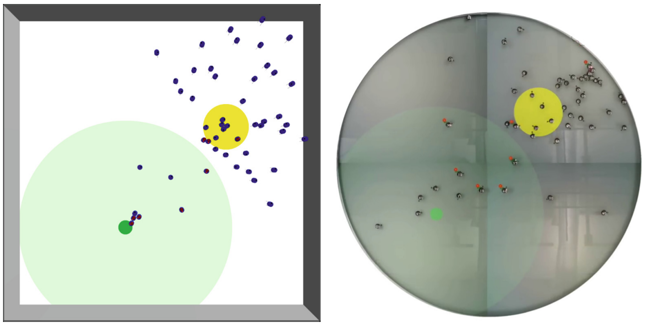


Fig. 4. Two screenshots of the same experiment in simulation (left) and reality (right). We (re)implemented the Demo C from [18] in which 50 Kilobots sense and modify two virtual environments. The full video is available at https://youtu.be/kioZR99hnU4.

match the real ARK's frequency) or to simulate an unlimited ARK message frequency.

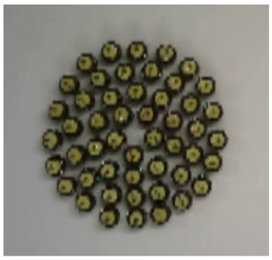
To showcase the ALF functioning, we reproduced a simulated version of one experiment based on ARK, the Demo C of [18]. Figure 4 shows two screenshots of the experiment in simulation (left) and reality (right) featuring 50 Kilobots that operate in two virtual environments (flower field and nest).

3 Experimental Validation

We run a set of experiments to assess the reliability of ARGoS in simulating Kilobot swarms and to compare the ARGoS performance with the existing Kilobot simulator Kilombo [12]. In Sect. 3.1, we tested how reliably ARGoS and Kilombo simulate physical interactions between Kilobots through a random diffusion experiment. In Sect. 3.2, we compared the simulation speed of the two simulators. Finally, in Sect. 3.3, we show that ARGoS successfully simulates physical interactions between the robots and physical objects (e.g. a box) and that force factors are taken into consideration in the simulation.

3.1 Random Diffusion Experiment

Kilobots are equipped with minimal sensing capabilities which do not allow robots to implement robust mechanisms of collision avoidance, therefore collisions between robots and objects in the environment are frequent. In this experiment, we assess how realistically the collisions between Kilobots are simulated in ARGoS and Kilombo. To perform this study, we designed an experiment that maximises the number of collisions in a repeatable setup. The 50 Kilobots have an initial compact distribution as illustrated in Fig. 5 (left). The robots are placed in concentric circles heading toward the centre of the group (more precisely, the robot are placed on the vertices of four concentric regular polygons,



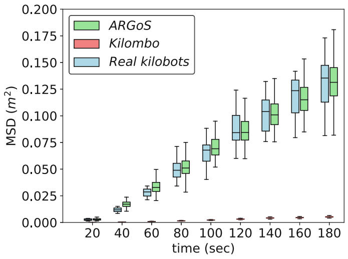


Fig. 5. (left) Initial distribution in four concentric circles with all 50 Kilobots facing towards the centre. (right) Comparison between real 50 Kilobots (19 runs) and 50 simulated Kilobots (100 runs) in ARGoS and Kilombo. We show the average mean square displacement (MSD) in a highly dense environment. ARGoS shows a good agreement with reality, whereas Kilombo does not. Video footage is available at https://youtu.be/6HYti0ABuxc.

starting from the innermost regular pentagon with radius 35 mm, each polygon has the same centre, twice the number of vertices, and double radius of its internal polygon). In this experiment, the 50 Kilobots perform an isotropic random walk [5] through which they repetitively move forward for ~10s and turn in a random direction (left or right) for a random time drawn from a uniform distribution $\mathcal{U}(0,4)$s. We performed 19 runs of this experiment with real robots and 100 runs with simulated Kilobots in both ARGoS and Kilombo. For every run, we recorded the trajectory of each Kilobot for a period of 3 min to compute the mean square displacement (MSD) of each robot. We combined the 50 MSD in each experiment and we show in Fig. 5 (right) how the average MSD changes over time. Figure 5 (right) clearly shows that ARGoS correctly simulates physical interactions between robots while Kilombo does not.

3.2 Speed and Scalability

To test the scalability of the Kilobot plugin, we performed experiments based on the `disperse.c` example provided on the Kilobot website. This behaviour allows a group of Kilobots to evenly disperse in the environment. We deemed `disperse.c` a good benchmark because it involves both motion and communication, and the robots are initially deployed in a tight cluster that stress-tests the collision management of ARGoS's 2D physics engine.

Scalability is measured through the *wall clock time*, which is the real time elapsed between the beginning and the end of the simulation of 60 virtual seconds. We performed these experiments on a node of a computing cluster with 48 Intel Xeon Platinum 8168 CPUs at 2.70 GHz. The experiments were performed without graphical visualisation.

We considered several parameters: (i) the number of robots $N \in \{10, 100, 1000\}$; (ii) the number of "worker" threads used by ARGoS $T \in \{0, 4, 16\}$; and

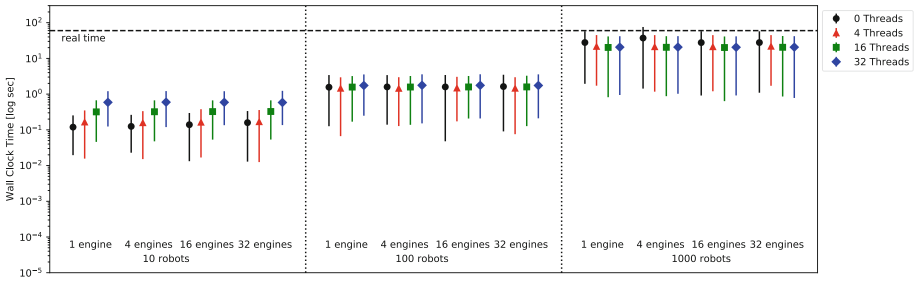
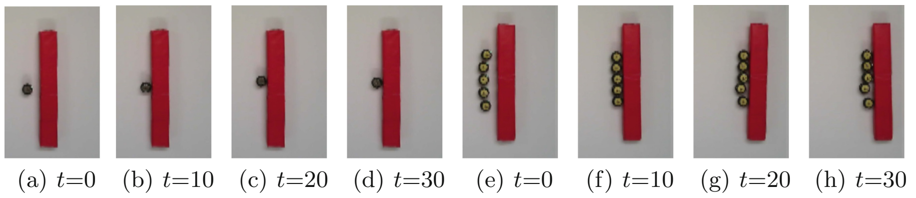


Fig. 6. Scalability experiments. We report the median, max and min wall clock time in a log sec scale for experiments simulating 60 virtual seconds.



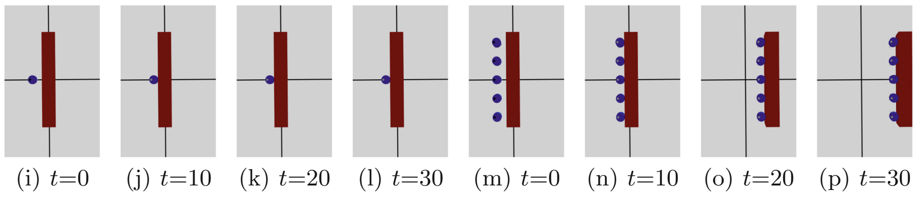


Fig. 7. Box-pushing experiments. Top: real robots. Bottom: ARGoS simulation. Time in seconds. Video footage at https://youtu.be/fwL9ePWttiU.

(iii) the number of physics engines in which the environment is partitioned $P \in \{1, 4, 16\}$. Every parameter set $\langle N, T, P \rangle$ was tested 30 times.

The results are reported in Fig. 6. The use of multiple threads and physics engines is beneficial when large swarms are simulated—particularly with 1000 robots. Conversely, with small swarms of 10 robots, multi-threading and multiple physics engines decrease performance. With 1000 robots, 1 physics engine and 0 working threads, the runtime mean was 27 s (45% of real time); when 16 physics engines and 16 working threads were used, the simulation was completed in 20 s (33% of real time).

3.3 Accuracy: Box Pushing

Remarkable collective transport experiments have been demonstrated with the Kilobots [1,21]. Our plugin also supports simulations that involve robots that push objects. In our experiments, we found that one robot pushing an 18 g box is not sufficient to move the box. Rather, it is necessary to use at least 5 robots to exert sufficient force to move the box. Figure 7 shows a real-world box pushing experiment and its simulated counterpart.

4 Conclusions

An essential feature of any simulator are usability and realism, and scalability. Kilobots, despite their simple hardware design, present specificities that make their simulation non-trivial. As a matter of fact, most of the simulators available for the Kilobots are rather minimalistic, and prove useful only for proof-of-concept studies without guarantees of respecting real-world behaviour. Our effort in developing a Kilobot model for ARGoS fills the need of a usable and reliable simulation, as demonstrated by the validation experiments we performed. We believe this will be a precious tool for swarm robotics research. To this end, the simulator is released open source for the benefit of the community (available at https://github.com/ilpincy/argos3-kilobot).

Besides the Kilobot simulation, this paper also introduces methodologies for tuning the simulation that can be replicated whenever a close matching between simulation and reality is desired. In particular, inter-individual variation between Kilobots (due to differences in hardware and calibration) needs to be considered as it represent an important problem when moving from simulation to reality. Tuning the simulation as discussed in Sect. 2.2 brings the full complexity of real-world Kilobots into the simulations, allowing for the design of controllers that are robust to inter-individual variability and that bridge the “reality gap”.

Further support to swarm robotics research with Kilobots can be obtained by automatisation of practices that are now performed manually. The ARK offers tools to this aim, and the integration of ARK within ARGoS is useful to streamline experimentation. A stronger integration of simulated and real robots can be performed through ARK, letting simulated and real Kilobots run in parallel, therefore opening the way for online learning [13], self-modelling [2] or embodied evolution [4].

Acknowledgments. This work was partially supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme under Grant 647704 to James Marshall. Vito Trianni acknowledges support from the project DICE (FP7 Marie Curie Career Integration Grant, ID: 631297). The authors thank Alex Cope for assistance in the preparation of Fig. 3.

References

1. Becker, A., Habibi, G., Werfel, J., Rubenstein, M., McLurkin, J.: 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 (IROS), pp. 520–527. IEEE (2013)
2. Bongard, J., Zykov, V., Lipson, H.: Resilient machines through continuous self-modeling. Science **314**(5802), 1118–1121 (2006)
3. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intell. **7**(1), 1–41 (2013)
4. Bredeche, N., Haasdijk, E., Prieto, A.: Embodied evolution in collective robotics: a review. Front. Robot. AI **5**, 12 (2018)

5. Dimidov, C., Oriolo, G., Trianni, V.: Random walks in swarm robotics: an experiment with kilobots. In: Dorigo, M. (ed.) ANTS 2016. LNCS, vol. 9882, pp. 185–196. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44427-7_16
6. Font Llenas, A., Talamali, M.S., Xu, X., Marshall, J.A.R., Reina, A.: Quality-sensitive foraging by a robot swarm through virtual pheromone trails. In: Dorigo, M., et al. (ed.) Swarm Intelligence (ANTS 2018), LNCS, vol. 11172, pp. X-XY. Springer, Heidelberg (2018). In press
7. Francesca, G., Birattari, M.: Automatic design of robot swarms: achievements and challenges. Front. Robot. AI **3**, 224–9 (2016)
8. Francesca, G., Brambilla, M., Brutschy, A., Trianni, V., Birattari, M.: AutoMoDe: a novel approach to the automatic design of control software for robot swarms. Swarm Intell. **8**(2), 89–112 (2014)
9. Halme, A.: Kilobot app–a kilobot simulator and swarm pattern designer. https://github.com/ajhalme/kbsim (2012). Accessed 20 Apr 2018
10. Jakobi, N.: Evolutionary robotics and the radical envelope-of-noise hypothesis. Adapt. Behav. **6**(2), 325 (1997)
11. Jakobi, N., Husbands, P., Harvey, I.: Noise and the reality gap: the use of simulation in evolutionary robotics. In: Morán, F., Moreno, A., Merelo, J.J., Chacón, P. (eds.) ECAL 1995. LNCS, vol. 929, pp. 704–720. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59496-5_337
12. Jansson, F., et al.: Kilombo: a Kilobot simulator to enable effective research in swarm robotics. arXiv.org:1511.04285 (2015)
13. Li, W., Gauci, M., Gross, R.: Turing learning: a metric-free approach to inferring behavior and its application to swarms. Swarm Intell. **10**(3), 211–243 (2016)
14. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: a multi-agent simulation environment. Simulation **81**(7), 517–527 (2005)
15. Miglino, O., Lund, H.H., Nolfi, S.: Evolving mobile robots in simulated and real environments. Artif. Life **2**(4), 417–434 (1995)
16. Mondada, F., et al.: SWARM-BOT: a new distributed robotic concept. Auton. Robots **17**(2), 193–221 (2004)
17. Pinciroli, C., et al.: ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intell. **6**(4), 271–295 (2012)
18. Reina, A., Cope, A.J., Nikolaidis, E., Marshall, J.A.R., Sabo, C.: ARK: augmented Reality for Kilobots. IEEE Robot. Autom. Lett. **2**(3), 1755–1761 (2017)
19. Rohmer, E., Singh, S.P.N., Freese, M.: V-REP: a versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1321–1326 (2013)
20. Rubenstein, M., Ahler, C., Hoff, N., Cabrera, A., Nagpal, R.: Kilobot: a low cost robot with scalable operations designed for collective behaviors. Robot. Auton. Syst. **62**(7), 966–975 (2014)
21. Rubenstein, M., Cabrera, A., Werfel, J., Habibi, G., McLurkin, J., Nagpal, R.: Collective transport of complex objects by simple robots: theory and experiments. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), pp. 47–54. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2013)
22. Rubenstein, M., Cornejo, A., Nagpal, R.: Programmable self-assembly in a thousand-robot swarm. Science **345**(6198), 795–799 (2014)
23. Trianni, V., Dorigo, M.: Self-organisation and communication in groups of simulated and physical robots. Biol. Cybern. **95**(3), 213–231 (2006)
24. Valentini, G., et al.: Kilogrid: a novel experimental environment for the Kilobot robot. Swarm Intell. **4**(4), 1–22 (2018)