Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

# Design and analysis of an E-Puck2 robot plug-in for the ARGoS simulator

Daniel H. Stolfi [a,*], Grégoire Danoy [a,b]

[a] Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg
[b] FSTM/DCS, University of Luxembourg, Luxembourg

A R T I C L E   I N F O

*Article history:*
Received 14 December 2022
Received in revised form 17 February 2023
Accepted 18 March 2023
Available online 24 March 2023

*Keywords:*
E-puck2
ARGoS
Computer simulations
Sensors
Swarm robotic

A B S T R A C T

In this article we present a new plug-in for the ARGoS swarm robotic simulator to implement the E-Puck2 robot model, including its graphical representation, sensors and actuators. We have based our development on the former E-Puck robot model (version 1) by upgrading the existing sensors (proximity, light, ground, camera, and battery) and adding new ones (time of flight and simulated encoders) implemented from scratch. We have adapted the values produced by the proximity, light and ground sensors, including the E-Puck2's onboard camera according to its resolution, and proposed four new discharge models for the battery. We have evaluated this new plug-in in terms of accuracy and efficiency through comparisons with real robots and extensive simulations. In all our experiments the proposed plug-in has worked well showing high levels of accuracy. The observed increment of execution times when using the studied sensors varies according to the number of robots and types of sensors included in the simulation, ranging from a negligible impact to 53% longer simulations in the most demanding cases.

## 1. Introduction

Simulations are a vital part of robotic research [1] in order to design, develop and test different algorithms before going for real robots experiments, improving reproducibility, scalability and safety. There is a trade-off between accuracy and simulation time [2] which is especially relevant when thousands of simulations using many entities have to be performed. Other aspects [3] to be taken into account are programming language support, documentation, user interface, debugging techniques, physical fidelity, functional fidelity, ease of development and cost.

ARGoS [4] is a widespread, well-known swarm robotic simulator featuring a modular multithread architecture which is able to efficiently simulate multi-robot swarms, including sensors, actuators, and communications. Optionally, it also allows the visualisation of the 3D simulation environment (arena). Its performance has been proven [3] to be one of the highest with a low use of memory when it was compared to other simulators, e.g. V-Rep [5] and Gazebo [6]. As mentioned, ARGoS' modular architecture allows extending its features, adding new robots, sensors, etc., without modifying the original core code. The current version (3.0.0-beta59) includes the following robot plug-ins: E-Puck, Eye-Bot, Foot-Bot, Pi-puck, Spiri, and some extensions for Kilobot, KhepheraIV, etc.

The E-Puck robot [7] is a two-wheel small ground robot, specifically designed for educational purposes, featuring diverse types of sensors and extensions. The real robot and its simulated model have been used in different research works since 2005, such as formation [8,9], acoustic identification [10], testing fuzzy logic controllers [11], neural control of wheeled mobile robots [12], mobile wireless sensor networks [13], trajectory planning and collision avoidance [14], just to name a few.

On the contrary, the E-Puck2 (E-Puck version 2, Fig. 1) is still a relatively new development (2018) although it has been already seen in proposals related to cooperative localisation using limited communication resources [15], testing a landslide victim detection system [16], decentralised event-triggered cooperative localisation [17], and robot formation surrounding a central object [18], among others. It includes new sensors and some upgrades with respect to the previous model which are not present in ARGoS nowadays.

Hence, in this article we present the implementation, study, and validation through experiments of our ARGoS plug-in for the E-Puck2 robot, based on the original E-Puck plug-in. New sensors such as distance (Time of Flight) and simulated encoders where implemented and others (proximity, ambient light, ground, camera and battery discharge models) were upgraded to achieve more accurate simulations and range of output values. We have

* Corresponding author.
*E-mail addresses:* daniel.stolfi@uni.lu (D.H. Stolfi), gregoire.danoy@uni.lu (G. Danoy).

https://doi.org/10.1016/j.robot.2023.104412

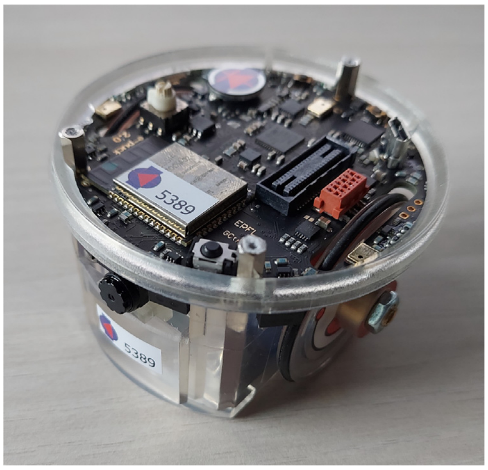


**Fig. 1.** The E-Puck2 robot (E-Puck version 2).

validated our implementation by comparing results from simulations with data obtained from real robots. Additionally, we have addressed the execution times for different number of simulated robots and sensors to better know the extra computing resources needed when they are active. We believe that this plug-in can be useful for researchers who would like to test and tune their proposals involving E-Puck2 robots in a simulator (*in silico*) before validating their results with real E-Puck2 robots (*in vivo*).

The rest of this paper is organised as follows. In the next section, we review the state of the art related to our work. In Section 3 our proposed plug-in and the different sensors and actuators included are presented. The conducted experiments, their results and validation are discussed in Section 4. And finally, in Section 5, conclusions and future work are given.

## 2. Related work

In this section we review some related works in which the authors have developed and evaluated different plug-ins for the ARGoS simulator, including the former E-Puck robot and the current E-Puck2.

The E-Puck robot is presented in [19] as a simple, robust and user-friendly open-hardware robotic platform. In this paper the authors describe the robot's sensors, capabilities and comment on communication experiments such as a foraging arena where food and poison are differently coloured [20]. The E-Puck robots were able to identify these areas using ground sensors and communicate each other using omni-directional camera turrets. The E-Puck robot is still being used in several research works [21,22].

The Pi-puck extension consists of a Raspberry Pi interface and software, initially supporting E-Puck robots [23], and lately extended to E-Puck2 robots [24]. This board is intended to enhance the processing power, memory and communication of the robots leaving the robot to deal with motor control and sensor interfacing. The Raspberry Pi Zero board is connected to the robot's microcontroller via I2C bus which makes it compatible with both the E-Puck and E-Puck2. It features long range sensors, audio amplifier plus speaker, an OLED display, extra RGB LEDs and USB ports, allowing developing new applications such as image processing (up to $640 \times 480$, 15 fps) without relying on external resources, taking advantage of an embedded Linux system. Additionally, it provides an extra battery and two spring contacts allowing the use of a charging wall. Recent research works using the Pi-puck robot include the use of blockchain technology to secure robot swarms [25] and the analysis of the Buzz Swarm programming language [26].

An infrastructure composed by E-Puck robots and task abstraction modules (TAM), hosted by the DEMIURGE project, is presented in [27]. The robots used in the project feature extension boards such as ground sensors, an omni-directional camera, and an embedded computer running Linux. The robots' firmware was adapted to the new hardware configuration and the new model has been integrated into the ARGoS simulator. Additionally, the TAM included in the project's infrastructure allows researchers to omit the details of the task execution, focusing on the logical relations between multi-robot tasks. The TAM shape resembles a booth where an E-Puck robot can enter. It is equipped with two light barriers, three RGB LEDs, and an infrared link for communications.

An open source simulator based on ARGoS framework for the Thymio [28] robotic platform is available at [29]. It uses an interface, which is developed using core libraries from ASEBA [30]. Thymio is an educational two-wheel robot equipped with a loudspeaker, Li-Po battery, microphone, three-axis accelerometer, five proximity sensors, two ground sensors, two proximity sensors, 39 LEDs, USB connection, buttons, activity display, and a memory-card slot. It runs the ASEBA open-source programming environment designed to be easy to use, featuring a lightweight virtual machine that runs on the Thymio microcontroller.

A plug-in for the ARGoS simulator to experiment with Kilobots is presented in [31]. The Kilobot is a small robot which moves by using two vibrational motors. It has infrared transceivers for communications and detection of other robots, light sensors, and LEDs. It has been modelled in ARGoS using a simple 3D representation and its simulated motors were calibrated according to the nominal robot characteristics. After describing the robot plug-in, the authors present a set of experiments to validate the simulations in terms of accuracy and scalability, by comparing their models with real robots. Recently, Kilobots have been used to study different communication ranges in a swarm of 50 robots [32] and to implement collective decision-making using a distributed Bayesian approach [33].

A Swarm Robotics Construction System (SRoCS) composed of Stigmergic Blocks as building material and a BuilderBot robot is discussed in [34]. The authors proposed a set of block algorithms where the intelligence that coordinates the building process is in the blocks, and compare it with standard algorithms where the intelligence is in the robots. In order to perform simulations in ARGoS, the authors have developed *ad-hoc* plug-ins to provide comprehensive models of robots and blocks.

Regarding other simulators, an E-Puck2 plug-in for the Webots [35] simulator is available. It includes support for the differential wheel motors, simulated encoders, proximity and light sensors, accelerometers, gyroscopes, ground sensors, camera and LEDs. Webots is an open source, multi-platform robot simulator which presents much more elaborated graphic representations than ARGoS as it is very focused on 3D visualisations. In addition to the models included in Webots, some research works propose new sensors for this simulator such as the case of the Servosila Engineer crawler robot [36] and new ultrasonic sensors [37] for the Spherical Underwater Robot (SUR) [38].

We propose a new plug-in for the E-Puck2 robot which includes the new LEDs, adapted proximity, light and ground sensors, and novel simulated encoders, battery discharge models, time of flight sensor, and onboard camera. It will allow the researchers using these robots to conduct their experiments in the ARGoS simulator, having a more accurate model (sensors and range of values) than the currently offered by the available E-Puck (version 1) model. In the following sections we describe the plug-in and its sensors, analyse their behaviour and performance, and validate them with actual robots.

## 3. E-Puck2 plug-in for ARGoS

The E-Puck2 robot was developed in 2018 by GCtronic [39], a spin-off of the Autonomous System Lab at the Swiss Federal Institute of Technology Lausanne. It is an open hardware development which features a STM32F407 microcontroller @ 168 MHz, DSP and FPU, 192 kB of RAM, 1020 kB of ROM, eight proximity and ambient light sensors, one distance sensor (Time of Flight), a 3D inertial measurement unit (accelerometers and gyroscopes), 4 omnidirectional microphones, a VGA colour camera (typical use: 160 × 120), IR receiver for remote control, 10 LEDs (4 RGB) and a speaker. Two stepper motors control the corresponding two wheels to allow moving the robot up to 15.4 cm/s, providing also a pair of simulated encoders. The robot is powered by a LiPo (lithium-ion polymer) rechargeable battery capable of providing 1800 mAh. The E-Puck2 is a 7-cm-diameter cylindrical robot, 4.5-cm height, weighting 130 g. It can be controlled by embedded software or via Bluetooth, USB or Wi-Fi links.

Besides the onboard sensors, there are several expansion boards available for the E-Puck2. Range and bearing [40] is an expansion board which is installed on the upper part of the robot, using the I2C bus, to provide short-range infrared communication between robots. Another expansion consists of three ground sensors [7] meant to detect different levels of grey in the surface below the robot.

Our implemented E-Puck2 plug-in is based on the original E-Puck plug-in developed by Carlos Pinciroli in the ARGoS simulator [4]. We have implemented new actuators (LEDs) as well as new sensors (wheel encoders, time of flight) and battery discharge models corresponding to the new robot's features. We have also updated the already existing sensors (proximity, ambient light, camera, and battery) to match the data observed in the E-Puck2 through our experiments, increasing the accuracy of the simulations and allowing to work with scenarios that were impossible before, e.g. measuring accurate distances. It can be used in ARGoS by programming and compiling the controller's C++ code, although Lua is also supported. This is especially useful for learning robotic programming without having to setup C++ build environments (Lua is a lightweight, embeddable scripting language).

Regarding the expansion boards, the range and bearing sensor/actuator already exists in the ARGoS simulator, and can be easily added to the simulated robots through the configuration file, including our E-Puck2 plug-in. However, we have developed a specific ground sensor for our plug-in, based on the existing generic ground sensor implemented in ARGoS, by adapting the measured values to the range observed in the real robots.

Fig. 2 shows a schema with the sensors and actuators modelled in the E-Puck2 plug-in. A comparison between the new plug-in and the former one is available in Table 1. As it can be seen the range of values has been modified according to the values provided by the E-Puck2 robot when communicating using the SERCOM protocol. The implemented sensors allow using the same noise model provided by ARGoS, i.e. uniformly distributed between −1 and 1. The following sections describe the modifications and additions performed, individually.

### 3.1. Body and LEDs

The implemented E-Puck2 plug-in presents a redesigned 3D model to resemble the real robot, although the level of detail was kept low in order to ensure low computing resources usage when rendering the robot model (only important when the 3D visualisation is active). The number and types of LEDs were updated with respect to the original E-Puck, to represent now the four red LEDs and the other four RGB LEDs found on the top of the new robot. Additionally, the green body LED was added as well as the red front LED. All these LEDs can be detected by other robots' (not only E-Puck2's) using their corresponding sensors. Fig. 3 shows the new robot's 3D model and its LEDs, while Table 2 lists the LEDs characteristics.

**Table 1**
Characteristics of the implemented E-Puck2 plug-in compared to the former E-Puck plug-in.

| Actuator/Sensor | E-Puck | E-Puck2 |
|---|---|---|
| LEDs | 8 RGB | 4 RED + 4 RGB + FRONT + BODY |
| Proximity | 8 [0.0–1.0] | 8 [0 – 4095] |
| Light | 8 [0.0–1.0] | 8 [4095 – 0] |
| Ground | 3 [0.0–1.0] | 3 [0 – 1023] |
| Time of flight | – | 1 [20 – 2000] |
| Simulated encoders | – | 2 [−32 768 – 32 767] |
| Camera | – | 1 (160 × 120) |
| Battery | Generic | 4 discharge models |

**Table 2**
LEDs included in the proposed E-Puck2 plug-in.

| LED | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Colour | RED | RGB | RED | RGB | RED | RGB | RED | RGB | RED | GREEN |
| Position | 0° | 45° | 90° | 135° | 180° | 225° | 270° | 315° | Front | Body |

**Table 3**
Orientation of the E-Puck2's proximity and light sensors.

| Sensor | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Angle | 15° | 50° | 90° | 150° | 210° | 270° | 310° | 345° |

### 3.2. Proximity and light sensors

The proximity and ambient light sensors in the proposed E-Puck2 plug-in were placed according to the real positions in the new robot (Table 3). They are described together in this section because both functionalities are provided by the same reflective sensor (TCRT1000) including an infrared emitter and phototransistor which detects infrared light.

The proximity sensors have a maximum detection distance of 50 mm while the light sensors can detect light sources up to 50 cm away according to the robot specifications. The returned range of values is between 0 and 4095. For the proximity sensors, 0 corresponds to no object detected and 4095 to an object touching the sensor. On the contrary, a saturated light sensor would produce a 0-output value and 4095 when there are not light sources detected.

### 3.3. Ground sensor

The ground sensor actually comprises three different sensors located at the bottom front of the E-Puck2. They are arranged in a row perpendicular to the robot moving direction and measure the grey level of the ground under the robot. Although this sensor is an optional part of the commercial E-Puck2 robot, we have decided to model it due to its utility for many experiments based on sensing data from the ground, e.g. robots following a fixed path painted in the ground. These sensors have the same implementation as in the original E-Puck but in this case the range of the measured values has been adapted to the actual sensor specifications. Consequently, its range goes from 0 (black) to 1023 (white).

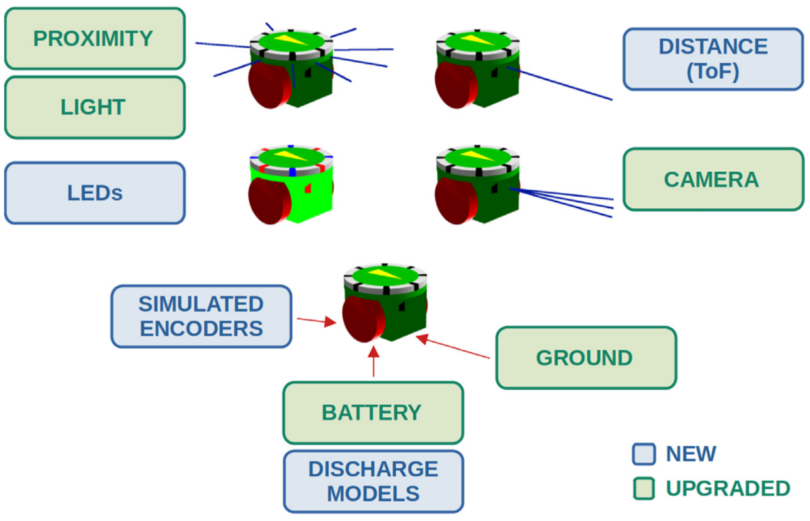


**Fig. 2.** Sensors and actuators modelled in the E-Puck2 plug-in.

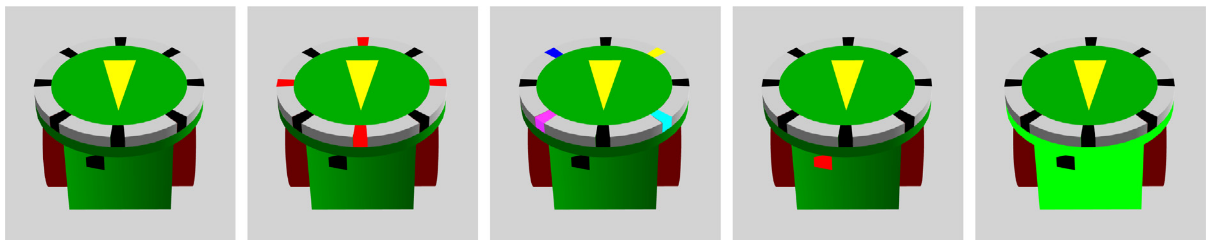


**Fig. 3.** E-Puck2's 3D model and the new LEDs included, i.e. four red, four RGB, the front LED and the green body. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

### 3.4. Time of Flight (ToF) sensor

We have implemented this new sensor in the ARGoS simulator for the E-Puck2 robot plug-in using the ray intersection functions provided by the simulator libraries. It mimics the laser sensor (STM-VL53L0X) installed at the front of the actual E-Puck2 robots which is capable of measuring distances to obstacles up to 2 m away. The values provided by the ToF sensor are in millimetres, having a minimum distance value of 20 mm. as observed in the real robots.

### 3.5. Simulated encoders

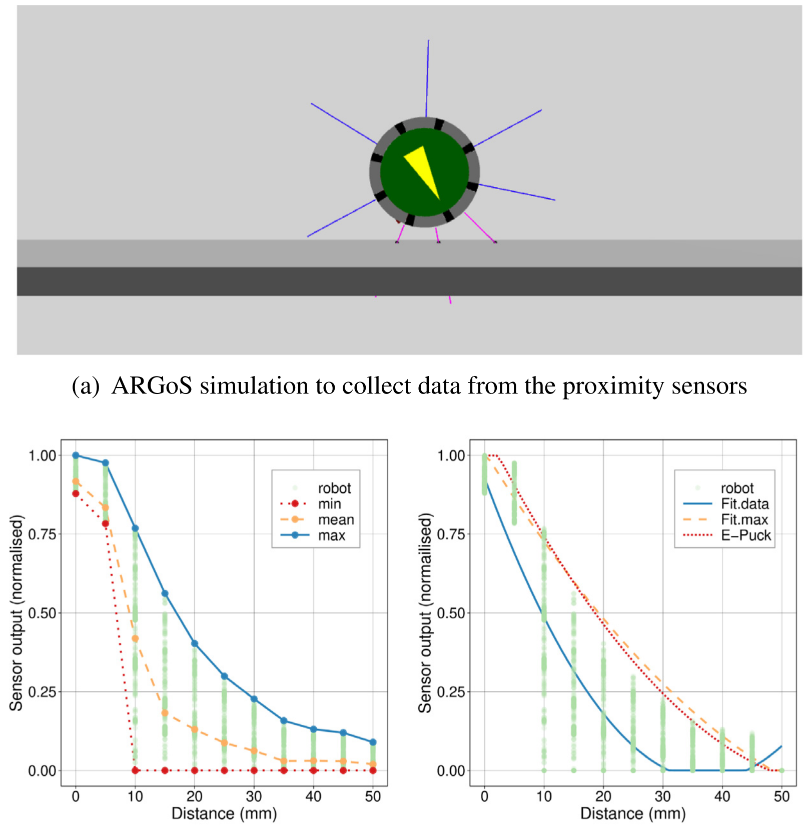
This is another sensor we have implemented from scratch for the E-Puck2 robot plug-in as it was not present in the former robot plug-in. It consists of two registers to count the number of steps that each wheel rotates with a precision equal to 0.36 degrees (1000 steps per wheel revolution). The range of values is (−32 768–32 767) corresponding to the robot's 16-bit registers. Internally, we have used real values to represent partial wheel rotations that do not end in a new step but add to the next future value, e.g. two rotations of 0.18 degrees would produce 1 step after the second rotation but no extra step after the first one.

### 3.6. Camera

The E-Puck2's onboard camera is the Omnivision OV7670 CMOS image sensor, featuring a resolution of 640 × 480 pixels. Due to the limited robot's hardware capabilities, it provides by default an image 160 pixels wide by 120 pixels high (full size frames can be obtained using an extension board). It was replicated in our plug-in using the existing model of a coloured perspective camera included in ARGoS. In contrast, the original E-Puck robot provides a 52 × 39-pixel image or alternatively 480 × 1, as it has a lower onboard memory available. This camera has not been implemented in the currently available E-Puck plug-in. The simulated camera is able to detect LEDs and project their 3D coordinates in a rectangular coordinated space, reporting also their colours. Since the aperture and maximum range of the camera are not available in the robot specifications, they are experimentally calculated in Section 4.1.6

### 3.7. Battery

The original E-Puck plug-in does not include its own battery model, relying on a generic implementation provided by the simulator. Consequently, we wanted to define and test different battery models for our E-Puck2 plug-in. With that objective in mind, we have collected data from several robots to assess the actual battery discharge curves and proposed different discharge models, from a simpler and lightweight linear model to a more accurate but also more complex model, based on polynomials. These experiments are presented in Section 4.1.7.



(a) ARGoS simulation to collect data from the proximity sensors

(b) Collected data

(c) Curve fitting

**Fig. 4.** Proximity Sensors: ARGoS simulation scenario, data collected from the robots, and polynomial fitting compared to the E-Puck's (version 1) model.

## 4. Experiments and results

In this section we present the experiments conducted to model and evaluate the implemented sensors as well as the observed results. We have used E-Puck2 robots (Firmware 11.01.21 cf7e095,[1] Radio 25.02.19 e2f4883[2]) and the ARGoS simulator (Version 3.0.0-beta59) running in a DELL XPS 15 9570 (12 Intel Core i7-8750H CPU @ 2.20 GHz and 16 GB of RAM). We have set up 10 simulation ticks per second as resolution (the default value) and each experiment was conducted in its own arena, i.e. a wall close to robots for testing the proximity sensors, some light sources for testing the light sensors, etc. The main objectives of the experiments were to evaluate the implemented sensors and compare them with the actual robot, and also assess the performance impact of using those sensors in the simulation model.

### *4.1. Sensor and battery discharge models*

The model of each sensor has been calculated based on data we collected from real robots and validated to produce realistic simulations with the proposed E-Puck2 plug-in. The following sections describe the process followed for each sensor.

[1] https://github.com/e-puck2/e-puck2_main-processor/commit/cf7e095.

[2] https://github.com/e-puck2/esp-idf/commit/e2f4883.

#### *4.1.1. Proximity sensors*

To calibrate the proximity sensors an experiment was conducted by measuring the readings from two different static robots using their front right sensor (PROX0) during 15 s (about 24 readings per second) to calculate the maximum, minimum and mean values as shown in Fig. 4(b). The different measures were collected by placing the real robot with its sensor under test facing a white cardboard surface, at the predefined distances. These measures can be affected by several factors such as the angle of incidence, surface colour and texture, ambient light, etc. We have tried our best to keep the same conditions for all these experiments to achieve reliable and reproducible results.

Once completed the data collection, a quadratic polynomial (Eq. (1)) was fitted to both, the mean points and the maximum values, as shown in Fig. 4(c) where the polynomial proposed by Garattoni and Francesca [27] for the E-Puck robot was also drawn. It can be seen that if we choose the model fitted to the maximum values, the resulting polynomial is almost coincident (when the values are normalised in the range 0–1) with the E-Puck's model (version 1), what makes sense as they use the same sensor component. The values of the calculated coefficients for the polynomial fitted to the maximum points are $a = 197.8633$, $b = -30.47182$, and $c = 1.011635$, where the values used in the original E-Puck were slightly different, i.e. $a = 298.701$, $b = -36.8961$, and $c = 1.08212$. We have used the new calculated polynomial and also adapted the range of values produced by the plug-in to match the real robot's, i.e. 0 – 4095, where 0 means that no object was detected by the sensor. The ARGoS simulation scenario used for collecting data from the proximity sensors is shown in Fig. 4(a).

$$Prox(d) = \begin{cases} \min(1, a \times d^2 + b \times d + c), & \text{if } d \leq 0.05 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

### 4.1.2. *Light sensors*

We have tested the ambient light sensors by collecting data from different distances using two sensors (PROX0 and PROX7) of a real robot facing an infrared LED, to assess their characteristic response curve. The distance between the light source and the sensor was set to 10, 20, 30, 40, and 50 cm, corresponding to light intensities of 1.24, 0.62, 0.40, 0.25, and 0.20 lx, respectively. The collected data points, shown in Fig. 5(b), were used to calculate a new transfer function (Eq. (2)) based on the one proposed for the original E-Puck robot. There, $i$ is the intensity of the light source, $d$ is its distance to the sensor, and $\gamma = 0.075$ is a constant to obtain the new response curve for the E-Puck2 plug-in, also shown in Fig. 5(b). The intensity values were normalised to the range [0-1] to easily compare the new transfer function (E-Puck2) with the former one (E-Puck). Note that the ARGoS implementation assigns a value 1.0 to a saturated light sensor while the actual values obtained from the robots are the opposite.

$$Light(i, d) = \begin{cases} \max(0, 1 - \frac{(\gamma \times i)^2}{d^2}), & \text{if } d \leq 0.50 \\ 1, & \text{otherwise} \end{cases} \quad (2)$$

Our implementation of the light sensor was tested in simulations using different light sources (intensities) producing the results shown in Fig. 5(c). It can be seen that the function fitted to the collected data corresponds to a light source of intensity 1 (light sources do not have any unit in ARGoS). Different light intensities are producing the saturation of the sensor (output equal to 0) at different distances, as expected. There is a limitation in the simulations using these sensors. As in the case of the original E-Puck, the detections in ARGoS are meant to detect LEDs (visible light) in the scenario. On the other hand, the infrared sensors of the real robots will not detect the LEDs from other robots but the infrared emissions from the other proximity sensors. The ARGoS simulation scenario used for collecting data from the light sensors is shown in Fig. 5(a).

### 4.1.3. *Ground sensor*

We have set up an experiment to test the implemented ground sensor (made of three individual sensors), consisting in two scenarios where it is exposed to different levels of grey. In the first scenario (Fig. 6(a)) an E-Puck2 robot crosses the simulation arena from left to right whilst in the second, the robot rotates at the centre of the arena (Fig. 6(b)). The detections obtained from the first scenario show that the readings from the three individual sensors are almost coincident as the robot moves in a straight line (Fig. 6(c)). However, the second scenario presents detections shifted in time as each new grey level is detected in order, sequentially, following the sensor locations at the bottom of the robot (Fig. 6(d)).

When testing the same patterns on the E-Puck2 robot we have observed some residual detections for black (likely due to some infrared light still being reflected by the black surface) and also some discrepancies among them, i.e. slightly different readings for the same level of grey. Figs. 6(e) and 6(f) show a comparison between the actual sensor readings and the values provided by our ground sensor implemented for the E-Puck2 plug-in. Despite the clear differences observed from the ideal sensors and the real devices, the experiment outcome shows that detecting different grey levels is possible, whether it is done using the plug-in or the actual robot.

**Table 4**
Angles calculated from the camera experiment and from the simulated scenarios.

| E-Puck2 robot | | | ARGoS Plug-in | | |
|---|---|---|---|---|---|
| x | y | $\alpha$ | x | y | $\alpha$ |
| 250 | 144 | 32.1° | 263 | 141 | 30.0° |
| 150 | 86 | 29.5° | 143 | 77 | 30.1° |
| Mean: | | 30.8° | Mean: | | 30.1° |

### 4.1.4. *Time of Flight (ToF) sensor*

For the ToF distance sensor we have run a series of experiments comparing the measures from simulations (Fig. 7(a)) with the values collected from the robots. We have set up a robot moving in straight line, at six different speeds, to collide with an obstacle placed 30 cm away. Our results can be seen in Fig. 7, where the values collected from the robots show a classical noise (which can be also simulated by the plug-in, as in the original E-Puck, using the ARGoS' noise generator, although it was not used in our study), plus a few bounces when the obstacle used in the experiment was reached. Since we have used the robot speed to estimate the distance travelled by time unit, the plotted lines are not exactly coincident. It can be attributed to wheel spins and speed inaccuracies happening in the real robot which are bigger at higher accelerations.

### 4.1.5. *Simulated encoders*

The simulated encoder sensor was tested in the same scenarios as in the ToF sensor (Fig. 8(a)), comparing the simulation results with a real E-Puck2 robot. Fig. 8 shows the results obtained where we can see that wheel spins and probably speed inaccuracies were also present in this experiment. There were some irregularities in the measures obtained from the real robot, due to the fact that the sampling rate, when accessing to the sensor readings via WiFi connection, presents some variability (hundredths of a second), more noticeable at higher speeds. Although, the observed offset between the measures do not invalidate the experiment, it is possible that internally recording the encoder values directly on the robot would have provided more accurate readings.

### 4.1.6. *Camera*

Two experiments were performed to calculate the vision angle of the robot's camera as shown in Fig. 9(a). The measured values are in Table 4 as well as the resulting angle values ($\alpha$). We have chosen $\alpha = 30°$ to model the camera in our plug-in, and tested it as shown in Fig. 9(b). It can be seen that the red LEDs of the robots placed at the border of the camera vision are mapped at the minimum and maximum horizontal coordinates of the points detected by ARGoS (Fig. 9(c)). Moreover, the central blue light is detected at the centre of the image as it was expected, according to its position in the simulation scenario. The maximum range for the camera implementation was experimentally set to 1 m as the limited resolution makes it difficult to identify objects farther away when using the real robot.

### 4.1.7. *Battery*

The battery model in ARGoS relies on an associated discharge model. We wanted to provide a new set of discharge models for the E-Puck2 plug-in so that researchers can decide between a faster or a more accurate model, depending on their needs. We have evaluated robots moving at different speeds as we have assumed that the main components draining the batteries were the stepper motors.

As a first step, we have collected data from eight E-Puck2 robots rotating at different speeds, from static (0 cm/s) to maximum speed (15.0 cm/s), with increments of 1 cm/s, to assess

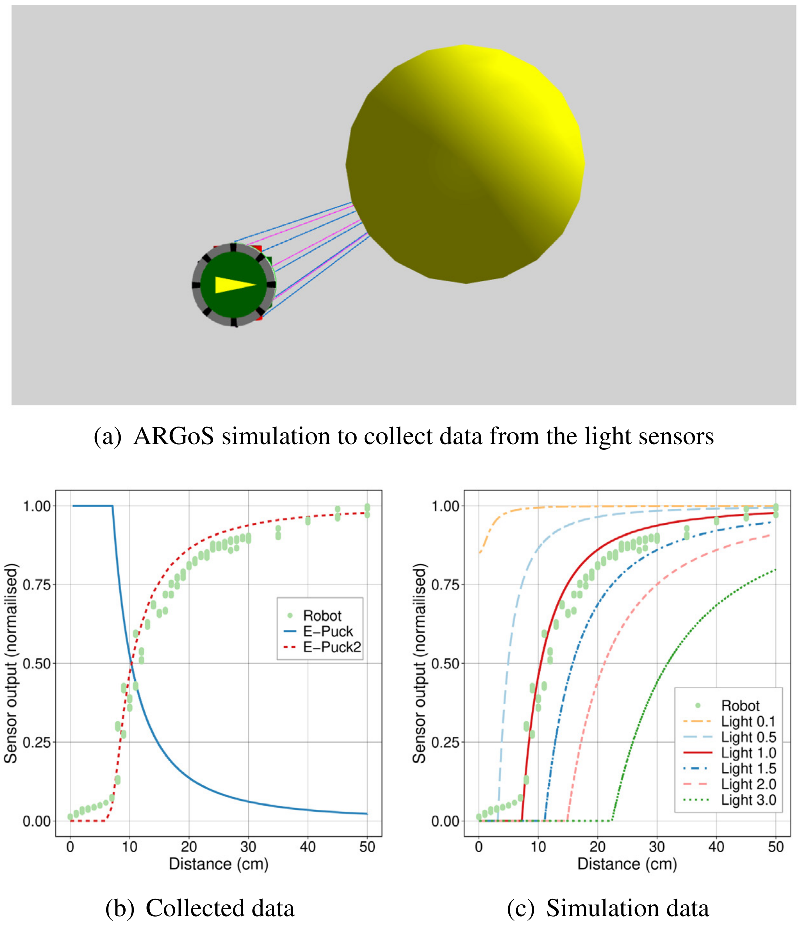


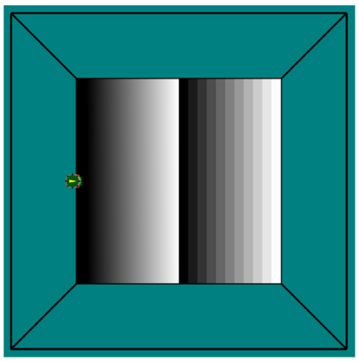
**Fig. 5.** Ambience Light Sensors: ARGoS simulation scenario, data collected and proposed sensor response curves, and data from simulations using different light intensities.

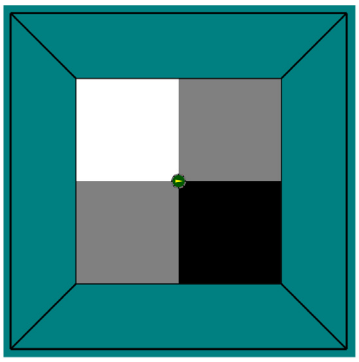
the discharge curve of their batteries in different conditions. The robot movement was implemented by rotating one wheel in one direction and the other in the opposite. We repeated each experiment twice using different robots (and batteries) to obtain the data shown in Fig. 10(a), with the aim of being used to calculate four discharge models: simple, linear, cubic, and approximated.

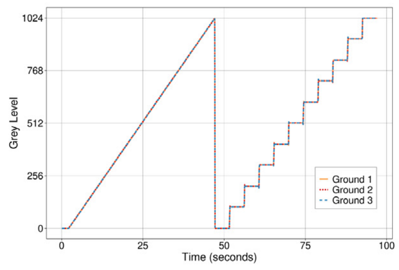
Our first finding was that the autonomy of the battery is not linear, nor monotonously decreasing with the speed, as it was to be expected. On the contrary, as it can be seen in Fig. 10(b), the maximum autonomy decreases when the speed is increasing until 5 cm/s. Beyond that point, an extra battery duration has been observed which is increasing until 14 cm/s are reached. From that point it begins to decrease again. We believe that this behaviour is due to the characteristics of the stepper motors whose coils consume energy permanently, not only when they are rotating. As a consequence, they consume different amounts of energy not only according to the rotational speed but also to the time they are stopped. Beyond 14 cm/s the robots stopped moving despite the fact that some little battery charge was still available. We have considered that point as the end of the experiment as the robots were unusable in such conditions. Having acquired some knowledge about the robot's battery, we propose four discharge models: Simple, Linear, Cubic, and Approximated, shown in Fig. 11, and described in the following sections.

*Simple discharge model.* This model is the simplest one, featuring a low complexity and thus low usage of computational resources. It consists of two linear discharge functions, one fitted to the maximum autonomy, observed when the robot was static (speed equal to 0 cm/s), and the other was calculated using the average autonomy when the robot is moving. The calculated discharge slope for the function corresponding to the static robot is $m_0 = -5.064963 \times 10^{-5}$ while for the moving robots is $m_1 = -8.699124 \times 10^{-5}$. The implemented pseudocode can be seen in Appendix, and the discharge functions compared with the collected battery points are shown in Fig. 12(a).

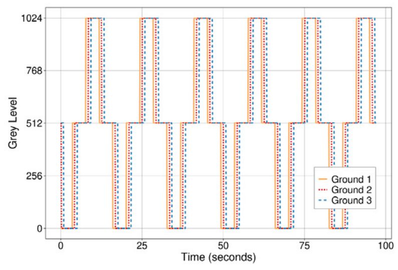
*Linear discharge model.* A more precise model is proposed using not two but seventeen discharge functions, each one fitted to the collected data from the robot for the corresponding speeds. As the robot would (probably) be moving at a different speed from the predefined functions (tested speeds), an interpolation might be required to calculate the next battery charge value. The pseudocode of this model is in Appendix and its block diagram is shown in Fig. 11. The first step is calculating the robot speed and if it corresponds to one of the existing discharge functions, the next battery charge level is obtained. Otherwise, an extra step is needed consisting in interpolating the values of two existing discharge functions, corresponding to the immediately lower and upper speeds, e.g. if the robot is moving at 13.3 cm/s, values from 13 cm/s and 14 cm/s will be used in the interpolation. Now the next battery charge value can be calculated and returned by the algorithm. Note that, this discharge algorithm (and the following ones) is focused on efficiency as it has to be calculated


(a) Moving from left to right

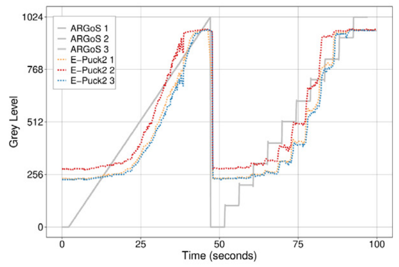

(b) Rotation at the centre



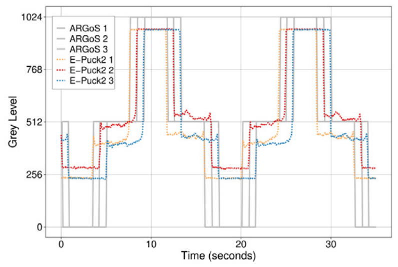
(c) Values from the three ground sensors (moving in a straight line)



(d) Values from the three ground sensors (rotating robot)



(e) Values from the robot's sensor compared to the implemented ARGoS sensors



(f) Values collected from the rotating robot compared to the implemented ARGoS sensors

**Fig. 6.** Ground sensor tests. The testing scenarios are in the upper row, the data collected from the implemented ground sensors, in the central row, and a comparison with the data obtained from the robot is presented in the lower row.

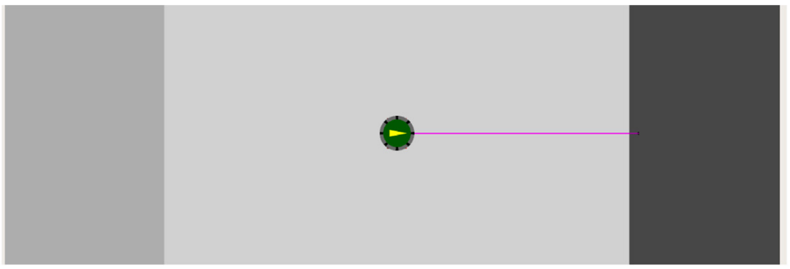
at each simulation step. The different discharge curves for the tested speeds of the Linear model are shown in Fig. 12(b).

*Cubic discharge model.* The Cubic discharge model uses third-order polynomials fitted to the battery data points obtained from the robots. Again, we use 17 curves as this is the number of speeds tested in our experiments. Similarly to the linear algorithm, one discharge function is obtained if the robot's moving speed corresponds to one of the pre-calculated (fitted) polynomials. Otherwise, an interpolation is needed to obtain the next battery charge level. This algorithm is expected to be more demanding in terms of computing resources as it requires calculating cubic roots to obtain the corresponding polynomials. The pseudocode of the Cubic discharge model is also in Appendix and its block diagram, in Fig. 11. The different discharge curves for the Cubic model are shown in Fig. 12(c).

*Approximated discharge model.* Finally, the Approximated discharge model presents a simplification of the Cubic model in which no cubic roots are calculated. Mainly, what it does is dividing each fitted cubic curve into four linear sections and after finding the corresponding line segment, the next battery charge value is calculated using two known speeds and an interpolation, if needed. The discharge curves made of line segments can be seen in Fig. 12(d), the algorithm's block diagram is shown in Fig. 11, and its pseudocode is included in Appendix. This model was expected to have an accuracy comparable to Cubic but at lower computational times. Both metrics are discussed in the following sections for the four proposed battery discharge models.

### 4.2. Accuracy of the discharge models

We have compared the battery charge values obtained from each model with the collected values from the robots in order to assess its accuracy. We have used the Mean Square Error (MSE) as



(a) ARGoS simulation scenario to collect data from the ToF sensor

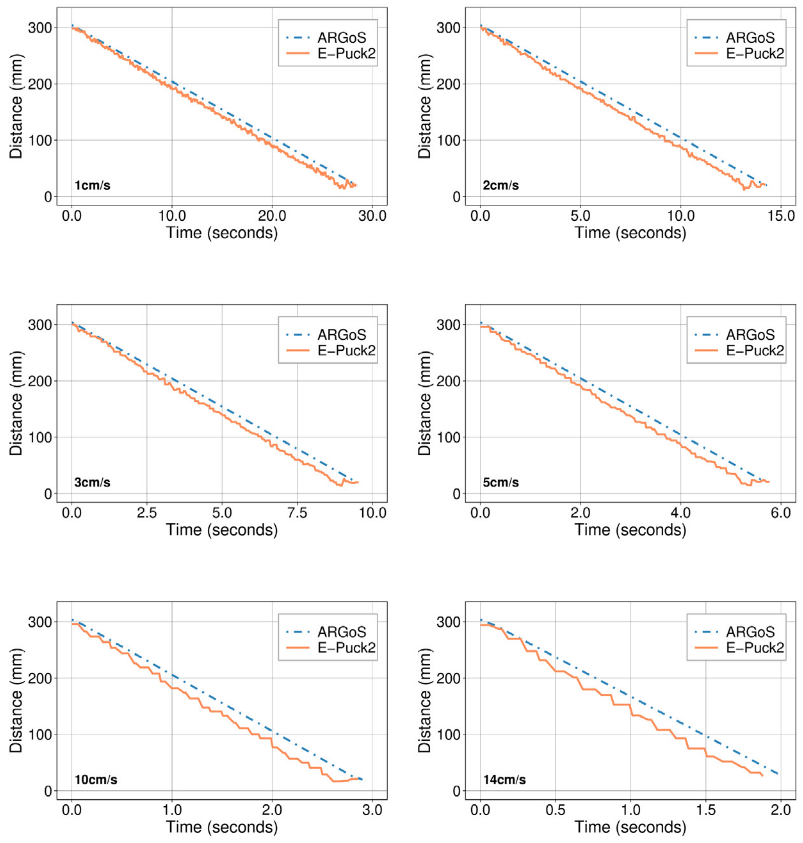


**Fig. 7.** ARGoS simulation scenario for the experiment and distances measured from simulations in ARGoS using the ToF sensor, compared to a real E-Puck2 robot, for six different speeds.

**Table 5**
Average MSE values of the analytic model and those obtained from the ARGoS simulations.

| Source | Simple | Linear | Cubic | Approximated |
|---|---|---|---|---|
| Analytic | 0.0221 | 0.0170 | 0.0020 | 0.0025 |
| ARGoS | 0.0216 | 0.0167 | 0.0019 | 0.0023 |

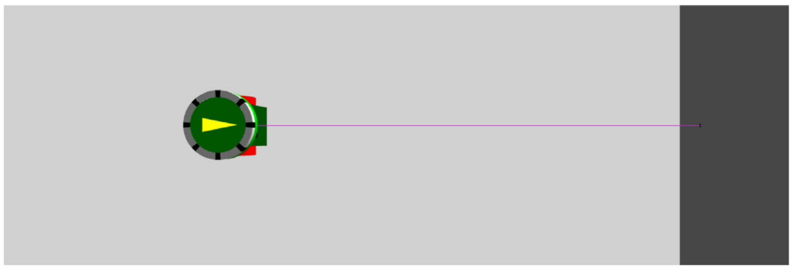
a metric, calculated as shown in Eq. (3), where $n$ is the number of data points, $Y_i$ are the observed values, and $\widehat{Y_i}$ are the estimated values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y_i})^2 \quad (3)$$

The average MSE values are presented in Table 5. They have been obtained from the functions calculated for each discharge model (analysis) and also from the simulations performed in ARGoS. Firstly, we can see that for each model the analytic and ARGoS values are quite similar (the maximum difference is $5 \times 10^{-4}$), denoting an accurate implementation of the models in our plug-in. Secondly, the accuracy values are as expected, being the Simple model the least accurate ($MSE = 0.0216$) and the Cubic, the most ($MSE = 0.0019$). We can also see, that the accuracy of the Approximated model is quite close to the Cubic ($MSE = 0.0032$ vs. $MSE = 0.0019$), representing a viable alternative, providing that the former is more lightweight than the later. In the next section we will try to answer this question by testing each model in terms of execution times.

## 4.3. Execution times

Experimenting with robot simulations usually implies testing many scenarios and problem instances where a few tenths of



(a) ARGoS simulation scenario to collect data from the simulated encoders

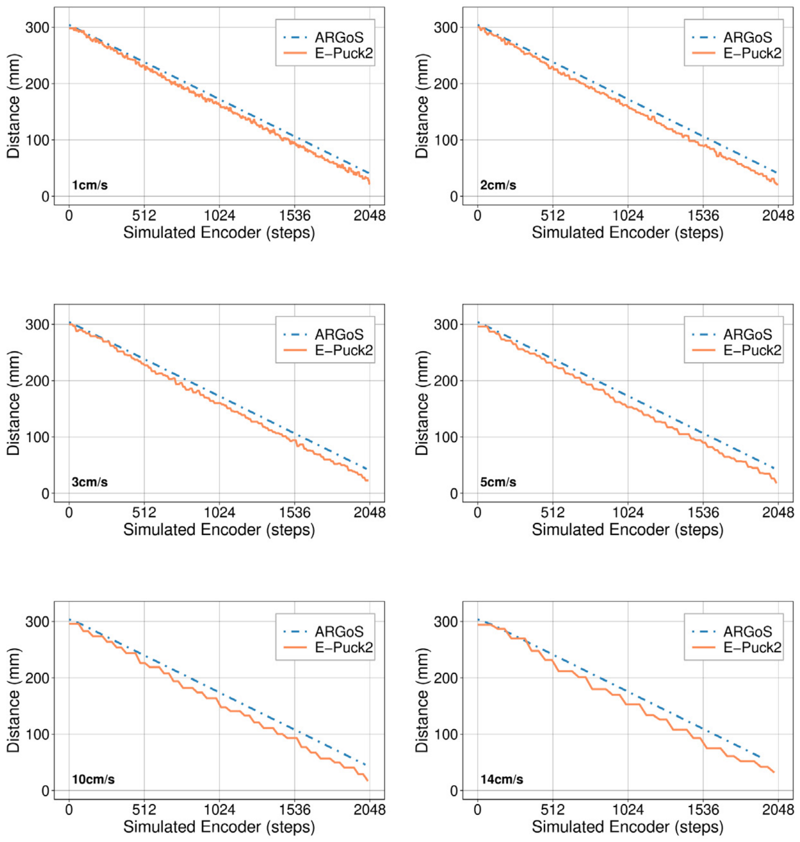
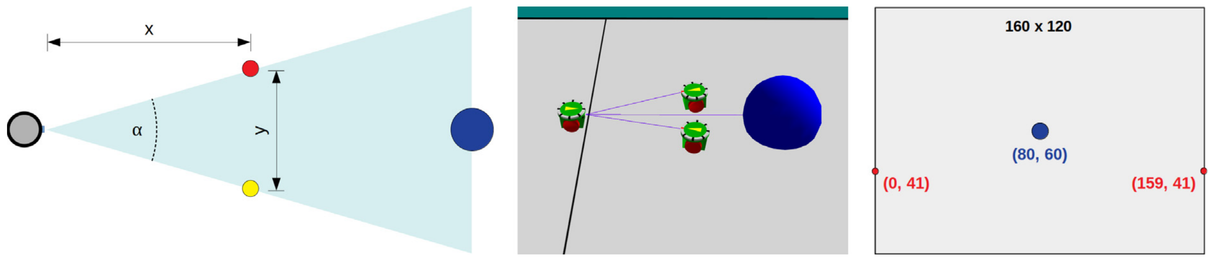
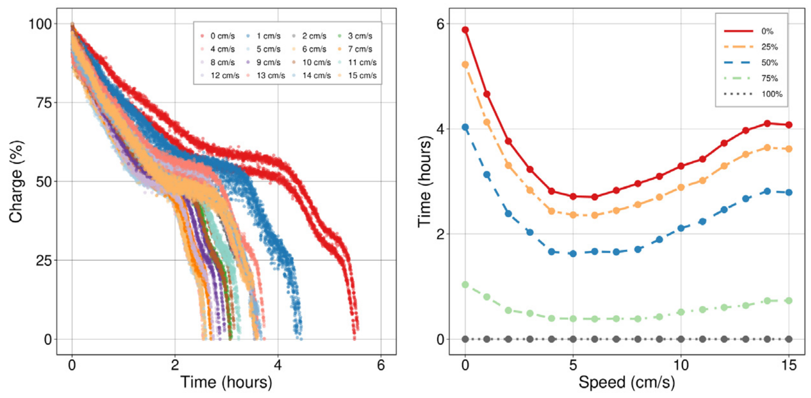


**Fig. 8.** ARGoS simulation scenario for the experiment and distances vs. Simulated Encoder values from simulations in ARGoS compared to a real E-Puck2 robot, for six different speeds.



(a) Experiment setup with the real robot (b) ARGoS simulation scenario (c) LED detections

**Fig. 9.** Experiments conducted with the camera of the real robot and the simulated camera sensor of the proposed E-Puck2 plug-in. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) Collected data for different speeds (b) Battery discharge time vs. speed

**Fig. 10.** Data collected from the battery experiment and curves showing how the depletion time changes depending on the robot speed. Intermediate points for a 25%, 50% and 75% charge are also represented.

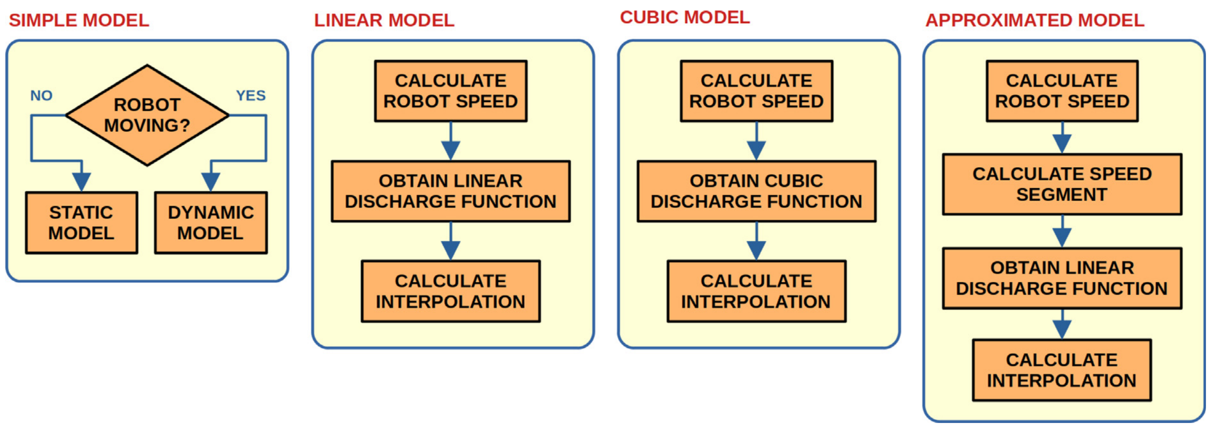
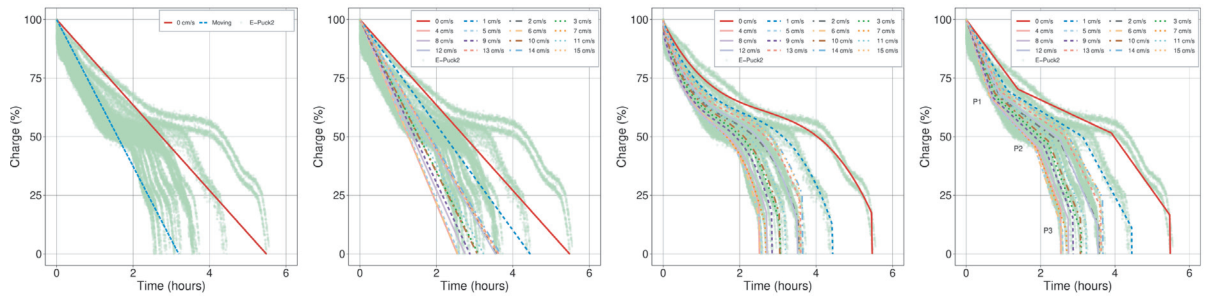


**Fig. 11.** Block diagram of the algorithms proposed for the four battery discharge models.



(a) Simple (b) Linear (c) Cubic (d) Approximated

**Fig. 12.** Plots showing the different battery discharge models and the collected data from the robots in the background.

seconds in a single simulation run can represent several extra hours in the computing cluster. In the following sections we propose the study on how the execution times are affected by using the new sensors in the ARGoS simulations.

### 4.3.1. *Setup*

We have conducted a series of experiments to assess how the simulations are affected in terms of execution times when the studied sensors are active. Different characteristics can increase

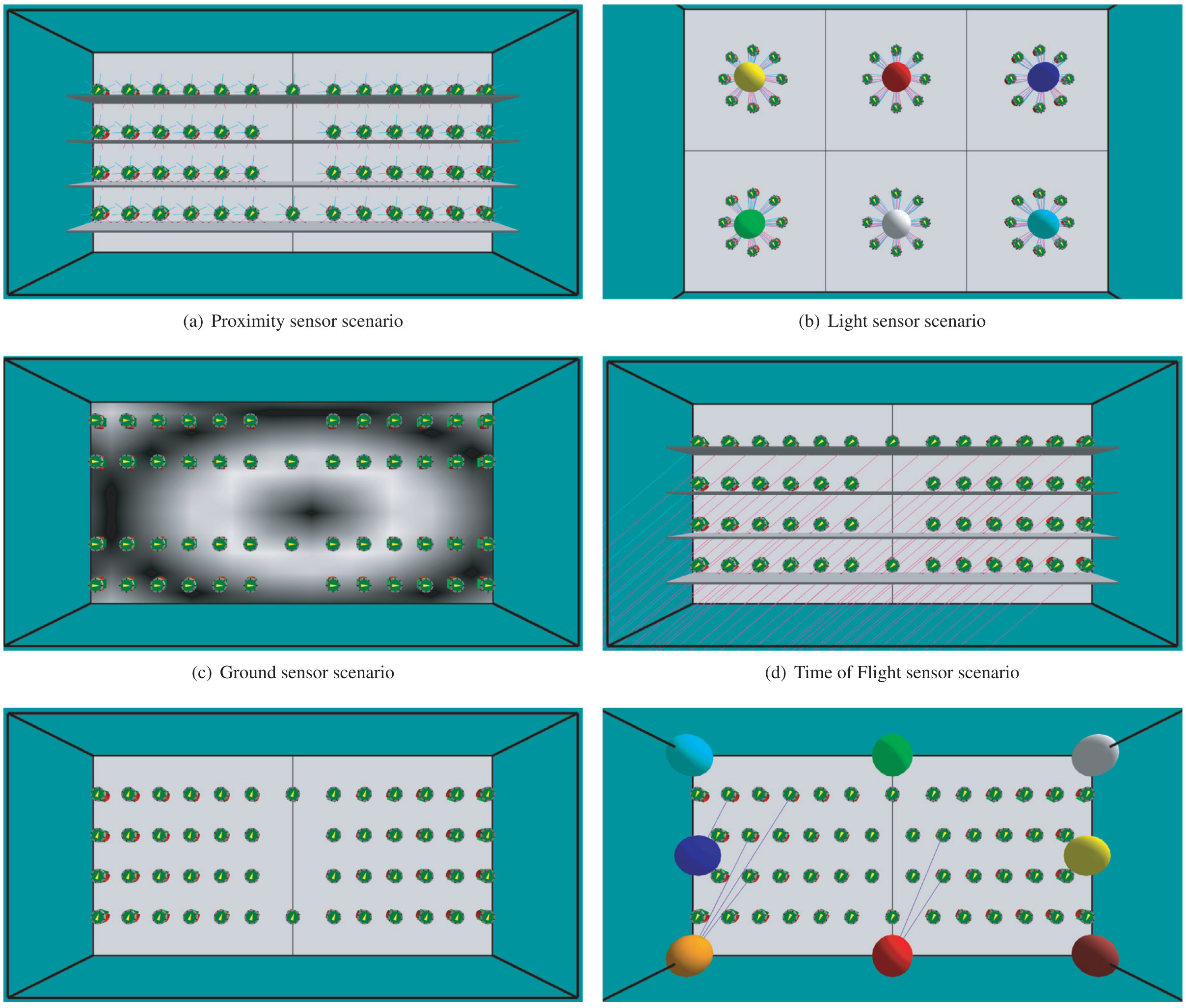

(a) Proximity sensor scenario
(b) Light sensor scenario
(c) Ground sensor scenario
(d) Time of Flight sensor scenario
(e) Simulated encoder and battery discharge model scenario
(f) Camera scenario

**Fig. 13.** ARGoS simulation scenarios, corresponding to the 50-robot experiments, for measuring the execution times using the different E-Puck2's sensors.

the use of computing resources such as arena size, number of robots and lights. Therefore, each sensor was tested in its particular scenario, without modifications, whether robots were using sensors or not, to ensure a fair comparison. We performed 30 independent simulations using one, five, ten, twenty, and fifty robots as well as eleven robot speeds (0 cm/s, 1.5 cm/s, 3 cm/s, 4.5 cm/s, 6 cm/s, 7.5 cm/s, 9 cm/s, 10.5 cm/s, 12 cm/s, 13.5 cm/s, and 15 cm/s), running for 2 simulated hours. The robots in each scenario would be rotating at a given speed while interacting with the different objects in the simulation, depending on the sensor under evaluation.

Fig. 13 shows the ARGoS scenarios designed for testing the execution times when using the different sensors in simulations involving 50 E-Puck2 robots. Some robots were removed to test the other scenarios keeping the rest of elements unaltered. Four walls were placed alongside the robots in the simulation arena to test the proximity and ToF sensors (Figs. 13(a) and 13(d)). Six light sources were used to produce readings in the robot's light sensors (Fig. 13(b)) and eight light sources were used for the camera (Fig. 13(f)). The ground sensor performance was evaluated using a greyed floor (Fig. 13(c)) and no extra objects were needed to evaluate the simulated encoders (Fig. 13(e)). Finally, this same scenario was used to evaluate all the different battery discharge models. In contrast to the former, the robots would rotate at a variable speed (one wheel forward and the other backward and vice versa) in the latter. The range of speeds used was from −15 cm/s (counter clockwise) to 15 cm/s (clockwise) changing in steps of 0.1 cm/s. This is relevant to test the extra computing resources needed to calculate the interpolation points at different speeds.

As each researcher's hardware configuration is going to be different, we report the values obtained using the equipment described at the beginning of Section 4, to be used as a reference of the expected efficiency when using each sensor in the E-Puck2 plug-in.

### 4.3.2. Sensors

Fig. 14 shows the mean execution times and standard deviation obtained from simulations where the robots were using the sensor under evaluation, compared with the base case where all sensors are inactive. Each mean value was calculated from 330 independent simulations to increase the accuracy of the measures (30 simulations using eleven different robot speeds). Note

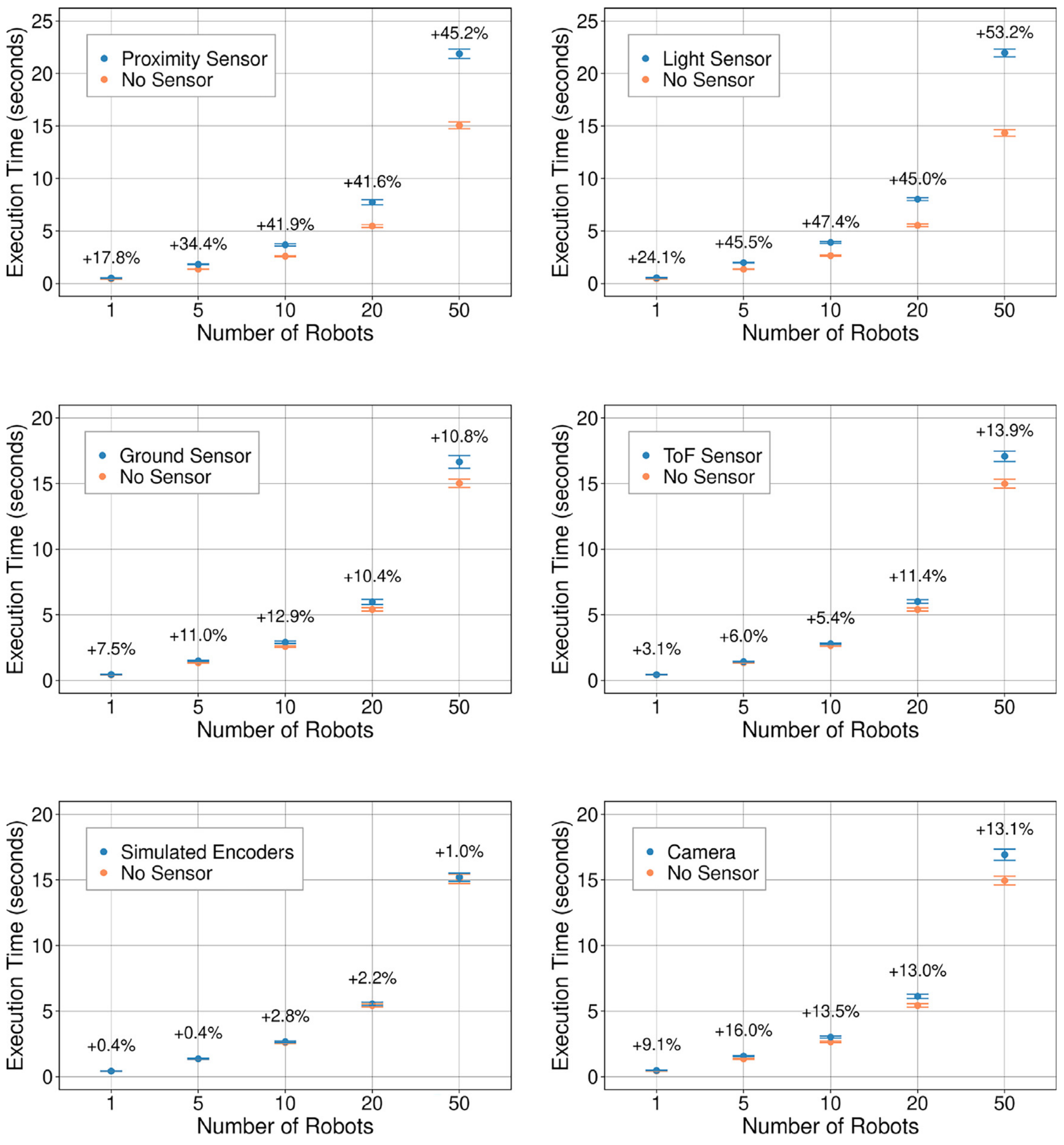


**Fig. 14.** Mean execution times in seconds and standard deviation when using the proposed sensors in simulations with different number of robots.

that the mean execution times when no sensor is active differs through the experiments due to the fact that each scenario is different, independently of the sensors being tested. For example, there are always six light objects in the light sensor experiment even when no light sensor is active. However, there are not light objects in the ToF experiment.

We can see that using proximity sensors results in a maximum 45% longer simulations for fifty robots and that simulations using light sensors are up to 53% slower (50 robots). The increment of the execution time observed is similar beyond ten robots using proximity sensors (40%–45%), although the time values in seconds are different. Simulations with robots using light sensors have also shown a comparable time increment for five, ten, and twenty robots (around 45%).

Regarding the computational resources required after activating the ground sensor in ARGoS simulations, it can be seen that when using one robot, there is a small increment of the execution time (7.5%) which represents 30 negligible milliseconds. However, for five, ten, twenty, and fifty robots, the average execution times are about 10%–13% slower, not negligible but much less than some of the other sensors.

Simulation of fifty robots using the ToF sensor are 14% slower than the case in which it is inactive, while one robot is only 3% slower. We can see that simulation times when using the ToF sensor, scale accordingly with the number of robots as expected. The performance test done for the simulated encoders showed that this sensor has a little effect on the simulation times showing an maximum increment of 2.8%. This is mainly because there is no ray calculations involved to detect intersections with other objects in the arena.

Experiments involving the new camera presented a time increment in the range 9%–16%. Interestingly, the simulation of five

**Table 6**
Mean execution times in seconds and standard deviation when using the proposed battery discharge models in simulations with different number of robots, compared with the case base where no battery model was used.

| Robots | None | | Simple | | | Linear | | | Cubic | | | Approximated | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | SD | Mean | SD | Slower | Mean | SD | Slower | Mean | SD | Slower | Mean | SD | Slower |
| 1 | 0.43 | 0.01 | 0.43 | 0.01 | 0.3% | 0.43 | 0.01 | 0.6% | 0.44 | 0.01 | 1.8% | 0.43 | 0.01 | 0.7% |
| 5 | 1.36 | 0.03 | 1.37 | 0.03 | 1.2% | 1.38 | 0.04 | 1.7% | 1.41 | 0.04 | 4.2% | 1.38 | 0.04 | 1.6% |
| 10 | 2.60 | 0.06 | 2.63 | 0.07 | 1.2% | 2.65 | 0.07 | 2.1% | 2.70 | 0.07 | 3.8% | 2.64 | 0.07 | 1.8% |
| 20 | 5.39 | 0.13 | 5.37 | 0.13 | −0.4% | 5.34 | 0.14 | −0.9% | 5.40 | 0.15 | 0.2% | 5.37 | 0.14 | −0.3% |
| 50 | 15.10 | 0.34 | 14.84 | 0.37 | −1.7% | 15.38 | 0.39 | 1.8% | 15.54 | 0.44 | 2.9% | 15.26 | 0.41 | 1.0% |

robots has shown the biggest percentage difference in execution times, although the absolute values (in seconds) are according to the expected. We believe that it is mainly due to the limited precision of our experimental setup when measuring execution times since 3% represents here, 4 hundredths of a second.

### 4.3.3. Battery

To asses the complexity of the battery sensor, we have conducted 30 simulations per discharge model, for one, five, ten, twenty and fifty robots. Then, we have compared the obtained values with the same scenarios where no battery consumption has been measured. Table 6 shows the average results from the simulations where we can see that the Simple model is at the most 1.2% slower than the case without battery, the Linear model presents a maximum 2.1% longer execution times, the Cubic model is up to 4.2% slower, and the Approximated model, up to 1.8%.

We have observed experiments in which the robots using battery models were faster that the base simulations (up to 1.7% faster) which *a priori* is something that ought not to happen. We believe that since we are dealing with so small time values (tenths and hundredths of second), the obtained results are falling into the precision of our experiments, not to say possible optimisations done by the C++ compiler and at microprocessor level. All in all, we can see that the impact of using any of the battery models is almost negligible, at least for simulations up to fifty robots, especially if we compare the obtained values with the other sensors. Cubic model is confirmed to be the more accurate model (see Section 4.2) but at the price of having slightly longer simulations.

## 5. Conclusion

In this article we have presented a plug-in for the ARGoS simulator to perform experiments using the E-Puck2 robot. We have described the new graphic model for the robot and the implementation of its sensors, tested them on different number of robots, and studied the results obtained to address their accuracy and how much they impact on execution times. We have modelled the new robot body and added the existing LEDs, updated the positions and detection response of the proximity and light sensors. We have also adapted the existing ground sensors to the E-Puck2 characteristics and tested them in two scenarios featuring different ground patterns.

Additionally, we have implemented the new distance (Time of Flight) and simulated encoder sensors and calibrated them using data from the actual robots. We have observed some speed deviations and wheel spin in the robots that resulted in a small difference between the simulated and actual trajectories. Using the ARGoS' coloured perspective camera we have implemented the specific E-Puck2 camera taking into account its resolution and aperture. We have conducted two experiments to measure all those values using real robots and validated them through simulations. We have also proposed four different battery discharge models based on the data collected from 32 experiments (2 different robots and 16 speeds) to obtain the actual discharge curves. They were used to model the proposed battery models taking into account different levels of accuracy and efficiency (computing times).

Our results, after testing the implemented sensors and comparing their behaviours with the real robots, show accurate models based on the response curves of the Time of Flight, ground, proximity and light sensors. The simulated encoders worked similarly as in the E-Puck2 robots while the representation of the objects observed by the onboard camera were placed in the corresponding 2D coordinates with precision. From the study of the four proposed battery discharge models, Simple, Linear, Cubic, and Approximated, we have obtained different accuracy metrics (MSE) in accordance with the complexity of each model, i.e. the more complex the more accurate.

We have evaluated the execution times of the simulations when using the proposed sensors as a metric of computing efficiency. In general, the simulations were longer when there were more robots in the scenario, as expected. Some sensors presented a negligible influence in execution times, e.g. simulated encoders, while the sensors internally using ray traces had a higher impact, e.g. light sensors. Finally, after evaluating the different battery discharge models, we can say that none represented a high impact on the simulation times, especially if they are compared with the sensors that use rays to calculate intersections, e.g. proximity and light sensors.

As a matter of future work we would like to further test our plug-in and evaluate the possibility of adding other external extension boards such as the cliff sensor. We plan to make the plug-in source code freely available at https://gitlab.uni.lu/adars/e-puck2 to be used by researchers in their experiments involving E-Puck2 robots simulated on ARGoS.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request

## Acknowledgements

We would like to thank the reviewers for their thoughtful comments and efforts towards improving our manuscript. This work is supported by the Luxembourg National Research Fund (FNR) – ADARS Project, ref. C20/IS/14762457. The experiments presented in this paper were carried out using the SwarmLab facility of the FSTM/DCS.

## Appendix. Discharge model algorithms

See Algorithms 1–4.

**Algorithm 1** Simple Model.

```
function SIMPLE($\Delta_t$, $\Delta_x$, $CurrentCharge$)
    if $\Delta_x = 0$ then                                              ▷ Static robot
        $NextCharge \leftarrow CurrentCharge + m_0 \times \Delta_t$
    else                                                                ▷ Moving robot
        $NextCharge \leftarrow CurrentCharge + m_1 \times \Delta_t$
    end if
    return $NextCharge$                                                 ▷ Next battery charge value
end function
```

**Algorithm 2** Linear Model.

```
function LINEAR($\Delta_t$, $\Delta_x$, $CurrentCharge$)
    $Speed \leftarrow \frac{\Delta_x}{\Delta_t}$                        ▷ Current robot speed
    $Speed_L \leftarrow \lfloor Speed \rfloor$                          ▷ Lower speed line
    $m_L \leftarrow Line(Speed_L)$                                      ▷ Slope for $Speed_L$
    $Next_L \leftarrow CurrentCharge + m_L \times \Delta_t$
    $Speed_H \leftarrow \lceil Speed \rceil$                            ▷ Upper speed curve
    if $Speed_L = Speed_H$ then
        $NextCharge \leftarrow Next_L$                                  ▷ No interpolation needed
    else
        $m_H \leftarrow Line(Speed_H)$                                  ▷ Slope for $Speed_H$
        $Next_H \leftarrow CurrentCharge + m_H \times \Delta_t$
        $\omega \leftarrow \frac{Speed-Speed_L}{Speed_H-Speed_L}$       ▷ Interpolation factor
        if $Next_L < Next_H$ then
            $NextCharge \leftarrow Next_L + \omega(Next_H - Next_L)$
        else
            $NextCharge \leftarrow Next_H + \omega(Next_L - Next_H)$
        end if
    end if
    return $NextCharge$                                                 ▷ Next battery charge value
end function
```

**Algorithm 3** Cubic Model.

```
function CUBIC($\Delta_t$, $\Delta_x$, $CurrentCharge$)
    $Speed \leftarrow \frac{\Delta_x}{\Delta_t}$                                ▷ Current robot speed
    $Speed_L \leftarrow \lfloor Speed \rfloor$                                  ▷ Lower speed curve
    $P_L \leftarrow Poly(Speed_L)$                                              ▷ Polynomial for $Speed_L$
    $Tmax_L \leftarrow MaxT(Speed_L)$
    $t_L \leftarrow FindRoot(P_L, CurrentCharge)$                               ▷ Finds t
    if $t_L \geq Tmax_L$ then
        $Next_L \leftarrow 0$                                                   ▷ Maximum t was reached
    else
        $t_L \leftarrow t_L + \Delta_t$
        $Next_L \leftarrow P_L(t_L)$                                            ▷ Next charge value for $t_L$
    end if
    $Speed_H \leftarrow \lceil Speed \rceil$                                    ▷ Upper speed curve
    if $Speed_L = Speed_H$ then
        $NextCharge \leftarrow Next_L$                                          ▷ No interpolation needed
    else
        $P_H \leftarrow Poly(Speed_H)$                                          ▷ Polynomial for $Speed_H$
        $Tmax_H \leftarrow MaxT(Speed_H)$
        $t_H \leftarrow FindRoot(P_H, CurrentCharge)$                           ▷ Finds t
        if $t_H \geq Tmax_H$ then
            $Next_H \leftarrow 0$                                               ▷ Maximum t was reached
        else
            $t_H \leftarrow t_H + \Delta_t$
            $Next_H \leftarrow P_H(t_H)$                                        ▷ Next charge value for $t_H$
        end if
        $\omega \leftarrow \frac{Speed - Speed_L}{Speed_H - Speed_L}$           ▷ Interpolation factor
        if $Next_L < Next_H$ then
            $NextCharge \leftarrow Next_L + \omega(Next_H - Next_L)$
        else
            $NextCharge \leftarrow Next_H + \omega(Next_L - Next_H)$
        end if
    end if
    return $NextCharge$                                                         ▷ Next battery charge value
end function
```

**Algorithm 4** Approximated Model.

```
function APPROXIMATED($\Delta_t$, $\Delta_x$, $CurrentCharge$)
    $Speed \leftarrow \frac{\Delta_x}{\Delta_t}$                                ▷ Current robot speed
    $Speed_L \leftarrow \lfloor Speed \rfloor$                                  ▷ Lower speed segment
    $m_L, h_L \leftarrow Line(Speed_L)$                                         ▷ Segment for $Speed_L$
    $t_L \leftarrow \frac{CurrentCharge - h_L}{m_L}$                            ▷ Finds t
    $Next_L \leftarrow m_L(t_L + \Delta_t) + h_L$                               ▷ Next charge
    $Speed_H \leftarrow \lceil Speed \rceil$                                    ▷ Upper speed segment
    if $Speed_L = Speed_H$ then
        $NextCharge \leftarrow Next_L$                                          ▷ No interpolation needed
    else
        $m_H, h_H \leftarrow Line(Speed_H)$                                     ▷ Segment for $Speed_H$
        $t_H \leftarrow \frac{CurrentCharge - h_H}{m_H}$                        ▷ Finds t
        $Next_H \leftarrow m_H(t_H + \Delta_t) + h_H$                           ▷ Next charge
        $\omega \leftarrow \frac{Speed - Speed_L}{Speed_H - Speed_L}$           ▷ Interpolation factor
        if $Next_L < Next_H$ then
            $NextCharge \leftarrow Next_L + \omega(Next_H - Next_L)$
        else
            $NextCharge \leftarrow Next_H + \omega(Next_L - Next_H)$
        end if
    end if
    return $NextCharge$                                                         ▷ Next battery charge value
end function
```

## References

[1] C. Symeonidis, N. Nikolaidis, Chapter 18 - simulation environments, in: A. Iosifidis, A. Tefas (Eds.), Deep Learning for Robot Perception and Cognition, Academic Press, 2022, pp. 461–490, http://dx.doi.org/10.1016/B978-0-32-385787-1.00023-3.

[2] Y.-H. Yoo, M. Ahmed, S. Bartsch, F. Kirchner, Realistic simulation of extraterrestrial legged robot in trade-off between accuracy and simulation time, in: IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society, 2010, pp. 1603–1608, http://dx.doi.org/10.1109/IECON.2010.5675443.

[3] L. Pitonakova, M. Giuliani, A. Pipe, A. Winfield, Feature and performance comparison of the V-REP, gazebo and argos robot simulators, in: M. Giuliani, T. Assaf, M.E. Giannaccini (Eds.), Towards Autonomous Robotic Systems, Springer International Publishing, Cham, 2018, pp. 357–368.

[4] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L.M. Gambardella, M. Dorigo, ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems, Swarm Intell. 6 (4) (2012) 271–295, http://dx.doi.org/10.1007/s11721-012-0072-5.

[5] E. Rohmer, S.P.N. Singh, M. Freese, V-REP: A versatile and scalable robot simulation framework, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013, pp. 1321–1326, http://dx.doi.org/10.1109/IROS.2013.6696520.

[6] N. Koenig, A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, in: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Vol. 3, IEEE, pp. 2149–2154, http://dx.doi.org/10.1109/IROS.2004.1389727.

[7] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, A. Martinoli, The E-puck, a robot designed for education in engineering, in: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, Vol. 1, IPCB: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65.

[8] A. Gautam, A.U. Saxena, P. Mall, S. Mohan, Positioning multiple mobile robots for geometric pattern formation: An empirical analysis, in: 2014 Seventh International Conference on Contemporary Computing (IC3), 2014, pp. 607–612, http://dx.doi.org/10.1109/IC3.2014.6897242.

[9] B. Khaldi, F. Cherif, Swarm robots circle formation via a virtual viscoelastic control model, in: 2016 8th International Conference on Modelling, Identification and Control (ICMIC), 2016, pp. 725–730, http://dx.doi.org/10.1109/ICMIC.2016.7804207.

[10] D. Nemec, A. Janota, M. Hruboš, M. Gregor, R. Pirník, Mutual acoustic identification in the swarm of E-puck robots, Int. J. Adv. Robot. Syst. 14 (3) (2017) 1729881417710794, http://dx.doi.org/10.1177/1729881417710794.

[11] M. Shayestegan, S. Din, Fuzzy logic controller for robot navigation in an unknown environment, in: 2013 IEEE International Conference on Control System, Computing and Engineering, 2013, pp. 69–73, http://dx.doi.org/10.1109/ICCSCE.2013.6719934.

[12] A. Noormohammadi-Asl, M. Saffari, M. Teshnehlab, Neural control of mobile robot motion based on feedback error learning and mimetic structure, in: Electrical Engineering (ICEE), Iranian Conference on, 2018, pp. 778–783, http://dx.doi.org/10.1109/ICEE.2018.8472657.

[13] W.-A. Zhang, X. Yang, L. Yu, S. Liu, Sequential fusion estimation for RSS-based mobile robots localization with event-driven WSNs, IEEE Trans. Ind. Inform. 12 (4) (2016) 1519–1528, http://dx.doi.org/10.1109/TII.2016.2585350.

[14] M.M. Almasri, A.M. Alajlan, K.M. Elleithy, Trajectory planning and collision avoidance algorithm for mobile robotics system, IEEE Sens. J. 16 (12) (2016) 5021–5028, http://dx.doi.org/10.1109/JSEN.2016.2553126.

[15] T. Kargar Tasooji, H.J. Marquez, Cooperative localization in mobile robots using event-triggered mechanism: Theory and experiments, IEEE Trans. Autom. Sci. Eng. 19 (4) (2022) 3246–3258, http://dx.doi.org/10.1109/TASE.2021.3115770.

[16] Wulandari, M.H. Arrazi, K. Priandana, Development of landslide victim detection system using thermal imaging and histogram of oriented gradients on E-Puck2 robot, in: 2020 International Conference on Computer Science and Its Application in Agriculture (ICOSICA), 2020, pp. 1–6, http://dx.doi.org/10.1109/ICOSICA49951.2020.9243244.

[17] T.K. Tasooji, H.J. Marquez, Decentralized event-triggered cooperative localization in multirobot systems under random delays: With/without timestamps mechanism, IEEE/ASME Trans. Mechatronics (2022) 1–13, http://dx.doi.org/10.1109/TMECH.2022.3203439.

[18] D.H. Stolfi, G. Danoy, Optimising autonomous robot swarm parameters for stable formation design, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 1281–1289, http://dx.doi.org/10.1145/3512290.3528709.

[19] D. Floreano, S. Mitri, J. Hubert, E-puck, in: S. Nolfi, M. Mirolli (Eds.), Evolution of Communication and Language in Embodied Agents, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 303–306, http://dx.doi.org/10.1007/978-3-642-01250-1_19.

[20] S. Mitri, D. Floreano, L. Keller, Evolutionary conditions for the emergence of communication, in: S. Nolfi, M. Mirolli (Eds.), Evolution of Communication and Language in Embodied Agents, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 123–134, http://dx.doi.org/10.1007/978-3-642-01250-1_8.

[21] Y.M. Adam, N. Binti Sariff, N.A. Algeelani, E-puck mobile robot obstacles avoidance controller using the fuzzy logic approach, in: 2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE), 2021, pp. 107–112, http://dx.doi.org/10.1109/ICSCEE50312.2021.9497939.

[22] H.J.M. Lopes, D.A. Lima, Cellular automata in path planning navigation control applied in surveillance task using the E-puck architecture, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2020, pp. 1117–1122, http://dx.doi.org/10.1109/SMC42975.2020.9283048.

[23] A.G. Millard, R. Joyce, J.A. Hilder, C. Fleeriu, L. Newbrook, W. Li, L.J. McDaid, D.M. Halliday, The pi-puck extension board: A raspberry pi interface for the E-puck robot platform, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 741–748, http://dx.doi.org/10.1109/IROS.2017.8202233.

[24] J.M. Allen, R. Joyce, A.G. Millard, I. Gray, The pi-puck ecosystem: Hardware and software support for the E-puck and E-Puck2, in: M. Dorigo, T. Stützle, M.J. Blesa, C. Blum, H. Hamann, M.K. Heinrich, V. Strobel (Eds.), Swarm Intelligence, Springer International Publishing, Cham, 2020, pp. 243–255.

[25] V. Strobel, E. Castelló Ferrer, M. Dorigo, Blockchain technology secures robot swarms: A comparison of consensus protocols and their resilience to Byzantine robots, Front. Robot. AI 7 (2020) http://dx.doi.org/10.3389/frobt.2020.00054.

[26] A. Neale, A.G. Millard, Integration and robustness analysis of the buzz swarm programming language with the pi-puck robot platform, in: S. Pacheco-Gutierrez, A. Cryer, I. Caliskanelli, H. Tugal, R. Skilton (Eds.), Towards Autonomous Robotic Systems, Springer International Publishing, Cham, 2022, pp. 223–237.

[27] L. Garattoni, G. Francesca, A. Brutschy, C. Pinciroli, M. Birattari, Software Infrastructure for E-Puck (and TAM) Technical Report TR/IRIDIA/2015-004, Tech. rep., 2015.

[28] F. Mondada, M. Bonani, F. Riedo, M. Briod, L. Pereyre, P. Retornaz, S. Magnenat, Bringing robotics to formal education: The thymio open-source hardware robot, IEEE Robot. Autom. Mag. 24 (1) (2017) 77–85, http://dx.doi.org/10.1109/MRA.2016.2636372.

[29] S. Sarparast, D. Tarapore, Thymio simulator based on ARGoS framework, 2023, URL https://github.com/resilient-swarms/thymio.

[30] S. Magnenat, P. Rétornaz, M. Bonani, V. Longchamp, F. Mondada, ASEBA: A modular architecture for event-based control of complex robots, IEEE/ASME Trans. Mechatronics 16 (2) (2011) 321–329, http://dx.doi.org/10.1109/TMECH.2010.2042722.

[31] C. Pinciroli, M.S. Talamali, A. Reina, J.A.R. Marshall, V. Trianni, Simulating kilobots within ARGoS: Models and experimental validation, in: M. Dorigo, M. Birattari, C. Blum, A.L. Christensen, A. Reina, V. Trianni (Eds.), Swarm Intelligence, Springer International Publishing, Cham, 2018, pp. 176–187.

[32] M.S. Talamali, A. Saha, J.A.R. Marshall, A. Reina, When less is more: Robot swarms adapt better to changes with constrained communication, Science Robotics 6 (56) (2021) eabf1416, http://dx.doi.org/10.1126/scirobotics.abf1416.

[33] K. Pfister, H. Hamann, Collective decision-making with Bayesian robots in dynamic environments, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 7245–7250, http://dx.doi.org/10.1109/IROS47612.2022.9982019.

[34] Y. Zheng, M. Allwright, W. Zhu, M. Kassawat, Z. Han, M. Dorigo, Swarm construction coordinated through the building material, in: M. Baratchi, L. Cao, W.A. Kosters, J. Lijffijt, J.N. van Rijn, F.W. Takes (Eds.), Artificial Intelligence and Machine Learning, Springer International Publishing, Cham, 2021, pp. 188–202.

[35] Webots, Webots: Open source robot simulator, 2022, URL https://cyberbotics.com/.

[36] A. Dobrokvashina, R. Lavrenov, Y. Bai, M. Svinin, E. Magid, Sensors modelling for servosila engineer crawler robot in webots simulator, in: 2022 Moscow Workshop on Electronic and Networking Technologies (MWENT), 2022, pp. 1–5, http://dx.doi.org/10.1109/MWENT55238.2022.9802400.

[37] C. Li, S. Guo, J. Guo, Study on obstacle avoidance strategy using multiple ultrasonic sensors for spherical underwater robots, IEEE Sens. J. 22 (24) (2022) 24458–24470, http://dx.doi.org/10.1109/JSEN.2022.3220246.

[38] S. Gu, S. Guo, Performance evaluation of a novel propulsion system for the spherical underwater robot (SURIII), Appl. Sci. 7 (11) (2017) http://dx.doi.org/10.3390/app7111196.

[39] GCtronic, GCtronic – electronics and mechatronics, 2022, URL https://www.gctronic.com/.

[40] A. Gutierrez, A. Campo, M. Dorigo, J. Donate, F. Monasterio-Huelin, L. Magdalena, Open E-puck range & bearing miniaturized board for local communication in swarm robotics, in: 2009 IEEE International Conference on Robotics and Automation, 2009, pp. 3111–3116, http://dx.doi.org/10.1109/ROBOT.2009.5152456.

**Dr. Daniel H. Stolfi** received the Ph.D. in Computer Science in 2018 and the M.S. degree in software engineering and artificial intelligence in 2010, both from the University of Malaga, Spain. He awarded an FPU grant from the Spanish Ministry of Education, Culture and Sports in 2013, which allowed him to be a Ph.D. candidate at the LCC department of the aforementioned university. Daniel Stolfi participated in eight research projects and worked for private companies as a software engineer. His current research interests include bio-inspired algorithms, robotic swarms, evolutionary game theory, and computer simulations. Currently, Daniel Stolfi is a Research Associate in the Parallel Computing and Optimisation Group (PCOG) at the Interdisciplinary Centre for Security, Reliability and Trust (SnT) at the University of Luxembourg.

**Dr. Grégoire Danoy** is a Research Scientist at the University of Luxembourg and Deputy-Head of the Parallel Computing and Optimisation Group (PCOG). He received his Industrial Engineer degree in Computer Science from the Luxembourg University of Applied Sciences (IST) in 2003. He obtained his Master in Web Intelligence in 2004 and his Ph.D. in Computer Science in 2008 from the Ecole des Mines of Saint-Etienne, France. His main research interests include artificial intelligence techniques applied to unmanned autonomous systems, satellite communications, vehicular networks, bioinformatics, highperformance and cloud computing. With more than 15 years of experience in Artificial Intelligence and Mobile Networks, Dr. Danoy co-authored more than 140 publications in international scientific conferences and journals in the field.