Inquiry-Based Learning with RoboGen: An Open-Source Software and Hardware Platform for Robotics and Artificial Intelligence

Joshua E. Auerbach, *Member, IEEE*, Alice Concordel, Przemyslaw M. Kornatowski, and Dario Floreano, *Senior Member, IEEE*

***Abstract*—It has often been found that students appreciate hands-on work, and find that they learn more with courses that include a project than those relying solely on conventional lectures and tests. This type of project driven learning is a key component of "Inquiry-based learning" (IBL), which aims at teaching methodology as well as content by incorporating the student as an actor rather than a spectator. Robotics applications are especially well-suited for IBL due to the value of trial and error experience, the multiple possibilities for students to implement their own ideas, and the importance of programming, problem-solving and electro-mechanical skills in real world engineering and science jobs. Furthermore, robotics platforms can be useful teaching media and learning tools for a variety of topics. Here, we present RoboGen: an open-source, web-based, software and hardware platform for Robotics and Artificial Intelligence with a particular focus on Evolutionary Robotics. We describe the platform in detail, compare it to existing alternatives, and present results of its use as a platform for Inquiry-based learning within a master's level course at the Ecole Polytechnique Fédérale de Lausanne (EPFL).**

***Index Terms*—Robotics, Education, Inquiry-Based Learning, Artificial Intelligence, Evolutionary Robotics, 3D-Printing**

I. Introduction

SCIENCE instructors across a diversity of disciplines agree on the pedagogical value of hands-on exercises [16]. However, several authors [22], [32] have drawn attention to certain shortcomings of typical laboratory exercises. Specifically, these authors stress that hands-on components of courses tend to be formulaic, too narrowly focused, and at times outdated.

On the other hand, it has often been found that students appreciate hands-on work, and find that they learn more with courses that include a project than those relying solely on conventional lectures and tests [7]. This type of project driven learning is a key component of "Inquiry-based learning" (IBL), a pedagogical approach developed in the 1960s in the logic of constructivism, which aims at teaching methodology as well as content by incorporating the student as an actor rather than a spectator. In IBL the work is problem or research-based and emphasizes resolution methodology over theoretical content [20]. This allows teaching skills that are important for a researcher and increases student involvement in the project [23]. With such an approach it has been found that students are more enthusiastic, creative, have a strong team spirit [40], and are willing to work longer hours [25], [40]. Moreover, when IBL is applied to scientific research topics, it has been shown to improve students' skills in report writing and technical English [50]. Accordingly, this approach is now widespread in universities and high schools.

Robotics projects are especially well-suited for IBL due to the numerous possibilities for students to implement their own ideas, as well as the importance problem-solving and electro-mechanical skills in real world engineering and science jobs [25]. Furthermore, robotics platforms can be useful teaching media and learning tools for a variety of topics [41]. They can be used to teach programming, allowing the users to see their code working in a visual way, and even be used for grasping social behaviors and emotion [15]. However, existing educational robotics platforms frequently rely on pre-fabricated robots and/or emphasize instruction-based programming of the robot behavior. Therefore, they are not ideal tools for teaching advanced robotics and Artificial Intelligence methods, such as neural networks, and morphological evolution.

Here, we present RoboGen, an open-source, web-based, software and hardware platform for robotics, specifically in the context of its use for Inquiry-based learning within a master's level course at the Ecole Polytechnique Fédérale de Lausanne (EPFL). While capable of supporting more traditional robotics functionality, RoboGen has primarily been designed as a platform for Bio-Inspired Artificial Intelligence techniques [13] including Evolutionary Robotics (ER) [19], [37].

ER provides a framework for overcoming the biases and limitations of human engineers in designing robots by leveraging the use of Evolutionary Algorithms (EAs): a class of heuristic search/optimization algorithms loosely inspired by natural evolution. Most typically in Evolutionary Robotics an EA is applied to optimize the control policy of an existing (often simulated) robot for a pre-defined task. However, EAs may also be used to optimize a robot's physical form including its mechanical design as well as its use of sensors and actuators. The idea of using EAs to optimize the physical form of a robot (i.e. its morphology) in addition to its controller dates back at least to Karl Sims' pioneering study on the evolution of virtual creatures in 1994 [44]. This work has been followed by many additional studies [3], [8], [12], [17], [28], [30], [38] in which

Joshua Auerbach is with the Dept. of Computer Science and Innovation at Champlain College, Burlington, VT, USA.
E-mail jauerbach@champlain.edu
All authors were with the Laboratory of Intelligent Systems at Ecole Polytéchnique Fédérale de Lausanne, Lausanne, Switzerland.
E-mails alice.concordel@alumni.epfl.ch, przemyslaw.kornatowski@epfl.ch, dario.floreano@epfl.ch

Fig. 1. A graphical overview of evolving robots with RoboGen. Abstract robot representations (here depicted as a strand of DNA) define the brains and bodies of robots, which are evaluated in the physics-based simulation engine. After many iterations of selection and reproduction based on the robots' simulated performance, the best robot is fabricated via the use of 3D-printing, an Arduino based microcontrol board, and other off-the-shelf electronic components.

aspects of simulated and/or physical robots' morphologies have been evolved. This approach has the major advantage of discovering body plans and sensorimotor configurations uniquely suited for the machine's task environment.

The majority of these studies have been confined to simulation [3], [8], [28], [44], oftentimes not even professing to evolve robots per se, but rather "virtual creatures" [28], [44], where assumptions of physical realism have been relaxed. Others have tried to faithfully model real physics in their simulations, and some, starting with [30], have evolved robot morphologies and controllers in simulation and then manufactured these via 3D-Printing.

It is this last approach that has most directly inspired RoboGen. While in the year 2000, 3D-printers were expensive, niche products, and physical simulation was slow, a plethora of inexpensive printers now exists[1], and many people have access to fast computers with multiple computing cores, which are especially useful for EAs—often called "embarrassingly parallelizable" [46]. These reasons, as well as the widespread availability of low-cost, off-the-shelf electronics components, standardized microcontroller programming environments, such as Arduino[2], as well as the adoption of the web browser as a cross platform application container[3] have all led to the ability to make such a system widely available.

Due to the comprehensive nature of the platform, and the many disciplines it draws upon, RoboGen offers a unique experimental test-bed for users to explore a variety of topics including computer programming, evolutionary algorithms, artificial neural networks, physical simulation, 3D printing, electro-mechanical assembly, and embedded processing. In addition, it is a fully open-source platform, relying only on free software and low-cost, off-the-shelf components.

The rest of this paper is structured as follows: Section II presents an overview of related work on robotics platforms, specifically for educational uses. In Section III the RoboGen platform is described in greater detail—including the specific ways it is used within a master's level course. Section IV then presents the results of using RoboGen for class projects at EPFL in the Spring of 2016: examples of what students were able to accomplish with the platform, as well as their feedback about the project solicited through a questionnaire. This is followed by a discussion of the efficacy of RoboGen in this context, before closing with conclusions and directions for future work.

[1]See for example http://3dforged.com/good-cheap-3d-printers/

[2]http://arduino.cc

[3]And the maturation of tools such as Emscripten (https://github.com/kripken/emscripten) for compiling native code to run efficiently within web browsers.

II. Related Work

While RoboGen offers many novel features, it is certainly not the first robotics platform to be used for education, nor is it the first platform for Bio-Inspired AI or Evolutionary Robotics. In this section, the most relevant robotics platforms for education are described, and differences between those platforms and RoboGen are discussed. Unfortunately describing all such platforms is beyond the scope of this paper, and thus focus is paid to those most relevant to the goals of RoboGen: being full-featured, open-source, low cost, capable of simulating and fabricating a diversity of morphologies, and being an appropriate platform for teaching electro-mechanical skills and Bio-Inspired AI techniques to a diversity of students.

A useful (if slightly outdated) resource for additional possibilities is provided by Ruzzenente et al.'s [43] review of robotics platforms for tertiary education. A more recent review of modular robotics kits for classroom education is provided by Takács et al. [48].

The **e-puck**[4] [34] is a two-wheel, differential drive mobile robot, which includes a set of sensors, LEDs, microphones, and a loudspeaker. There also exist several add-on modules, which can be attached to the e-puck to extend its capabilities. The e-puck and its extensions are tools for teaching mobile robotics, real-time programming, and embedded systems. Due to the simple two-wheel, differential drive control of the e-puck, and its similarity to the older Khepera robot [36], which formed the basis of many pioneering studies in ER [14], [24], [33], the e-puck is also used to teach the fundamental techniques of ER such as neural networks and fitness function design. Additionally, it has successfully been used to study distributed systems, control theory, signal processing, behavior-based robotics, and swarm robotics [9], [35].

[4]http://www.e-puck.org/

While the e-puck is a well established robotics platform for both education and research, and it is still utilized in Evolutionary Robotics studies [21], it is severely limited by its morphology. The two-wheeled, differential drive design is effective for robotics tasks consisting of movement over flat terrains, but it does not support more biologically inspired modes of locomotion, which are useful in complex environments (e.g. moving over rough or uneven terrains). Moreover, while the morphology of an e-puck may be altered to some extent with different extensions, it has not been designed to allow for a diversity of body plans, and is therefore more appropriate for studying the behavior of an individual or a swarm, as opposed to advanced concept such as morphological computation [39]. Additionally, while accessible to well-funded universities, the e-puck is too expensive for many educational applications.

A lower cost differential drive mobile robot, **Thymio**[5] [41], may be a better option for those on a tight budget. It features tactile buttons, a diverse set of sensors, a loudspeaker, a hook for a trailer, and a pencil holder. In addition, the wheels can be used for other purposes (e.g. cable actuation), and several parts of the robot feature protruding knobs to which LEGO parts can be attached.

Thymio aims at teaching the logic of robotic programming to a wide public, including young children [26], [42]. For this purpose, several modalities are available for programming: visual programming, which relies only on graphical representations of desired functions, 'Blockly' programming—a hybrid between visual and textual programming, developed by Google, and finally, Aseba Studio—a text based IDE using a simple language that also allows a higher level of control. Thymio has been designed to make the hardware as easy to use as possible, and thus the features are modular and are provided as self-contained elements. This is great for introducing students to robotics, but less appropriate for higher-level engineering students, who need a deeper understanding of the electronics and mechanics.

Thymio's low cost, multiple programming options, and ability to attach LEGO components make it more suitable than the e-puck for teaching many bio-inspired robotics concepts. However, it is still not capable of producing the diversity of morphologies that RoboGen can produce, nor has it been designed to have a tightly integrated simulator and a diversity of bio-inspired algorithms as RoboGen has.

Using the versatility of LEGO parts, the **Mindstorms**[6] platform is based on a programmable brick that includes the microcontroller, inputs for sensors, outputs for motors, and a built-in speaker. The sensors and actuators come as blocks that can be placed on the body of the robot. The robots can be programmed with a computer via a diversity of programming interfaces (both graphical and textual), or controlled directly with a smartphone app and a bluetooth connection.

The Mindstorms kit offers many more morphological and control possibilities than the other platforms mentioned above. Additionally, simulation engines have been created for it[7]. Moreover, the platform is relatively affordable and the parts may be purchased separately to expand its capabilities. However, the existing simulation engines are not open source, nor freely available in the web browser, and while many parts are available, they are pre-defined. Finally, while Evolutionary Algorithms have been applied to evolving LEGO robots [31], no support for such AI techniques is included in the the Mindstorms kit, and manually designing control strategies for complex robots is an onerous task.

Another relevant option is the **LittleBits** platform[8], which consists of a series of modules, called *bits*, that can be assembled together with magnets. Each *bit* is an electronic component with a specific function: sensing, actuation, lights, speakers, power, and logic gates. The platform has been designed such that most of the mechanical structure should be created with paper, cardboard or other household items, while a mounting board is provided to hold the electronics. This "crafty" side is quite interesting for stimulating the creativity of young children.

In its original form, the platform is not programmable, relying instead on switches, sliders, knobs etc., connected to the parts that need to be controlled. This is akin to electronic circuit design kits, and avoids the abstraction level needed to understand code, thus making it easier for beginners. For more advanced users, a new *bit* includes an Arduino microcontroller, and can be programmed in a traditional way. While the platform shares some of the goals of RoboGen: being affordable and appropriate for students of diverse ages, it has primarily been tailored for younger students. Additionally, LittleBits includes neither simulation software, nor the extensive bio-inspired components that RoboGen does.

Most of the aforementioned robotics platforms provide the means for users to take either an existing robot (e.g. the Thymio), or one that they have manually designed (with e.g. LEGO Mindstorms) and program controllers using either a graphical programming language such as LabVIEW [49], or a more traditional, textual programming language. While, manually designing and programming robots in a similar way is possible with RoboGen, its novelty really stems from its ability to forgo such manual design and instead leverage Bio-Inspired AI techniques such as Evolutionary Robotics for the automatic design of robot morphologies and/or controllers.

While other projects have attempted to make ER easily accessible to students, these have been limited in scope and do not allow one to go from conception of a task, to evolving bodies and brains all the way to 3D-printing and building a physical version of the evolved robot. For instance, Ludobots[9] is a freely available, Evolutionary Robotics MOOC (massive open online course) created by Josh Bongard and colleagues, which is hosted on http://www.reddit.com. Like RoboGen, it is designed for students of a wide range of experience level. Ludobots guides students through a set of increasingly challenging exercises, starting with a simple web-based, "connect-the-dots"-bot application, through a series of programming exercises to the final assignment which yields a C++ based

[5]https://www.thymio.org/
[6]http://www.lego.com/mindstorms/
[7]E.g. Virtual Brick: http://robotvirtualworlds.com/virtualbrick/
[8]http://littlebits.cc/
[9]https://www.reddit.com/r/ludobots/

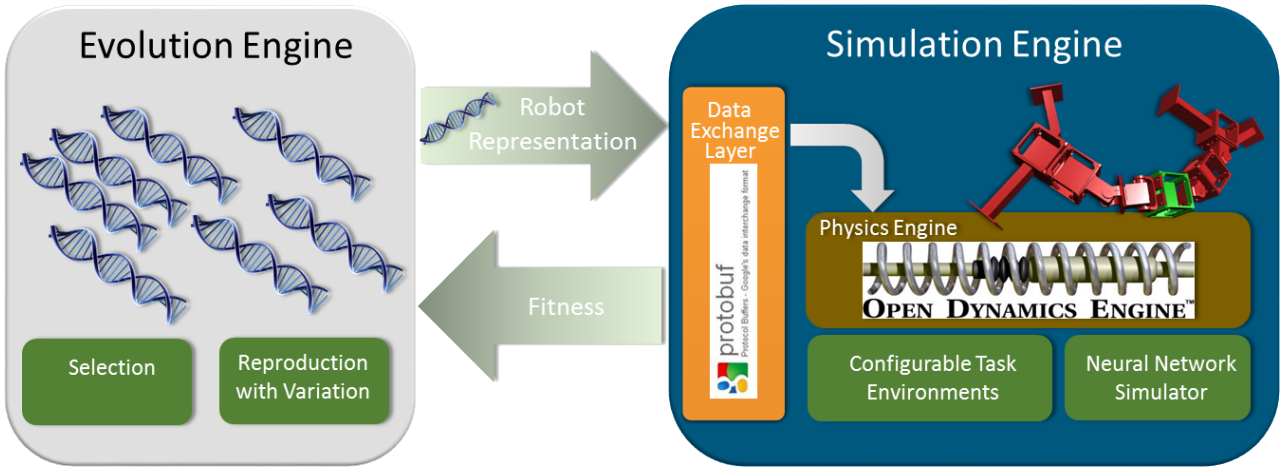

Fig. 2. Overview of the RoboGen software platform. The evolution engine (left) runs an evolutionary algorithm operating on a population of abstract representations of robot brains and bodies (depicted here as strands of DNA). Each of these robot representations is sent to the simulation engine (right), which first translates each abstract representation into a concrete, physically simulated robot with properties tuned to those of the real hardware, then simulates this robot being controlled by a neural network inside of a highly customizable virtual environment, and finally returns a measure of the robot's performance, known as a fitness function. The evolution engine in turn uses these fitness values to select good robots for reproduction.

ER platform. The course is designed to allow users to fork and modify all aspects, which makes it quite open-ended, but in the end, a lot of programming is required to explore more advanced topics. Most importantly, unlike RoboGen, there is no provided mechanism for transferring robots to reality.

In summary, while a number of robotics platforms have previously been created for educational purposes, none offer RoboGen's unique combination of open source software and hardware, which allows students to design and evolve robot bodies and brains in simulation, and then fabricate these bio-inspired, articulated robots at low cost.

III. Methods and Materials

In this section, the RoboGen platform is presented in greater detail, followed by a description of how it has been used as a platform for master's level course projects at EPFL. Section IV then presents results of using RoboGen in this context.

As previously mentioned, RoboGen is an open source hardware and software platform. The software platform is comprised of an evolution and a simulation engine, as depicted in Fig. 2. Once one or more robots have been successfully evolved or designed, it/they can be physically fabricated by 3D printing the open source design files and assembling the robot(s) with low cost, off-the-shelf electronics components (Fig. 1). Since the capabilities of the software platform have been engineered to support the ultimate creation of physical robots, it makes the most sense to first describe these robots—the components they are comprised of and how these components are assembled together—after which further details of the software platform will be presented.

A. RoboGen Robots

1) Robot Bodies: RoboGen robots are made up of predefined and parameterized body parts, each of which is composed of one or more 3D-printed structural elements in possible combination with off-the-shelf electronic components: sensors, actuators, and an Arduino-based microcontroller. In order to support a large diversity of morphological possibilities, these parts can be assembled together into a variety of configurations. The body parts are assembled together by means of plates (male) that slide into connection slots (female) (refer to Fig. 3). Additionally, the system includes body parts with modifiable parameters (e.g. dimensions and angles), which allow for leveraging the customization capabilities offered by 3D printing beyond what would be possible with an off-the-shelf kit of building blocks such as LEGO.

RoboGen robots are articulated modular structures, which can be represented as a tree (Fig. 4), where nodes represent body parts and links represent connections between parts. All RoboGen robots begin with the *CoreComponent*, which serves as the root of the tree, and houses the microntroller and battery capable of delivering the required $5V$[10]. This part is designed such that the microcontroller board and battery fit snugly and remain in place as the robot moves. It also includes a slot for the USB port of the microcontroller, which is used to upload the control code—including an evolved artificial neural network. Additionally, the microconctroller provides sensory information from an onboard 6-axis inertial measurement unit (IMU), which provides proprioceptive feedback.

The microcontroller and all other electronic elements of RoboGen have been chosen to be low-cost and readily available for online purchase so that the robots can be manufactured by anyone in the world. The reference information for all components can be found on http://robogen.org/.

The other body parts fall into four categories: **bricks**, **joints**, **sensors**, and **connectors** as shown in Fig. 3. The **bricks** include the *CoreComponent* and *FixedBricks*: while the *CoreComponent* houses the microcontroller and battery, the *FixedBricks* are empty, but otherwise these parts are identical.

[10]In practice the battery pack we use (Turnigy nano-tech 260mah 2S 35 70C Lipo Pack (http://www.hobbyking.com/hobbyking/store/__26742__Turnigy_nano_tech_260mah_2S_35_70C_Lipo_Pack.html) is used in conjuction with a voltage regulator to convert from $7.6V$ to the $5V$ that the microcontroller requires.

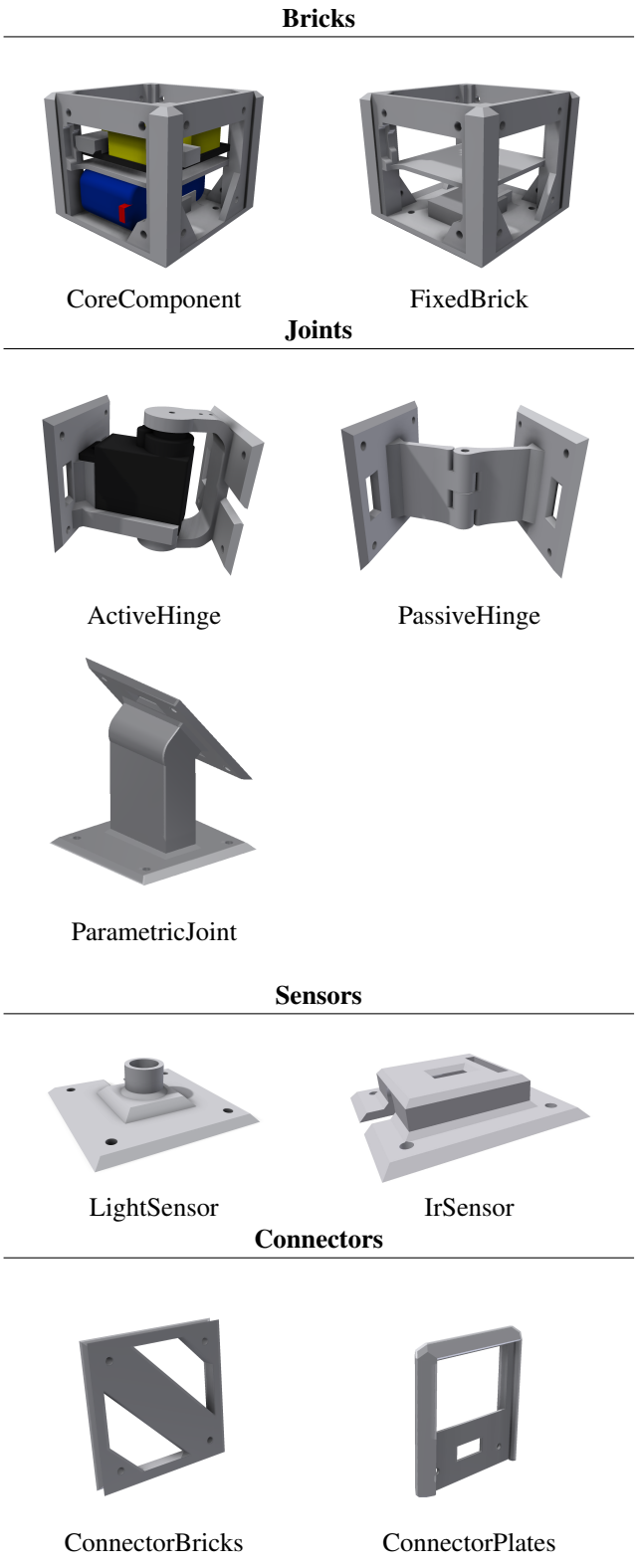

Fig. 3. CAD renderings of the main RoboGen parts.

Each *FixedBrick* measures 41x41mm horizontally, is 35mm high, and has (female) connection slots on its four sides[11].

There are currently three different types of **joints**: *ParametricJoints*, *ActiveHinges*, and *PassiveHinges*[12]. *ParametricJoints* are rigid connections with a variable length and angle that allow the robot's basic shape to not be confined by a Cartesian grid. *PassiveHinges* include a passive joint with one degree of freedom composed of a brass axle for minimal friction. *ActiveHinges* include an active joint with one degree of freedom actuated by a servomotor. All joints contain two (male) connections plates at opposite ends.

There are two types of **sensor** body parts: *LightSensors* and *IrSensors*, which can provide the robot with additional sensory feedback. The former are analog photoresisteros, which output a signal proportional to the perceived light intensity. They are directional to enable light following behavior and calibrated to discount for ambient light. The latter are infrared time-of-flight sensors. They use I2C communication and are not susceptible to ambient light. Both of these sensors connect via a single (male) plate, and so will always be leaf-nodes of the body tree.

Since bricks have female attachment slots and all other components have male attachment plates, it is also necessary to have **connector** parts. A *ConnectorBrick* slides into a slot on either side and connects two bricks flush against one another, while a *ConnectorPlate* slides over two plates and holds them together with a small gap. However, the RoboGen software will automatically determine when these parts are necessary, and so they are not part of the body tree defining a body (q.v. Fig. 4), and thus for all intents and purposes can be ignored until a robot is physically fabricated.

2) Robot Brains: In RoboGen, the brains of robots are Artificial Neural Networks (ANNs). The body of a robot implicitly defines the inputs and outputs of its ANN controller. Specifically, each sensor on the robot serves as an input to the ANN and each motor requires an output neuron. Additionally, a robot's brain may possess a number of internal or hidden units not directly connected to any inputs or outputs. Moreover, the ANNs may include recurrent connections from any hidden or output neuron to any hidden or output neuron (including self feedback connections). Each neuron computes its output by first computing a weighted sum of its inputs and then applying a sigmoid activation function as is standard in ANNs. However, in RoboGen, it is also possible for hidden or output neurons to be oscillator neurons, as these have been shown to drastically speed up the evolution of effective controllers for locomotion [4]. Specifically, these oscillator neurons do not receive any input, but rather output a sinusoid oscillation as a function of time. Each oscillator neuron is

[11]In the original design of RoboGen, each brick also had connection slots on its top and bottom. While this did allow for an even greater diversity of morphologies, it was decided to limit the connections to only be within the $x-y$ plane in order to prevent robots from having to fight gravity while evolving behaviors, while also providing easy access to all electronics. Robots in the system presented here can still stand up and move in three dimensions, but their initial configurations are necessarily planar.

[12]RoboGen also supports a larger set of body parts including wheels and whegs (both with paremeterized diameters) as well as rotational joints, but these are generally disabled for student projects in order to encourage the evolution of more biologically-inspired robots, and will not be discussed further here.

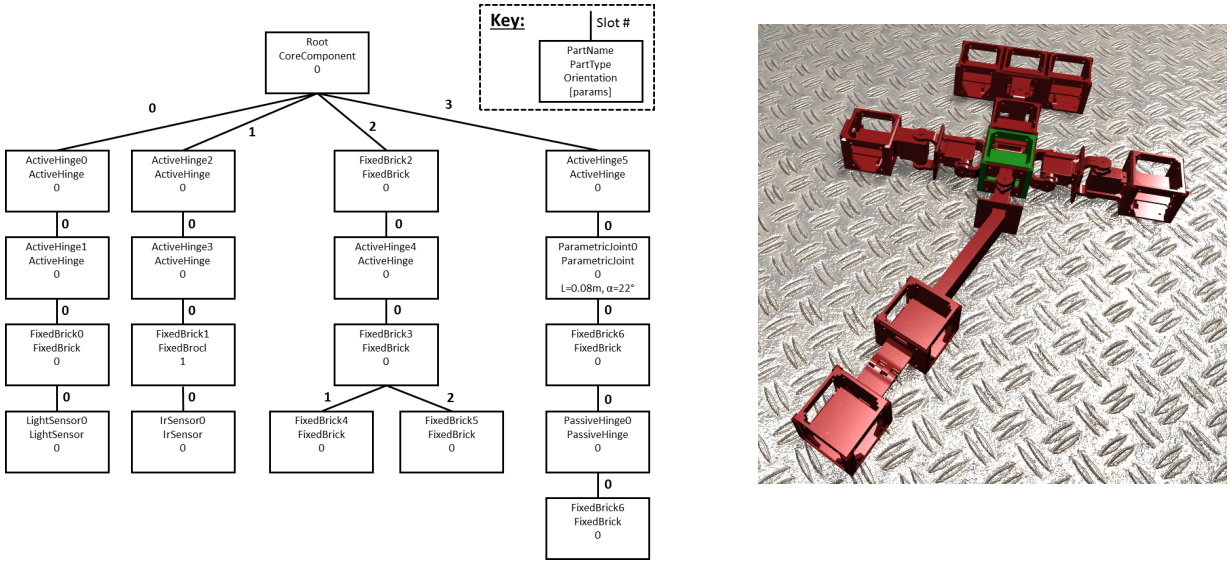

Fig. 4. An example of a robot body tree (left), and the simulated robot that this tree represents (right). Each node represents a body part and maintains (a) pointers to its children (other parts that attach to it) depicted by the labeled edges (labels depict which slot the child is attached to, empty slots omitted for clarity), (b) the type of part, (c) an orientation relative to its parent (given by an integer value representing increments of 90°), and (d) evolvable parameters (if applicable).

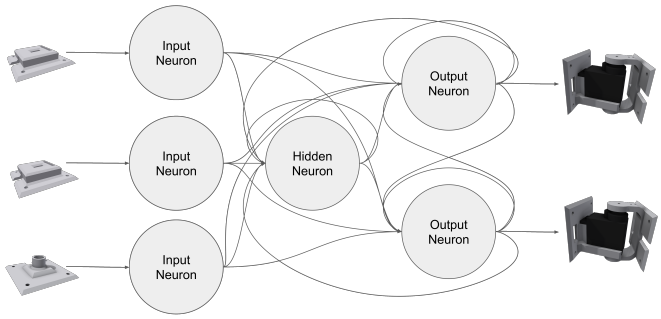

Fig. 5. An example of an artificial neural network 'brain' for a robot with three sensors, one hidden neuron, and two motors. Note: the sensor inputs from the IMU are omitted for clarity.

defined by its period, its amplitude, and its phase-offset from a central clock. An example RoboGen brain is depicted in Fig. 5.

In the RoboGen software, each body part maintains a list of neurons that it contains (hidden neurons are by default part of the *CoreComponent*). Each neuron representation in this list stores the layer (input, output, or hidden) of the neuron, the type (sigmoid or oscillator) of the neuron, and its parameters (bias for sigmoid neurons; period, amplitude, and phase-offset for oscillator neurons).

The connections and weights for the neural network are maintained in a single map: **weights** for the whole robot that maps ordered-pairs of neurons to floating point values. If an entry **weights**$[(i, j)]$ exists then the connection from neuron i to neuron j exists with weight **weights**$[(i, j)]$. If no such entry exists, then there is no connection from i to j in the neural network. For the purpose of applying evolutionary variation operators (see below), this representation is converted into a single vector containing all weights and parameters.

B. Software Suite

The software suite (Fig. 2) is comprised of two main components: an evolution engine and a physics-based simulation engine. In short, the evolution engine runs an evolutionary algorithm by starting with a randomly initialized population of abstract robot representations, sending each representation to the the simulation engine, which first translates each abstract representation into a concrete, physically simulated robot, simulates this robot in a virtual environment, and finally reports back a performance measure or fitness for the given robot. The evolution engine then eliminates the poor performing robot representations and reproduces the well performing ones (with variations). The newly created robot representations are next sent to be evaluated in the simulation engine and the process repeats until a stopping condition is reached (usually a fixed number of iterations).

1) Evolution Engine: The evolution engine contains all of the components needed to run such an evolutionary algorithm on a population of robots: representation, variation, and selection mechanisms. The evolutionary algorithm is capable of operating in two modes: *brain-only* mode, which only evolves the controller or 'brain' of an existing robot morphology, and *full* mode, which evolves both the brain and 'body' (morphology) of robots. In *brain-only* mode the body tree must be manually defined (or previously evolved). When using *full* evolution, it is possible to either "seed" evolution with an existing morphology or create the initial population completely at random. At each generation new body trees are created by mutating clones of well-performing robots using one or more of the mutation operators specified in Table I based on user-configurable probabilities.

The body plan defines the inputs and output of the robot's brain, i.e. as sensor or motor body parts are added or removed from the body tree, input and output neurons (respectively) are

added or removed from the brain. Additionally, hidden neurons may be added or removed based on configurable probabilities. When a neuron is removed, all connections to/from that neuron are also removed. When a neuron is inserted, zero-weight outgoing connections are created to all sigmoid hidden and output neurons. When a hidden or output neuron is inserted, it is chosen to be either a sigmoid or oscillator neuron based on a configurable probability. If it is a sigmoid neuron, zero-weight incoming connections are created from all neurons. In addition to these topological changes, the parameters of a robot's brain (connection weights, and neuron parameters) are mutated by Gaussian perturbations based on user defined probabilities and magnitudes.

Sexual reproduction (AKA crossover) is possible when evolving ANNs with a fixed topology (e.g. doing *brain-only* evolution with 0 probability of adding additional hidden neurons), but otherwise is disabled for simplicity. When two brains are crossed over, the vectors of weights and parameters are concatenated into a single vector, single point crossover is applied, and this crossed-over vector is used to create a new brain representation prior to the application of the parametric mutations.

At each generation the current μ parents compete in tournaments (of configurable size) to decide which parents get to produce offspring. This process repeats until λ children are produced. After the λ children have been evaluated in the simulation engine the next set of μ parents are chosen by choosing the most fit of either the current λ children ("comma" selection) or the most fit out of the λ children and previous μ parents ("plus" selection; terminology borrowed from Evolution Strategies [5]), and this process repeats for the user-specified number of generations.

In addition to the **basic** evolution mode just described, RoboGen also supports evolving brains via a more sophisticated EA, known as HyperNEAT [45][13], which indirectly encodes the weights and parameters of the neural network using an encoding inspired by developmental biology. The details of HyperNEAT are beyond the scope of this paper, but it has been shown to be successful in a number of evolutionary robotics applications [1], [4], [10].

2) Simulation Engine: Every newly created robot representation (brain and body) is sent to the physics-based simulation engine[14] in order to evaluate its fitness (refer to Fig. 2). The abstract representation is translated into a physical model of the robot along with code for operating its brain, and this robot is placed inside of a user-defined, simulated environment and allowed to act. The robot's fitness is computed either using one of the built-in fitness functions, or the user may write a custom fitness function in JavaScript.

By default the environment is an infinite flat plane with Earth normal gravity. However, this can be modified in a number of ways, the most important of which is through the inclusion of obstacles. By defining a set of obstacles and light sources (either stationary or movable) that the robot can interact with, it is possible to create more complex environments such as rough terrains, mazes, stairs, goal locations etc.

Mutation Operator	Description
NodeInsert	Insert a random node at a random location in the body representation tree.
NodeRemove	Remove a random node from the body tree representation.
SubtreeDuplicate	Duplicate a randomly chosen subtree and insert it at a random location on the body tree.
SubtreeSwap	Swap two randomly chosen subtrees of the body tree representation.
SubtreeRemove	Remove a randomly chosen subtree from the body tree representation. Unlike *NodeRemove* which attempts to remove a node and propagate its children upwards, *SubtreeRemove* removes a node and all of its descendants.
MutateParam	Mutate a randomly chosen parameter of a randomly chosen node. For the purpose of this operator a node's orientation relative to its parent is also consider to be a parameter.

TABLE I
THE AVAILABLE BODY-TREE MUTATION OPERATORS. THE PROBABILITY OF APPLYING EACH OPERATOR IS USER-CONFIGURABLE.

3) Interface and Distributed Computation: The RoboGen platform is written in C++. This C++ code has been "transpiled" to JavaScript using Emscripten[15]. Consequently, it is possible to run RoboGen entirely in a web browser, which is inherently cross-platform and does not require installing any additional software. Running in the web browser also means that it is possible to leverage modern web design tools in order to build an interface that is more accessible to novice users than the command line. The current web-based interface replaces command line executions with a web form for launching experiments, and a tabular status display for monitoring results and quickly viewing the best robot at every generation.

RoboGen software can either run on a single personal computer or make use of the Seamless Peer and Cloud Evolution (SPACE) Framework [27] for seamlessly distributing fitness evaluations to simulation engines running both in the web browsers of other users as well as optimized versions running on cloud compute nodes. With this framework, users are able to choose to run experiments on their own computers, share compute resources publicly, or create a private pool of computers to share among a particular group.

Each step in the process of using RoboGen is fully documented on http://www.robogen.org. This documentation covers both a desktop (C++) version and a web application. The full setup and use of the software is described along with examples and in depth descriptions of all parameters. Moreover, complete details of building the physical robots are provided: several options for obtaining the parts (printing them oneself and purchasing from a 3rd party service[16]) as well as details of how to assemble and connect the sensors and actuators, and finally how to upload the evolved "brain" to the microcontroller.

[13]Specifically, an adaptation of Peter Chervenski's MultiNEAT is used: http://www.multineat.com/

[14]The simulation engine is built on top of the Open Dynamics Engine: http://www.ode.org/

[15]https://github.com/kripken/emscripten

[16]http://www.shapeways.com

In addition, to make the fabrication process easier to follow, 7 short tutorial videos[17] are embedded in the RoboGen documentation pages. These feature step by step instructions for the most important steps of building a robot. They are shot in a first person view and include zoomed in shots for showing intricate details.

C. Using RoboGen for a master's level course

RoboGen has been used for project based learning within a master's level course (Micro-515, Artificial Evolution) at the Ecole Polytechnique Fédérale de Lausanne (EPFL). Students are first exposed to RoboGen through a series of exercise sessions, and then they are given free reign to evolve robot bodies and brains for a chosen scenario, which they will ultimately fabricate.

The exercise sessions aim to familiarize the students with the relevant concepts of Evolutionary Robotics (e.g. fitness functions and the parameters of the evolutionary algorithm) that they have already seen in lectures. At the same time they are learning to use the RoboGen platform. Before beginning these sessions, it is explained to the students that they will be using the RoboGen platform for their course projects, and so, as they work through the exercises, they can begin to think about what kinds of scenarios they would like to work on for their projects, and how they might implement them. Each of the exercises is structured as step-by-step instructions followed by checkpoint questions, which challenge the students' understanding of what they are doing. The answers to these questions are later made available so that the students can verify their responses. The details of these exercises may be found on http://robogen.org

After gaining a base understanding of RoboGen through this hands on experience, the students form project groups and begin working on their projects: writing fitness functions and performing experiments. Following the philosophy of IBL, the students are given lots of leeway in conducting experiments as they see fit, however they are encouraged to use proper research techniques and to conduct multiple repetitions of experiments in order to perform statistical analyses.

Many issues tend to arise when the students move from the guided exercises to their own, often lofty, project ambitions. For instance, the students quickly realize that, on non-trivial tasks, small population sizes and/or too strong of selection pressure will quickly lead to premature convergence to local optima. Another common lesson is that evolution frequently finds solutions that score well on a given fitness function, while not actually achieving the experimenter's intent. These are valuable lessons for students to learn, and following the IBL philosophy the students are left to first learn them from their own experience, and only later are they elaborated upon by the instructors.

In parallel to their experiments, the students are encouraged to begin building their robots, along with their test environments or arenas, as soon as possible. This allows them to see the real robot behavior and potential discrepancies that might arise, in order to address them in ongoing evolutionary experiments. Because the robot can be fully disassembled and reassembled, they can test intermediate solutions or different evolved morphologies in order to refine their results. This hardware work is an important part of their research process.

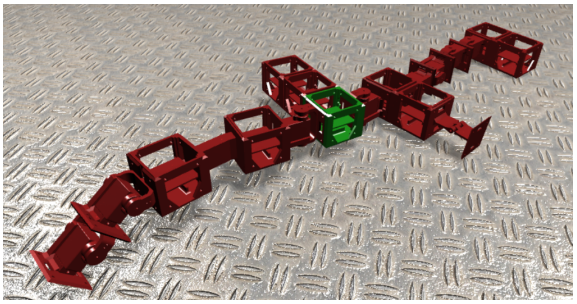

Fig. 6. Team Stabilize's robot evolved for delivering drinks.

The students are given an initial set of electronic parts including a microcontroller, two batteries, a power regulator, six servomotors, eight *FixedBricks*, and six *ActiveHinge* frames to get them started. They are then responsible for 3D printing any additional parts they need to build their robots[18], and can obtain sensors on request. While each group is responsible for their own materials and building their own robot, they are encouraged to share their skills and extra materials with others in a spirit of collaboration.

At the conclusion of the project the students have to prepare two deliverables for their final evaluation. The first is a written report, in the style of an IEEE conference publication, describing their project in detail. The second is a final presentation delivered in front of their peers and the instructors. For these presentations the students are allotted a hard time limit of eight minutes followed by three minutes of Q&A. The final report and presentation should demonstrate that the students understand the main principles of Evolutionary Robotics, including evolution of robot brains and bodies, and that they are able to apply these principles to interesting scenarios. The students are informed that their evaluations will be based on: (a) the quality and clarity of their report and presentation, (b) the strength of their results, (c) the use of systematic and scientific methodologies, (d) the complexity and feasibility of the scenario(s) investigated, and (e) their ability to work through difficulties.

IV. Results and Discussion

Here, the results of using RoboGen during the Spring semester of 2016 are briefly presented. First, we use a few example projects to showcase what the students were able to accomplish, discuss common lessons that they learned, and describe some of the self-directed learning experiences they underwent. After this, we present a sampling of the feedback that the students provided in response to a post-course questionnaire.

A. Student Project Examples

Due to the open-endedness of the projects, and the freedom that students were allotted to design their own scenarios, the

[17] https://www.youtube.com/channel/UCaCe181uC3GIZxWrxOECxIA

[18] A dedicated 3D printer is made available to them for this purpose.

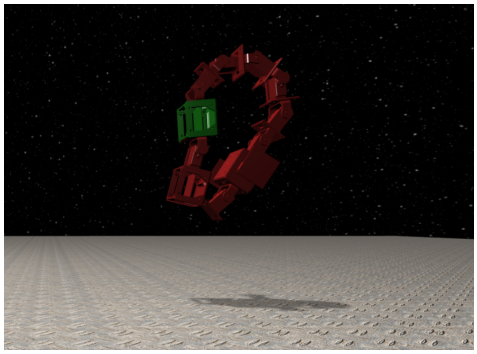

Fig. 7. A robot evolved by Team Jump, mid jump.

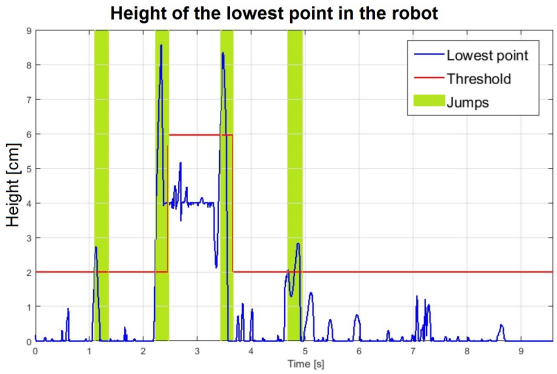

Fig. 8. The function that the students in Team Jump devised for detecting when a robot has jumped and landed on an obstacle. The red line depicts the current height threshold to be considered airborne, the blue line represents the current lowest point of the robot's axis aligned bounding box (AABB), and the jumping periods are highlighted in green.

students' projects produced a diversity of scenarios and robotic forms. We have chosen three example student projects to highlight this diversity, to shine light on the difficult task of writing fitness functions, and to highlight how the groups were able to transfer their simulated results to real robots. We will refer to these three groups as follows: **Team Stabilize**, **Team Jump**, and **Team Escape**.

Fitness Function Design

Team Stabilize: this group originally wanted to evolve a robot for delivering drinks. Since simulating the fluid dynamics of actually carrying a drink would be beyond the scope of RoboGen's simulation engine, their idea was that the robot should be able to move as fast as possible while maintaining stability. Since these two objectives are antagonistic, the choice of fitness function defines a given trade-off. Additionally, the way in which the fitness function is designed might make it nearly-impossible to move off of a local optimum in the fitness landscape. As a result, the students incrementally tuned their fitness function such that it gave enough importance to both objectives at the same time[19] in order to arrive at a good solution (depicted in Fig. 6)[20].

Team Jump: this group chose to evolve a robot capable of jumping onto an object. Once again devising an appropriate fitness function formed a sizable challenge. After careful experimentation they ended up using an incremental evolution [18] approach to subdivide this goal into three sub-goals to be fulfilled in succession: (1) jump as high as possible in place, (2) jump with lateral displacement, and (3) identify a target object and jump onto it.

Rewarding vertical jumps could be done with a straightforward function on the minimum z (vertical) coordinate of the robot's axis aligned bounding box. This resulted in a snake-like morphology (Fig. 7) whose motion consisted of a folding phase where it curved on itself, followed by a forceful, coordinated movement of all servos to propel itself into the air and curve the opposite way, resulting in a propulsive motion similar to that of a jumping worm or caterpillar.

[19]A more general solution would be to use a multi-objective Evolutionary Algorithm [11], and the students do learn about such algorithms in the course. However, this functionality is not currently part of RoboGen, and implementing it is beyond the scope of these projects.

[20]See also https://youtu.be/8g7POPzZ6n8 for a video of the real robot in action

Rewarding lateral displacement was slightly more difficult, as the students needed a way to only reward lateral movements that occurred during a jump, and not other lateral movements that could be achieved without jumping. Eventually this group settled on using a threshold of the robot's minimum z-coordinate to consider the robot to be "airborne". The fitness function then would compute the horizontal displacement achieved while the robot was in the air.

The final objective: jumping onto an obstacle, proved the trickiest to select for. This could not simply rely on a robot's lowest point crossing below a threshold, since if it jumped onto an object, its lowest point would never pass below the threshold again, and hence the jump would never be considered complete. Instead the students came up with the idea of considering the jump to be completed when the robot's lowest point did not change within a certain time window. When this happened, it could be inferred that the robot had "landed". Specifically, the group came up with the following criteria: a jump was considered complete when a robot's lowest point did not change by more than 2mm within $80ms$ of simulated time. At this point the height threshold for future jumps was updated to be 2cm above the robot's current resting height (h). With this new method of computing jump distances, the students could create a function that determined the status of the robot at all times (airborne, on a step of height h, or on the ground). This can be seen in Fig. 8. These modifications allowed them to use the same fitness metric as in previous steps: the maximum performance of all jumps performed during a trial, and therefore select for their ultimate goal: a robot capable of jumping a large distance onto an object.

Following this procedure, the students were able to evolve a robot that could jump an impressive 19 cm vertically, or about 540% of the height of the robot. In addition, it was capable of jumping with a lateral motion, onto a platform, balancing itself, and jumping back down.

Team Escape: this group's goal was to evolve a robot capable of escaping from a restricted environment. The robot initially was able to move in an open environment and then the environment was gradually restricted over evolutionary

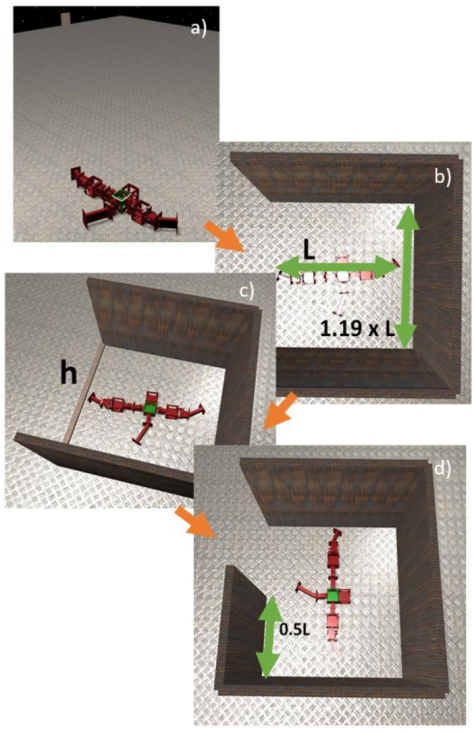

Fig. 9. The progression of environments devised by Team Escape in order to evolve "escapist" robots. See text for details

time. By modifying the task environment in this fashion they were able to gradually increase the complexity of the robot's behavior without modifying the way in which fitness was computed.

The robot was forced to evolve to free itself from its "cage" and approach an attractor. In order to encourage generalist solutions, the scenario involved evaluating each robot from several different starting orientations inside of the "cage". The progression of environments can be seen in Fig. 9. The first restriction was to build three walls surrounding the robot (b). The distance between the walls was chosen as 119% of the robot body's maximum dimension, to be as restrictive as possible, while allowing the robot to move. The second restriction was a 1cm high bar against the ground across the opening wall (c). This edge added a constraint that the robot had to lift its "feet" over this bar to make it outside. Finally, as an additional challenge, the students decided to see if the robot could make it out of its cage if the opening was smaller that its body. They closed the opening halfway leaving only a 23cm opening (d). The robot evolved to rotate its body until one "leg" was outside the cage, then folded itself around the wall and inched the second leg out.

In all three of these cases the students pursued complex behaviors and were forced to incrementally modify the way in which fitness was calculated and/or modify the task environments in which the robot was evaluated in. The students thus learned on their own to utilize techniques commonly employed in the Evolutionary Robotics literature such as incremental evolution [18], scaffolding [6], and shaping [2]

From Simulation to Reality

Team Escape carefully reproduced their simulated arena in reality. Then, from observing the movement of their robot in this real world arena, the students noticed that the friction between the robot's "feet" and the floor and walls had a large influence on its behavior. They thus proceeded to study the effect this would have on the performance of the robot. Running several additional evolutionary experiments with higher friction, they found that the fitness of their escapist robot was improved for a higher friction coefficient. They then validated this on the physical robot, by adding some tape to the robot's feet[21].

While **Team Stabilize** did not require an arena beyond a flat surface, they too needed to experiment with different ground surfaces to match the friction seen in the simulator. Then, on personal initiative, they went beyond the project requirements and performed an additional study: they installed a data logger on their physical robot to quantitatively evaluate its stability, which was then used to compare the simulated and real fitnesses, as well as to consider sources of discrepancy. In fact, the real robot gave varying fitnesses over several trials, all lower than the simulated fitness by an average of 25%. They cited three possible reasons for this: differences in the friction between the robot and the ground, noise in the servomotors' movement, and slight discrepancies in the geometry—either in the joints' range of movements or imperfections in the assembly of the robot.

Finally, due to the forceful ground impacts created by **Team Jump's** robot, several servo gearboxes were broken. To prevent this from happening, they started testing their real robot on a thin sheet of styrofoam. The styrofoam acted as a cushion to protect the servos, and even though it was thin, it still absorbed some of the energy, thus decreasing the height of the jump: in the real world, the robot could only jump up to 5 cm[22]. While not as impressive as their simulation results, this was still a non-trivial accomplishment.

Each of these groups highlights the success of the RoboGen projects. **Team Stabilize's** work highlights not only the successful evolution of a robot for a non-trivial task, but also the self-driven question and answering that comes out of these projects, and how students may leverage skills and techniques not directly taught in the course to carry their projects further. **Team Jump's** work also highlights the interesting ways that students went beyond the scope of the project. It showcases the students' willingness to build interesting physical arenas, and iteratively study a property of the system (in this case friction) that is made possible by the many configuration options available in RoboGen.

In summary, all three groups' careful engineering of appropriate fitness functions for complex behaviors, construction of test environments, and thoughtful consideration of discrepancies between simulation and reality were all important

[21]A video demonstrating this robot is available online at https://youtu.be/efz-mvpJDqI

[22]A video of the real robot is available online at https://youtu.be/Q0oBu_yQtJo

learning experiences that uniquely resulted from the open-ended, hands-on nature of evolving robots with RoboGen. While many other students groups also tackled a diversity of interesting scenarios, these three are the only ones we will discuss here due to space constraints. They highlight the diversity of tasks achieved, the interesting ways in which students learned from discrepancies between simulation and reality, and how the openness of the projects allowed the students to leverage their own knowledge and expertise to go beyond what was required of them.

B. Course Evaluation Results

At the end of the project, a questionnaire was given to the students in order to obtain as much feedback as possible about the course and the project. The complete results can be seen at http://robogen.org/research/questionnaire_results.html, and the most important points will be discussed here.

The questionnaire was complete by 27 out of 42 students, which is about 64% of the class. The students who responded came from six different majors: Microengineering (63%), Computer Science (14.8%), Electrical Engineering (7.4%), Bioengineering (7.4%), and one student apiece from Life Sciences and Mechanical Engineering. Most of these students (77.8%) stated that they were in the 2nd semester of their master's degree.

The first questions of the questionnaire required the students to evaluate their knowledge on each of the learning outcomes of the course that were covered in the project[23]. Then, the logistics of the project were examined[24]. These questions were formulated as statements and the student responded on a Likert scale from 1 (disagree) to 5 (agree) [29]. Finally, the students were presented with several questions where they were able to offer their own free-form responses. The results offer helpful insights into the students' understanding and perception of the RoboGen platform, its documentation, and the structure of the project.

The questions relating to the learning outcomes suggested that the students gained many important skills, such as how to simulate robots, how to evolve brains and bodies, how to fabricate robots, and how to control a robot with an Arduino microprocessor (see Table II). This lends support to the effectiveness of the projects as valuable learning experiences. However, the students' understanding of neural networks lagged behind the other skills (as evidenced by the last two rows in Table II), and is something that needs to be addressed in future iterations of the course.

Overall, the students evaluated the project positively, with many questions relating to the overall project content and organization receiving mean responses above 4 (refer to http://robogen.org/research/questionnaire_results.html). Additionally, many of the students enjoyed and felt they benefited from the project as evidenced by their free-form comments:

- *Simply the best [project] I've done at the EPFL !*
- *Finally a project during the Robotic cursus [sic] where we have to build a robot!*
- *It simulates a whole research process: from parameters selection, to simulation and finally to the real implementation.*
- *Think, design, built and run.. From zero to hero!*
- *It was quite great, I enjoyed a lot while learning the subjects and relating my knowledge with real life usage.*

[23]Questions under *Learning outcomes*
[24]Questions under *RoboGen Project Evaluation*

Statement	Mean Response +/- σ
I know how to use the RoboGen software to simulate a robot.	4.78 +/- 0.42
I know how to use the RoboGen software to simultaneously evolve the brain and body of a robot	4.70 +/- 0.46
I can design my own robot morphology in RoboGen by defining its tree structure.	4.78 +/- 0.42
I understand the benefits and drawbacks of including noise in my simulations.	4.74 +/- 0.44
I understand the benefits and drawbacks of performing multiple simulations (trials) per fitness evaluation.	4.92 +/- 0.26
I can build a robot that I evolve.	4.74 +/- 0.58
I can run my evolved neural controller on the Arduino microcontroller of the robot I have built.	4.59 +/- 0.68
I understand how the artificial neural network controllers (brains) of RoboGen robots are represented.	3.44 +/- 1.17
I understand how the brains of RoboGen robots operate.	3.56 +/- 1.26

TABLE II
A SAMPLING OF QUESTIONS AND AVERAGE RESPONSES (LIKERT SCALE, OUT OF 5) RELATING TO THE STUDENT'S POST-PROJECT UNDERSTANDING OF ROBOGEN AND EVOLUTIONARY ROBOTICS.

V. CONCLUSIONS AND FUTURE WORKS

In the spirit of Inquiry-Based learning, the goal of the RoboGen project is for students to learn the course material through a research-based method. Once they have been provided the basic level of competency to proceed, the students design their own methodology for better understanding the mechanisms of evolutionary algorithms, and then must report in detail on their experimental method and results. The results presented above support the idea that RoboGen is an effective platform for this purpose both through sample student experiences and through student responses to a post-project questionnaire.

The post-project questionnaire is also of crucial importance for improving RoboGen in the future. For instance, the relatively poor understanding of neural networks could be addressed by extending RoboGen with a mechanism to visualize the neural networks and the way in which they are evolved. There are also several additional places that the platform and project structure could be improved for future students. In fact, the final question presented to the students on the questionnaire was: "What do you think could be improved in the project?" These responses are crucially important for planning the future work on RoboGen.

One frequent comment was about the RoboGen web interface. Since this crop of students served as beta-testers for the new interface, this was to be expected, and their extensive feedback is being taken into account in ongoing improvements.

Another important point was raised by one of the students: "It would have been good to have much more examples and results [sic] of evolved robots. It was difficult to define a project without knowing the limitations of Robogen in advance." This is a point well taken, as it might be difficult for students to fully grasp what projects are feasible with the platform. While the students discussed their scenarios with the teaching assistants prior to beginning their experiments, this relies on the TAs' prior experience and runs counter to the open-ended, self-guided inquiry that is a driving principle behind these projects. Now that the platform has reached a level of maturity and several groups of students have completed it, it will be possible in the future to present new students with a multitude of previous results. However, in doing so, one must be careful so as not to bias the students too much into only investigating scenarios that are close to those used previously. Rather, seeing old results should give the students a good starting point for understanding what RoboGen can be used for, and hopefully they will be able to use this inspiration to create a diversity of novel scenarios on their own.

In order to address requests for greater functionality, additional future improvements will aim at incorporating more sophisticated techniques from the Evolutionary Robotics literature to help evolve robots for more complex tasks. Moreover, further work will go into designing new body parts to expand the diversity of morphologies. In particular, it will be interesting to incorporate additional sensors as well as body parts offering more compliance—including the use of soft and flexible materials. However, there is currently a limit to the types of softness that can be efficiently simulated using current technologies.

Finally, it is worth noting that while RoboGen has been employed in these projects for evolving robot bodies and brains, the platform may have broader appeal among educators seeking to teach robotics and/or AI, but not necessarily Evolutionary Robotics. By designing robot bodies manually and programming their control strategies, it is possible to rapidly test ideas inside of RoboGen's web-based simulation engine, and then build these robots cheaply and easily. Fabricating these robots would still offer valuable learning opportunities in 3D printing, electro-mechanical assembly, and embedded processing. Furthermore, due to RoboGen being fully open-source, it could be extended to develop interesting student projects around such topics as designing additional components (requiring knowledge of both mechanical design and physical simulation), and searching for controllers through alternative methods, such as Reinforcement Learning [47].

In this paper, we have presented RoboGen: an open-source, web-based, software and hardware platform for Bio-Inspired Artificial Intelligence and Evolutionary Robotics. We first surveyed the state of the art in educational robotics platforms to show that RoboGen is unique in offering open source software and hardware that allows students to design and evolve robot bodies and brains in simulation, and then fabricate these bio-inspired, articulated robots at low cost using 3D printers and off-the-shelf electronic components. We then described the platform in detail, and presented its use as a tool for project based learning within the confines of a master's level course at EPFL. The diversity of students' results, their enthusiasm for the project, and their mastery of the majority of the learning objectives all lend credence to RoboGen being an effective educational tool.

Now that RoboGen exists as a web-application, which can be used by anyone in the world with a modern web browser, it will be relatively easy for other educators to begin using RoboGen in their own classes and for individual interested students to use the software on their own. In fact, RoboGen is currently being adopted by other universities worldwide including Heriot Watt University in the United Kingdom and VU Amsterdam in the Netherlands for use with their own students. Ongoing improvements to the web interface and other enhancements will hopefully speed up this adoption, and having extensive student feedback offers an invaluable resource for this undertaking.

ACKNOWLEDGMENTS

We would like to thank past and present members of EPFL's Laboratory of Intelligent Systems, in particular Grégoire Heitz and Guillaume Leclerc, for their contributions to the RoboGen platform. We also thank the students of EPFL Micro-515 for all of their hard work and useful feedback.

This project was supported by the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 308943 (FET-OPEN Insight project).

REFERENCES

[1] J. E. Auerbach and J. C. Bongard. Evolving Complete Robots with CPPN-NEAT: The Utility of Recurrent Connections. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2011.

[2] J. E. Auerbach and J. C. Bongard. Evolving monolithic robot controllers through incremental shaping. In *New Horizons in Evolutionary Robotics*, pages 55–65. Springer, 2011.

[3] J. E. Auerbach and J. C. Bongard. Environmental influence on the evolution of morphological complexity in machines. *PLoS computational biology*, 10(1):e1003399, 2014.

[4] J. E. Auerbach, G. Heitz, P. M. Kornatowski, and D. Floreano. Rapid evolution of robot gaits. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, pages 743–744. ACM, 2015.

[5] H.-G. Beyer and H.-P. Schwefel. Evolution strategiesa comprehensive introduction. *Natural Computing: an international journal*, 1(1):3–52, 2002.

[6] J. Bongard. Behavior chaining: incremental behavioral integration for evolutionary robotics. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau, editors, *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 64–71. MIT Press, Cambridge, MA, 2008.

[7] D. J. Cappelleri and N. Vitoroulis. The robotic decathlon: Project-based learning labs and curriculum design for an introductory robotics course. *Education, IEEE Transactions on*, 56(1):73–81, 2013.

[8] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 167–174. ACM, 2013.

[9] C. M. Cianci, X. Raemy, J. Pugh, and A. Martinoli. Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics. In *International Workshop on Swarm Robotics*, pages 103–115. Springer, 2006.

[10] J. Clune, B. Beckmann, C. Ofria, and R. Pennock. Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing*, pages 2764–2771, 2009.

[11] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 1st edition, 2001.

[12] P. Eggenberger. Evolving morphologies of simulated 3D organisms based on differential gene expression. *Procs. of the Fourth European Conf. on Artificial Life*, pages 205–213, 1997.
[13] D. Floreano and C. Mattiussi. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. Intelligent Robotics and Autonomous Agents Series. Mit Press, 2008.
[14] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural network driven robot. In *Proceedings of the third international conference on Simulation of adaptive behavior: From Animals to Animats 3*, number LIS-CONF-1994-003, pages 421–430. MIT Press, 1994.
[15] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3):143–166, 2003.
[16] M. P. Freedman. Relationship among laboratory instruction, attitude toward science, and achievement in science knowledge. *Journal of Research in Science Teaching*, 34(4):343–357, 1997.
[17] P. Funes and J. Pollack. Computer evolution of buildable objects. *Fourth European Conference on Artificial Life*, pages 358–367, 1997.
[18] F. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(3-4):317–342, 1997.
[19] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, 20:205–224, 1997.
[20] M. Healey. Linking research and teaching exploring disciplinary spaces and the role of inquiry-based learning. *Reshaping the university: new relationships between research, scholarship and teaching*, pages 67–78, 2005.
[21] J. Heinerman, D. Drupsteen, and A. Eiben. Three-fold adaptivity in groups of robots: The effect of social learning. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 177–183. ACM, 2015.
[22] A. Hofstein and V. N. Lunetta. The laboratory in science education: Foundations for the twenty-first century. *Science Education*, 88(1):28–54, 2004.
[23] P. W. Hughes and M. R. Ellefson. Inquiry-based training improves teaching effectiveness of biology teaching assistants. *PloS one*, 8(10):e78540, 2013.
[24] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in artificial life*, pages 704–720. Springer, 1995.
[25] A. Kokosy, M. V. Micea, and P. Saey. A professional project based learning method in mobile robotics. In *Frontiers in Education Conference (FIE), 2014 IEEE*, pages 1–7. IEEE, 2014.
[26] S. Kradolfer, S. Dubois, F. Riedo, F. Mondada, and F. Fassa. A sociological contribution to understanding the use of robots in schools: the thymio robot. In *Social Robotics*, pages 217–228. Springer, 2014.
[27] G. Leclerc, J. E. Auerbach, G. Iacca, and D. Floreano. The seamless peer and cloud evolution framework. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 821–828. ACM, 2016.
[28] D. Lessin, D. Fussell, and R. Miikkulainen. Open-ended behavioral complexity for evolved virtual creatures. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 335–342. ACM, 2013.
[29] R. Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.
[30] H. Lipson and J. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978, 2000.
[31] H. H. Lund. Co-evolving control and morphology with lego robots. In F. Hara and R. Pfeifer, editors, *Morpho-functional machines: the new species; designing embodied intelligence*, pages 59–79. Springer, Tokyo, Japan, 2003.
[32] M. J. Mataric. Robotics education for all ages. In *Proceedings, AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education*, 2004.
[33] O. Miglino, H. H. Lund, and S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial life*, 2(4):417–434, 1995.
[34] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.
[35] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.
[36] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Experimental robotics III*, pages 501–513. Springer, 1994.
[37] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology,Intelligence,and Technology*. MIT Press, Cambridge, MA, USA, 2000.
[38] G. B. Parker, A. S. Anev, and D. Duzevik. Evolving towers in a 3-dimensional simulated environment. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, pages 1137–1144. IEEE Press, 2003.
[39] C. Paul. Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems*, 54(8):619–630, 2006.
[40] A. Price, R. Rimington, M.-T. Chew, and S. Demidenko. Project-based learning in robotics and electronics in undergraduate engineering program setting. In *Electronic Design, Test and Application, 2010. DELTA'10. Fifth IEEE International Symposium on*, pages 188–193. IEEE, 2010.
[41] F. Riedo. Thymio: a holistic approach to designing educational robots. *PhD Thesis EDPR*, 2015.
[42] F. Riedo, P. Rétornaz, L. Bergeron, N. Nyffeler, and F. Mondada. A two years informal learning experience using the thymio robot. In *Advances in Autonomous Mini Robots*, pages 37–48. Springer, 2012.
[43] M. Ruzzenente, M. Koo, K. Nielsen, L. Grespan, and P. Fiorini. A review of robotics kits for tertiary education. In *Proceedings of International Workshop Teaching Robotics Teaching with Robotics: Integrating Robotics in School Curriculum*, pages 153–162. Citeseer, 2012.
[44] K. Sims. Evolving virtual creatures. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, NY, USA, 1994. ACM.
[45] K. Stanley, D. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.
[46] S. Sumathi, T. Hamsapriya, and P. Surekha. *Evolutionary intelligence: an introduction to theory and applications with Matlab*. Springer Science & Business Media, 2008.
[47] R. Sutton and A. Barto. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.
[48] A. Takacs, G. Eigner, L. Kovacs, I. J. Rudas, and T. Haidegger. Teacher's kit: Development, usability, and communities of modular robotic kits for classroom education. *IEEE Robotics Automation Magazine*, 23(2):30–39, June 2016.
[49] L. K. Wells and J. Travis. *LabVIEW for everyone: graphical programming made even easier*. Prentice-Hall, Inc., 1996.
[50] Z. Zhang, C. T. Hansen, and M. A. Andersen. Teaching power electronics with a design-oriented, project-based learning method at the technical university of denmark. 2015.

Joshua E. Auerbach received a B.Sc. from McGill University in Computer Science followed by a GCert in Complex Systems and a Ph.D. in Computer Science from the University of Vermont. Joshua is an Assistant Professor in the Dept. of Computer Science and Innovation at Champlain College in Burlington, Vermont, USA. His research is centered around building artificial systems inspired by neuroscience, evolution, and other biological processes.

Alice Concordel received an M.Sc. (2014) in Mechanical Engineering and Robotics from Ecole Polytéchnique Fédérale de Lausanne, Switzerland. Her research interests include bio-inspired robotics and artificial intelligence.

Przemyslaw M. Kornatowski received an M.Sc. (2011) in Mechatroniccs from Warsaw University of Technology, Poland. In 2012 he started work in the Laboratory of Intelligent Systems at EPFL as a mechanical engineer. He has been involved in the design and manufacturing of different types of ground and flying robots. Since 2015, he has been working on a PhD in the field of foldable flying robots for parcel transportation.

Dario Floreano is Director of the Laboratory of Intelligent Systems at EPFL Switzerland and Director of the Swiss National Center of Robotics. His research focuses on the convergence of biology, artificial intelligence, and robotics. He has published more than 300 peer-reviewed papers and four books on the topics of evolutionary robotics, bio-inspired artificial intelligence, and bio-mimetic flying robots with MIT Press and Springer Verlag.