Adapt2Learn: A Toolkit for Configuring the Learning Algorithm for Adaptive Physical Tools for Motor-Skill Learning

Dishita Turakhia
MIT CSAIL
USA
dishita@mit.edu

Andrew Wong
MIT CSAIL
USA
andrewwo@mit.edu

Yini Qi
MIT CSAIL
USA
qyn@mit.edu

Lotta-Gili Blumberg
MIT CSAIL
USA
blumberg@mit.edu

Yoonji Kim
MIT CSAIL
USA
yoonji@mit.edu

Stefanie Mueller
MIT CSAIL
USA
stefanie.mueller@mit.edu

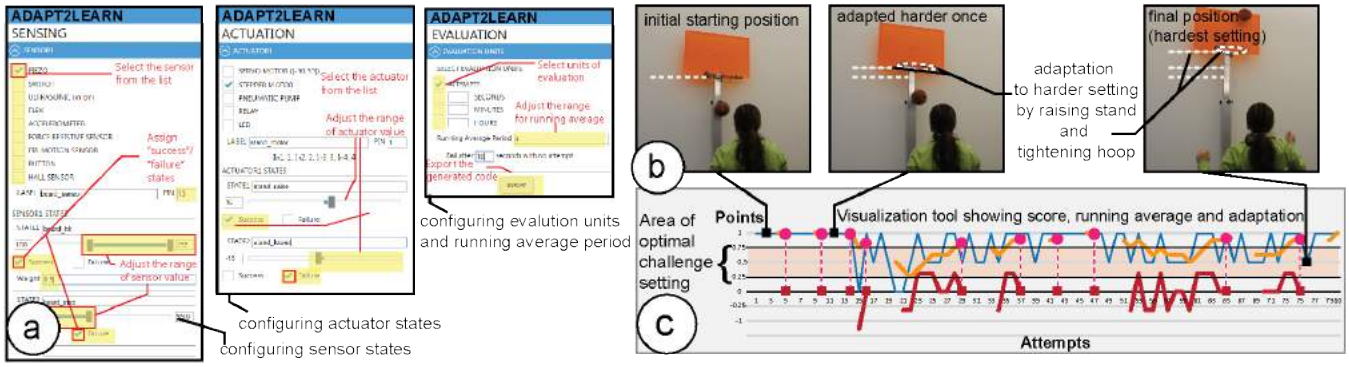

Figure 1: (a) Designers use *Adapt2Learn's* user interface to configure the adaptation of their adaptive training tools, such as (b) an adaptive basketball stand that adapts its hoop height and width. *Adapt2Learn* auto-generates the learning algorithm as a micro-controller script that can be deployed to the tool. The algorithm uses sensor values to assess a learner's performance, computes the optimal training difficulty, and then varies the training difficulty by adapting the hoop height and width. (c) *Adapt2Learn's* built-in visualization tool lets designers visualize the tool's adaptation and evaluate the learning algorithm.

ABSTRACT

A recent study on motor-skill training showed that adaptive training tools that use shape-change to adapt the training difficulty based on learners' performance can lead to higher learning gains. However, to date, no support tools exist to help designers create adaptive learning tools. Our formative study shows that developing the adaptive learning algorithm poses a particular challenge. To address this, we built *Adapt2Learn*, a toolkit that auto-generates the learning algorithm for adaptive tools. Designers choose their tool's sensors and actuators, *Adapt2Learn* then configures the learning algorithm and generates a microcontroller script that designers can deploy on the tool. Once uploaded, the script assesses the learner's performance via the sensors, computes the training difficulty, and actuates the tool to adapt the difficulty. *Adapt2Learn*'s visualization tool then lets designers visualize their tool's adaptation and evaluate the learning algorithm. To validate that *Adapt2Learn* can generate adaptation algorithms for different tools, we built several application examples that demonstrate successful deployment.

CCS CONCEPTS

• **Human-centered computing → User interface toolkits**.

KEYWORDS

adaptive learning, physical interfaces, motor-skill learning, toolkit design

DIS '21, June 28-July 2, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8476-6/21/06.
https://doi.org/10.1145/3461778.3462128

ACM Reference Format:
Dishita Turakhia, Andrew Wong, Yini Qi, Lotta-Gili Blumberg, Yoonji Kim, and Stefanie Mueller. 2021. Adapt2Learn: A Toolkit for Configuring the Learning Algorithm for Adaptive Physical Tools for Motor-Skill Learning. In *Designing Interactive Systems Conference 2021 (DIS '21), June 28-July 2, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3461778.3462128

1 INTRODUCTION

A recent study on motor skill training showed that training with physical tools that adapt their shape based on a learner's performance to maintain optimal task difficulty can lead to higher learning gains [21]. For example, training with an adaptive basketball hoop that automatically adjusts the hoop size and height based on a learner's score leads to higher learning gains than training with a conventional static basketball hoop [21]. Such adaptive training tools make personalized training accessible to a broader audience and expand the design space of training tools for motor skills. However, research on supporting designers in creating such adaptive training tools that adjust training difficulty is limited.

To study the challenges in creating adaptive training tools, we conducted an 8-week formative study with 32 participants (16 teams) in a bi-weekly studio format. The participants built adaptive training tools for various motor skills, such as skating and swimming, in teams of two. Each tool used actuation to vary the training difficulty to support learners' progress from a beginner level to an expert level. For example, to aid beginner skaters, one team built an adaptive skateboard that automatically decreases its length as the learner's balancing skill improves. Another team built an adaptive floaty that automatically deflates as the learner's swimming ability increases.

We observed the participants' design and prototyping process during this formative study and identified the challenges they faced. Our observations and participants' feedback indicated that they struggled the most with developing a learning algorithm for their adaptive training tools. In particular, the development of an algorithm that varies task difficulty through actuation to ensure training at the optimal difficulty level was challenging.

According to the literature in motor skill learning, an effective learning algorithm maintains the learner's training difficulty level at the optimal challenge point [10]. Optimal challenge point in training is when the difficulty level is neither too difficult nor too easy for the learner's skill levels. A recent study demonstrates how to develop such an algorithm for adaptive training tools [21]. However, no system exists to support designers in configuring it for their specific adaptive training tools.

To address this challenge, we built *Adapt2Learn*, a toolkit that auto-generates the adaptive learning algorithm for designers' adaptive training tools for motor skill learning. Adapt2Learn has a user interface that allows designers to configure their tool's adaptation by setting the sensors' and actuators' values. *Adapt2Learn* then auto-generates the adaptive learning algorithm and exports it as a microcontroller script that designers can deploy onto their tools. The script actuates the tool's shape to maintain optimal task difficulty based on the performance of the learner sensed during training.

Additionally, *Adapt2Learn* has a built-in visualization tool that helps designers evaluate if their training tool is adapting appropriately to maintain the learners' training difficulty level at the optimal challenge point. The visualization displays a learner's performance and shows when the tool adapts to an easier or a more difficult setting. The designers can then use the insights from this visualization to further fine-tune the algorithm.

In summary, we contribute the following:

- A formative study that highlights the need to support designers in configuring a learning algorithm for their adaptive training tools for motor skills.
- A toolkit that supports designers in configuring a learning algorithm for their adaptive training tools. The toolkit has two parts - (1) a user interface to configure sensor and actuator values and to export a microcontroller script, and (2) a visualization tool to evaluate the adaptation during training to fine-tune the algorithm.
- Applications to demonstrate that *Adapt2Learn* can create learning algorithms for a range of adaptive training tools, such as an adaptive arm-band for golf, an adaptive wobbleboard, an adaptive bike, and adaptive heels.

2 RELATED WORK

Our work is related to HCI research on physical tools for motor skill learning and toolkits that support designers in building adaptive physical tools.

2.1 Physical Tools for Motor Skill Learning

In HCI, there are three approaches for designing physical tools that support motor skill learning: (1) multimodal feedback (audio, visual, tactile), (2) actuation for motor task correction, and (3) actuation for varying motor task difficulty.

Multimodal feedback systems support learners by giving them audio, visual, or vibrotactile feedback on their performance. For example, a multimodal feedback system to learn ballroom dancing measures the learner's step patterns using force sensors and provides them with audio feedback if they miss the beats (*Saltate!* [8]). Another example is a system to learn swimming that measures the learner's arm-strokes and generates a visualization of their temporal movement patterns, thereby highlighting mistakes in task execution [11]. Similar systems for learning karate [4], tai-chi [19], running [14] and ergonomic sitting [15] track learner's body gestures and provide vibrotactile, visual or audio feedback on faulty task executions. Multimodal systems can also support learning of motor skills involving the use of physical tools, such as skateboarding [17, 18], snowboarding [16], archery [9], golf [8] and tennis [2, 3].

Actuated tools for motor task correction help users by preventing mistakes and supporting correct execution. For instance, an actuated knife [23] can lock the blade if the knife gets too close to the user's finger while cutting. Similarly, an actuated office desk setup can actuate the desk, chair, and the monitor height to correct the sitting posture of the user (*ActiveErgo* [22]).

In contrast to these tools for motor task correction, actuated tools can also be used to vary the motor task difficulty during training. Turakhia et al. showed that actuation can be used to vary task difficulty to maintain learner's at the optimal challenge point where the potential learning benefit is the largest [21]. For example, an adaptive basketball that adapts the hoop height and width can keep the training task of shooting hoops at the optimal challenge point (i.e. neither too easy nor too difficult) during training.

To design the above training tools, designers require expertise in multiple different areas, i.e. they have to not only build the hardware

but also configure their tools to respond to the learner to facilitate learning. However, to date, only few support tools for designing physical training tools for motor skill learning exist.

2.2 Toolkits to Build Physical Adaptive Tools

Several researchers have developed tools to convert existing passive objects into objects with integrated sensing and actuation. For example, *Robiot* [13] is a design tool that converts everyday objects into actuated objects that can perform motor tasks like opening trashcans and drawers. Similarly, *Reprise* [6] and *Encore* [5] are design tools for generating customized 3D printed actuators, such as levers, to simplify motor tasks including opening bottles or turning faucets. Desai et al. [7] developed a design toolkit that auto-generates the assembly of the sensor and microcontroller components for custom designed devices, such as for a bottle crusher. *Retrofab* [20] is a toolkit that allows non-experts to design and fabricate retrofitted physical interfaces on top of existing objects by auto-generating enclosures for sensors and actuators.

Similarly, several toolkits have been developed that support designers in generating the electronic circuits for their actuated tools. *Trigger-action-circuits* [1] autogenerates the circuitry and firmware and assembly instructions for designers to build interactive devices. Similarly, *HeyTeddy* [12] guides users with hardware assembly by interactively providing the assembly steps to build a circuit.

While these toolkits focus on generating the electronics enclosure, circuitry and assembly instructions, we focus on supporting designers with the configuration of the adaptation algorithm to support learning at the optimal challenge point and the subsequent generation of the microcontroller script that designers can deploy onto their tools.

3 FORMATIVE STUDY

To study the challenges that designers face in building adaptive training tools that train learners at the optimal challenge point, we conducted a formative study.

3.1 Study Design

Our formative study was an 8-weeks long bi-weekly design studio as part of an undergraduate course at our institution. We had 32 participants who were in their junior or senior year at the university, and all had prior knowledge of electronics, prototyping, and programming (at least 2 years). The participants were teamed up in groups of two.

We asked the teams to design, prototype, and build an adaptive training tool for a motor skill of their choice. In the first week, the participants brainstormed 10 ideas each and then selected one idea per team to work on for the rest of the studio. Letting the teams choose the motor skill themselves allowed us to examine the design space of adaptive training tools for various motor skills and understand the most common challenges in building them.

During the 8-weeks study period, we informally interacted with the teams to discuss their design concepts, prototype implementation, and the algorithm development for their adaptive tools. The teams presented their progress and discussed the challenges with us during the studio. At the end of the 8-week design studio, we conducted semi-structured interviews to gain further insights by surveying 8 volunteering participants representing 7 teams, with two participants from the same team.

3.2 Challenges Identified

Our observations and participants' interview responses showed that they faced challenges in the following steps of developing adaptive training tools for motor skills:

Design Concepts: Choosing Sensors and Actuators: After selecting the idea for their adaptive training tool, the participant teams had to make design decisions, such as choosing sensors to sense learner's performance and actuators to vary the task difficulty through actuation.

They addressed this challenge of making design decisions by answering high-level questions, such as (1) what aspect of the motor skill is being learned? (2) which sensors are useful to monitor the performance of that aspect of the motor skill? (3) what kind of actuation varies the difficulty level of the learning tasks? (4) which actuators support that actuation?

Figure 2 shows how all the teams chose the sensors and actuators for their respective adaptive training tools by addressing the above questions. For example, to support learners with walking in high heels, a team designed adaptive heels (Figure 2-(6)). To monitor the stability during balancing in high heels, the team mounted two ultrasonic sensors, one on each side of the heels. By measuring the deviation between the two sensors' readings, they sensed the learner's balancing skills. If the deviation between the two sensors' readings was zero, the learner was stable, otherwise, the learner was out of balance. To vary the training task difficulty, the adaptive heels increased or decreased the height of the heels. To achieve this actuation, the team used servo motors.

Implementation: Prototyping the Hardware: After choosing the sensors and actuators for their adaptive training tools, the teams spent the next several weeks prototyping. During this prototyping phase, the teams encountered difficulties integrating the mechanisms and electronic components with their physical tools. For example, the team building an adaptive floaty found it challenging to integrate the pneumatic pump within their inflatable prototype (Figure 2-(1)). Because several toolkits exist to support designers with integrating electro-mechanical components with their prototypes (as described in Section 2.2), we did not focus on addressing these challenges.

Creating the Learning Algorithm: After building the hardware prototype of their adaptive training tool, the teams developed the learning algorithm for their respective motor skill learning. While developing the algorithm, the goal of the participants was to adapt the tool according to the learner's performance so that the training difficulty is at the optimal challenge point, i.e., neither too easy nor too difficult. To achieve this, the algorithm needs to use the sensor values to assess the learner's performance, compute the appropriate training task difficulty level for that learner, and adapt the tool using the respective actuators.

No.	Adaptive Training Tool	Motor Skill	Sensor	Actuator	Adaptation
01	Adaptive Floaty	floating at different depths	water pressure	servo, pneumatic pump	inflate/deflate the floaty
02	Adaptive Piano -1	playing correct octave scales	conductive pressure	servo motor	raise/lower the right keys
03	Adaptive Cornhole	aiming at the cornhole	camera, break beam	DC motor, servo	moves cornhole closer/further
04	Adaptive Skateboard	balancing the skateboard	accelerometer	stepper motor	decreases/increases board length
05	Adaptive Jumprope	jumping high	switch	stepper motor	decreases/increases rope speed
06	Adaptive Heels	walking in high heels	ultrasonic distance	servo motor	decreases/increases heel height
07	Adaptive Gloves	sign language gestures	flex	DC motor	increases/decreases flex support
08	Adaptive Pitching Mach	pitching ball	accelerometer	DC motor	changes aim of the pitch
09	Adaptive Drawing	drawing straight lines	camera	DC motor	increases/decreases sdwg upport
10	Adaptive Piano -2	playing right keys on time	switch	servo motor	raises/lowers the right keys
11	Adaptive Pingpong	hitting the pingpong ball	motion	servo motor	moves the backwall closer/further
12	Adaptive Meditation	sitting still during distractions	BCI	stepper motor	decreases/increases distraction
13	Adaptive Fencing	fencing at the right target	piezo electric	servo motor	increases/decreases target area
14	Adaptive Dartboard	aiming darts	piezo	stepper motor	decreases/increases the distance
15	Adaptive Juggling	juggling balls	camera	ERM vibration motor	increases/decreases vibration
16	Adaptive Pullups	completing pull ups	flex	DC motor	increases/decreases support

Figure 2: Formative study: (top) The 32 participant (16 teams) built adaptive training tools for various motor skills of their choice. While the teams successfully prototyped the adaptive training tools, we observed that the participants struggled with developing a learning algorithm for their adaptive training tools. (bottom) List of the sensors, actuators, and their corresponding adaptation used in the adaptive training tools.

However, we observed that several teams found this step most challenging, and only 5 out of 16 teams deployed a learning algorithm of some form, while the other 11 teams demonstrated the adaptation by hard-coding it.

Of the 5 teams that implemented the algorithm, we observed that each team had their own method of developing the learning algorithm. There was no standard way of developing the learning algorithm across adaptive tools of different motor skills or even

adaptive tools for the same motor skills. For example, two teams developed adaptive pianos with actuated keys for learning to play the piano. However, one team used a brute force method for adaptation, while the other team used an algorithm that optimized the adaptation by sensing performance over time. Furthermore, the teams also struggled to select the right threshold values and where unsure how to evaluate if their chosen threshold enabled learning as desired. For example, one adaptive piano team member said, *'...[the biggest challenge was] optimizing [the learning algorithm]. It's quite brute force at the moment...'*, while the other adaptive piano team's member said, *'...the biggest problem was not knowing what was right, so we didn't come up with [an] optimal solution...'*.

In summary, we found that the designers faced challenges in three areas - choosing the sensors and actuators, prototyping the hardware, and configuring the learning algorithm for their adaptive training tools. The first challenge can be addressed by using the four guiding questions outlined in the design concept section. For the second challenge, several support tools already exist that help designers integrate their mechanisms into their tools, as listed in the related work. However, there is no support for designers to configure the learning algorithm that maintains the optimal challenge point during training. In particular, the designers faced two challenges concerning the learning algorithm:

(1) configuring the learning algorithm for their respective adaptive training tool that varies the task difficulty and maintains it at the optimal challenge point
(2) evaluating the learning algorithm and the tool's adaptation

To address these challenges, we built *Adapt2Learn*, a toolkit that auto-generates the adaptive learning algorithm for designers' actuated tools for motor skill learning. To evaluate the algorithm, *Adapt2Learn* has a visualization tool that lets designers visualize the learner's performance and the tool's respective adaptation.

In the next section, we first briefly describe the learning algorithm from prior work that researchers developed to create adaptive training tools that maintain the training difficulty at the optimal challenge point. We then demonstrate how we can configure this learning algorithm with the *Adapt2Learn* toolkit with the example of an adaptive basketball stand.

4 BACKGROUND - LEARNING ALGORITHM

Turakhia et al. have recently proposed a learning algorithm for effective motor skill learning with adaptive training tools [21]. This learning algorithm maintains the learner's training difficulty at the optimal challenge point during the training. To achieve this, the algorithm uses the tool's sensor values to assess the learner's performance, computes the optimal training difficulty, and then varies the training difficulty by adapting the tool's shape. The following pseudocode 1 based on Turakhia et al.'s learning algorithm explains how the learning algorithm allows a learner to progress from beginner level to expert level while training at the optimal challenge point, i.e., when the task is neither too easy nor too difficult.

Turakhia et al. showed in two user studies that the adaptive training tools with the above-mentioned learning algorithm lead to higher learning gains compared to training with non-adaptive and manually adaptive training tools [21]. This shows the potential of the learning algorithm to benefit the learning of a range of motor skills. However, there are no tools to support designers in configuring such a learning algorithm for their specific adaptive training tools. As seen in our formative study, every designer has to configure the learning algorithm according to the sensors and actuators of their respective adaptive training tools. To resolve this, we built the *Adapt2Learn* toolkit.

Algorithm 1 Pseudocode of the Learning Algorithm

```
Initialize at the lowest task difficulty
while task difficulty ≠ highest do
    (1) Assess Learner's Performance          ▷ By measuring sensor_value
    if sensor_value ≥ threshold_value then
        attempt score = 1                     ▷ i.e. attempt = successful
    else
        attempt score = 0                     ▷ i.e. attempt = failed
    (2) Compute if Training is at the Optimal Challenge Point
    Calculate the running average of the score over x attempts/time period
    Calculate the current derivative of the running average
    (3) Update task difficulty                ▷ By adapting the tool
    if derivative = 0 then ▷ i.e. running average plateaued at a value
        if running_average ≥ 0.75 then
            increase task difficulty → by adapting harder
        else if running_average ≤ 0.25 then
            decrease task difficulty → by adapting easier
        else
            maintain task difficulty → no adaptation
    Repeat steps (1) (2) and (3) until task difficulty = highest
```

5 ADAPT2LEARN TOOLKIT - WALKTHROUGH

The Adapt2Learn toolkit consists of two parts: (1) the *Adapt2Learn* user interface for configuring the learning algorithm for adaptive training tools, and (2) the *Adapt2Learn* visualization tool that helps designers assess when the tool adapts and how it affects the learner's performance. We explain both components in the next section at the example of an adaptive basketball stand that we replicated from Turakhia et al.

The adaptive basketball stand supports learners in learning to score baskets. To monitor a learner's successful and missed baskets, the stand has two sensors - a piezo sensor and a switch sensor. The piezo sensor that is mounted on the board detects when the ball hits the board (Figure 3a). The switch sensor that is mounted on the basket detects when a ball goes inside the basket (Figure 3b). If both sensors do not record a reading, the stand counts the attempt as a complete miss. To vary the task difficulty of scoring the baskets, the stand adapts the hoop height and width. To increase or decrease the hoop height, the stand uses a stepper motor (Figure 3c). To widen or tighten the hoop width, the stand uses a servo motor (Figure 3d). For beginners, the stand is set at its lowest height and widest hoop. As the learners improve in their performance of scoring baskets, the hoop height is increased, and the hoop width is decreased.

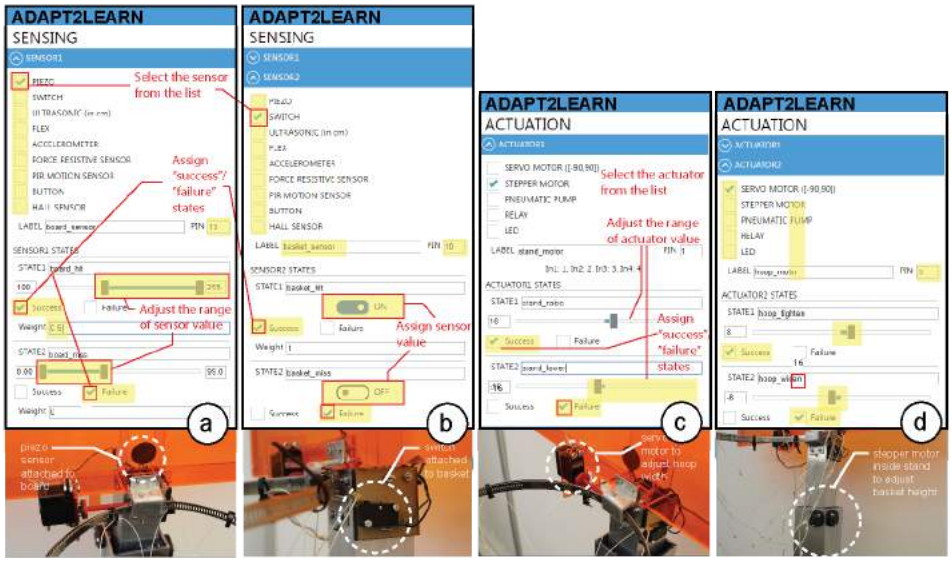

Figure 3: Configuring the learning algorithm using *Adapt2Learn*'s user interface by: (a,b) registering sensors and mapping sensor values onto success/failure states and corresponding scores, and (c,d) registering actuators and mapping actuator values onto success/failure states.

5.1 Configuring the Learning Algorithm using *Adapt2Learn*'s User Interface

To configure the learning algorithm, designers first register the sensors and then map the sensor values to success/failure states. They then repeat the process for actuators by registering the actuators and mapping the actuator values to success/failure states. Finally, they define how the performance should be evaluated by defining the evaluation unit and running average period. We next detail these steps for the example of the adaptive basketball stand:

Step 1: Register Sensors: We start by clicking the *'create new adaptive tool'* button and proceed to register the sensors. In the user interface, we can select among a range of different sensors, such as piezo, switch, ultrasonic, flex, accelerometer, force resistive, PIR motion, and hall sensors. For our purposes, we select the *'piezo'* sensor, label it *'board_sensor'*, and assign it pin 13 on our microcontroller (Figure 3a). We use the *'add another sensor'* button and repeat the procedure. We select the *'switch'* from the available sensors and label it as *'basket_sensor'* and assign it pin 10 on our microcontroller (Figure 3b).

Step 2: Map Sensor Values onto Success/Failure States: Next, we configure the sensor states and the respective threshold values that define successful and unsuccessful performance. Depending on the sensors chosen, the user interface provides the respective range of sensor values. For example, for the piezo sensor, the user interface provides a range of values (0-255) whereas for the switch sensor, it provides only boolean values (true/false).

We define states for the *'board_sensor'* (piezo sensor): *'board_hit'* with sensor values of *100 or more* units, and *'board_miss'* with sensor values of *0-99* units (Figure 3a). We mark *'board_hit'* as a *'success'* state and *'board_miss'* as a *'failure'* state using the checkboxes. By default, the algorithm scores every successful attempt as *'1'* and every failed attempt as *'0'*. However, because we want to count a board hit as a partially successful attempt, we manually assign it a score of *'0.5'* by updating the *'weight'* slider.

We then define the sensor states for the *'basket_sensor'* (switch): *'basket_hit'* as *'success'* whenever the sensor reads *'true'*, and *'basket_miss'* checked as *'failure'* whenever the sensor reads *'false'* (Figure 3b). We keep the default success and failure score weights of *'1'* and *'0'*, respectively.

Step 3: Register Actuators: Next, we register the actuators. In the user interface, we can select among a range of different actuators, such as a servo motor, stepper motor, pneumatic pump, and relay. Depending on the actuators chosen, the user interface provides the respective range of actuator values.

For our purposes of raising and lowering the hoop height, we select *'stepper motor'* from the list of actuators and label it *'stand_motor'*. We assign it pin 1 on our microcontroller, and the user interface automatically assigns the remaining pins 2, 3, and 4 required for the stepper motor (Figure 3c).

We then add the motor that widens and tightens the hoop by clicking *'add another actuator'* button. We then select *'servo motor'* and label it *'hoop_motor'*, and then assign it pin 9 on the microcontroller (Figure 3d).

Step 4: Map Actuation Values onto Success/Failure States: Next, we map the motors' actuation values onto adaptation states. We define states for the *'stand_motor'* (stepper motor): *'stand_raise'* as *'success'* and with the motor turning 16 revolutions to increase the hoop height, and *'stand_lower'* as *'failure'* and with the motor turning -16 revolutions to decrease the hoop height (Figure 3c). We repeat the process for the hoop motor by defining *'hoop_motor'*

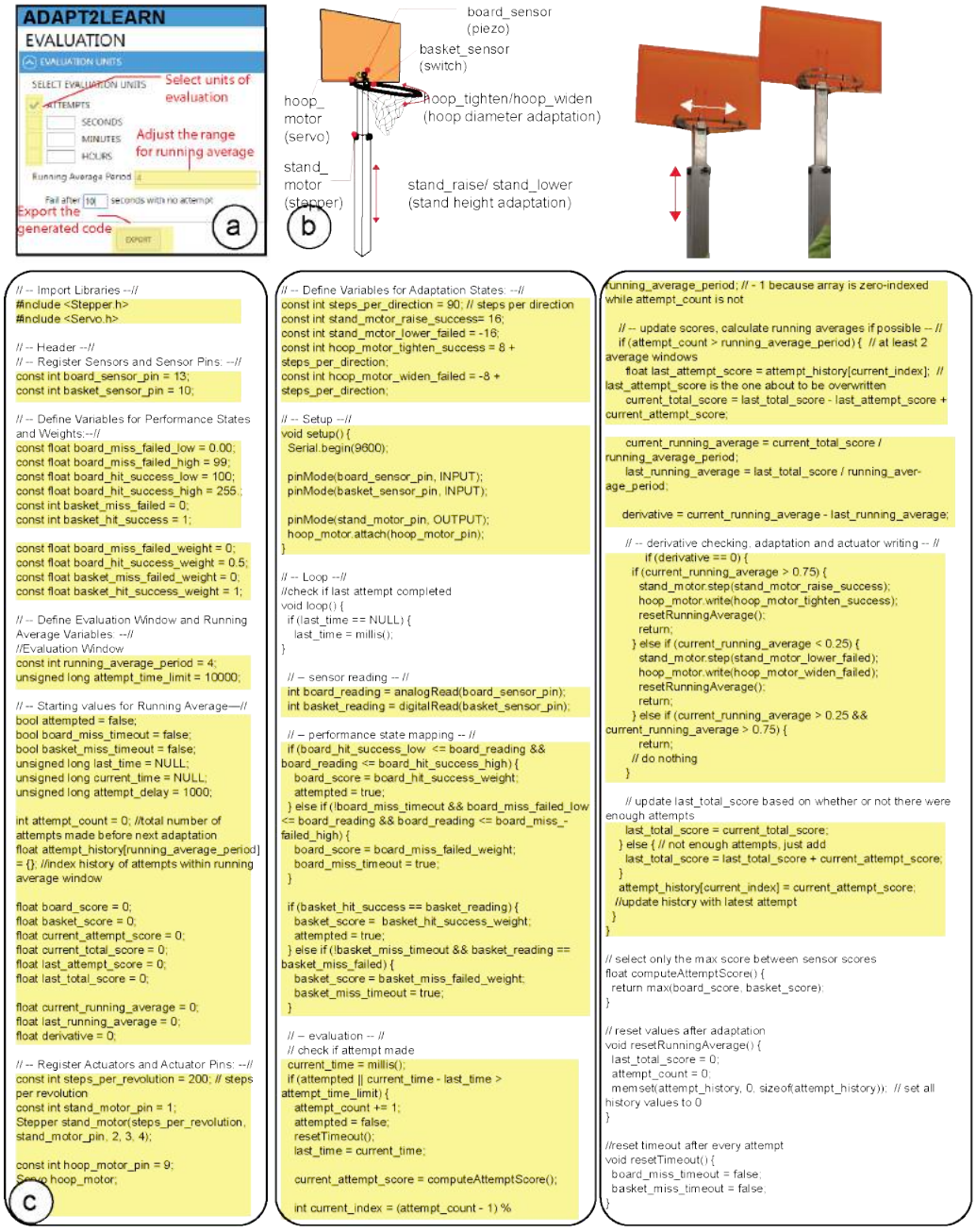

Figure 4: (a) Setting the evaluation units and the running average period, then exporting the microcontroller code and (b) deploying the microcontroller code onto the adaptive tool, i.e. the basketball stand. (c) The generated microcontroller code from the UI that automatically maps the readings from the sensor to actuator settings for adaptation of the physical tool.

(servo motor): *'hoop_tighten'* as *'success'* with the motor turning 8 revolutions, and *'hoop_widen'* as *'failure'* with the motor turning -8 revolutions (Figure 3d).

Step 5: Define Performance Evaluation Unit and Running Average Period: Finally, we set up the performance evaluation unit (Figure 4a). The *'evaluation unit'* can be either *'attempts'* or *'time'*. For our adaptive basketball prototype, we select *'attempts'* representing attempted throws at the basket. Next, we define the *'running average period'*, i.e. the period over which the algorithm will evaluate the learner's performance. Depending on which evaluation unit was selected, the running average period is either a number of attempts (after 4 throws in our basketball example) or a time period (after 10 minutes when balancing a bike). We also

set the time limit after which, if no sensor value is detected, the attempt is considered as a failure, for example as 10 seconds.

5.2 Generating the Microcontroller Script According to the Configuration

After configuring the learning algorithm using the steps described above, designers can hit the 'export' button (Figure 4a), which automatically generates the microcontroller code (Arduino script in .ino file format). After exporting, designers can then deploy the script onto the microcontroller integrated with their adaptive training tools.

In our example, the exported microcontroller script can be seen in the Figure 4c. We then uploaded the script to the basketball stand to make it adaptive (Figure4b).

5.3 Visualization Tool: Displaying Performance and Adaptation

To provide tool designers with a way to assess the learner's performance and when the tool adapts, we developed a visualization tool. The visualization tool plots the learner's attempt scores, the corresponding running average, and the computed derivative of the running average at that attempt. This performance data is plotted in real-time along with when the tool adapts to an easier or more difficult setting.

Figure 5 shows the visualization of plotting all performance scores of a learner for the adaptive basketball. The attempt scores are plotted between 0 to 1, as configured by us in the algorithm with a basket scored as 1pt, a board hit as 0.5pt, and a miss as 0pt. Figure 5 also plots the corresponding running average, derivative, and tool adaptation.

Because we configured the running average period to be 4 attempts and the first four throws were baskets scored, the running average equaled 1.0. In the 5th attempt, the learner scored the basket again, and the running average for attempt 2-5 also equaled 1.0. Thus, the derivative over the two running averages was computed as zero, which indicates a potential tool adaptation (as per the Algorithm 1). At this attempt, because the running average was above 0.75pts, the tool adapted to the next harder difficulty setting by executing the configured adaptation states - *'hoop_tighten'* and *'stand_raised'*. The tool adapted again to a more difficult setting in the same way at attempts 10 and 14.

From the 15th attempt, the learner started getting more challenged with the higher task difficulty, and the running average dropped between 0.25pts to 0.75pts. However, by the 29th attempt, the participant improved and their score increased. At this point, the derivative became zero and the moving average was at 0.75pts. Thus, the tool adapted again to the next difficulty setting. Similarly, the tool adapted at attempts 37, 42, and 47.

At attempts 55, 56, and 57, the performance plateaued again, but the running average was not high enough (0.62pts) to initiate an adaptation. Thus, the difficulty level was maintained at the same level to allow for more training. Finally, after a few additional attempts at attempt 65 and 75, the derivative became zero again, and performance now had increased to above 0.75pts, thus the tool adapted to the next difficulty level.

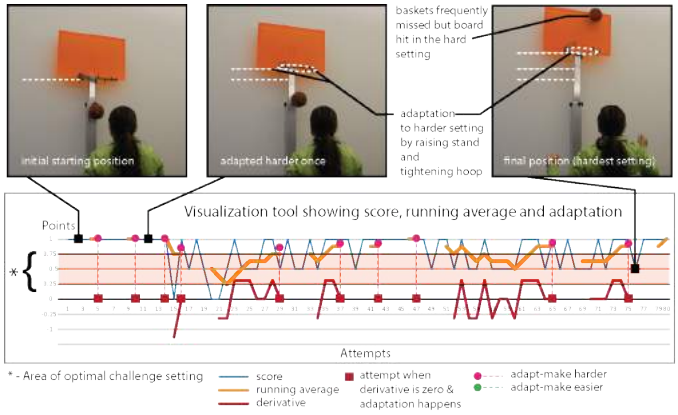

Figure 5: Visualization of the scoring of a learner and the adaptation frequency over a number of attempts. The visualization thus helps monitor how the configured learning algorithm takes a learner from a low difficulty setting to a high difficulty setting while maintaining their performance score at the optimal challenge point.

The visualization helps monitor how the configured learning algorithm takes a learner from a low difficulty setting to a high difficulty setting while maintaining their performance score at the optimal challenge point.

6 APPLICATIONS OF ADAPTIVE TRAINING TOOLS USING *ADAPT2LEARN*

In order to demonstrate that our toolkit can be used to configure the learning algorithm for a variety of adaptive training tools, we first built and then configured the learning algorithm for an adaptive armband for golf, a wobble-board, a bike with adaptive training wheels, and a pair of adaptive heels. These examples demonstrate different sensor-actuator combinations and different ways of configuring them.

6.1 Adaptive Armband for Golf: Single Sensor-Actuator Combination

We built and configured an adaptive armband that supports learners in keeping their elbow straight during a golf-swing. The armband has a flex sensor to detect if the learner's elbow is straight or bent, and a pneumatic pump to deflate and inflate the arm band to restrict bending of the elbow (Figure 6).

The adaptive armband has a single sensor-actuator combination, i.e., one flex sensor and one pneumatic pump. For the flex sensor values, we configured (Figure 6) the success state as as 0-5° (*'arm_straight'*) and the failure state as 6-180° (*'arm_bent'*). For the pneumatic pump, we configured the success state as -100 units (*'band_deflate'*) and the failure state as +100 units (*'band_inflate'*). We set the evaluation unit as *'attempts'* and set the time for each attempt as 4 seconds. In this way, we can convert a continuous sensor value reading over a period of time into several discrete readings.

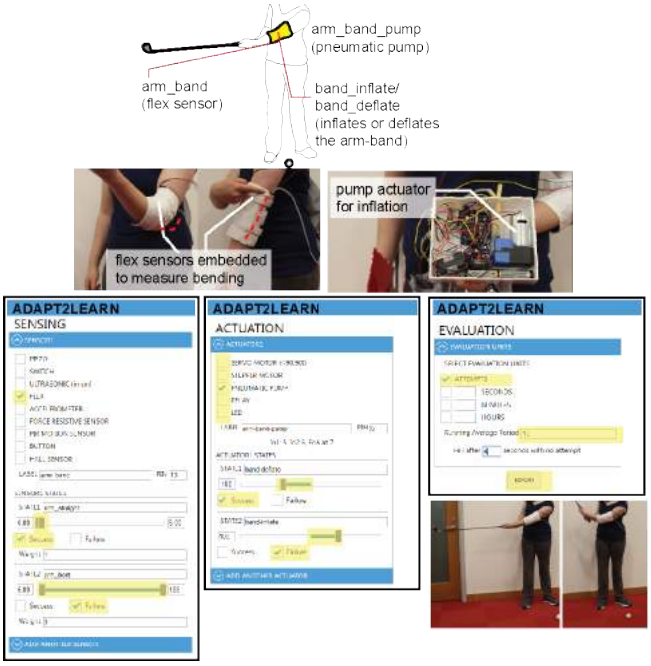

Figure 6: An adaptive armband that supports learners in keeping their elbow straight during a golf-swing, consisting of one flex sensor and one pneumatic pump. Configuring the learning algorithm for the armband using *Adapt2Learn*'s user interface

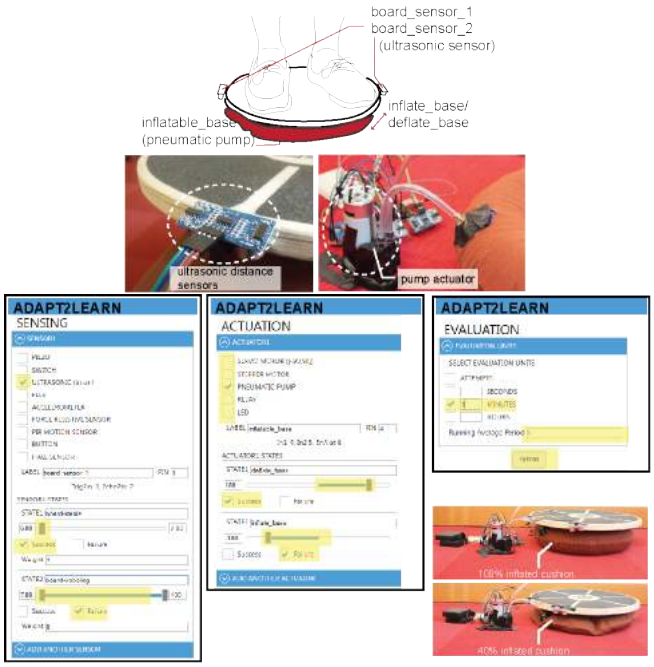

Figure 7: An adaptive wobbleboard with inflatable cushion that supports learners in learning to balance the board. Configuring the learning algorithm for the wobbleboard using *Adapt2Learn*'s user interface to set synchronized two ultrasonic sensors and one pneumatic pump.

The configured algorithm then senses the bending and inflates or deflates the armband to provide more or less support to the learner and thus varies the task difficulty. For example, if the learner bends the elbow too often during training, the algorithm makes the task difficulty easier by inflating the armband thereby restricting the bending, and thus providing more support to the learner by keeping the elbow straight.

6.2 Adaptive Wobbleboard: Synchronizing Sensors

We built and configured an adaptive wobbleboard with inflatable cushion that supports learners in learning to balance the board. The wobbleboard has two ultrasonic sensors mounted on diametrically opposite sides of the board to detect if it is stable or wobbling, and a pneumatic pump to deflate and inflate the support cushion that restricts wobbling (Figure 7).

The adaptive wobbleboard consists of a combination of two synchronized sensors and one actuator, i.e., two ultrasonic sensors and one pneumatic pump. The sensors are mounted on the wobbleboard such that they detect the distance from the ground. To set the ultrasonic sensor values (Figure 7), we configured the two sensors' success states as 5-7 cm (*'board_stable'*), which is the wobbleboard's height when balanced, and the failure states as 7-400 cm (*'board_wobble'*). For the pneumatic pump, we configured the success state as -100 units (*'cushion_deflate'*) and the failure state as +100 units (*'cushion_inflate'*). For the evaluation unit, we set the 'time period' to 1 minute, thus measuring a continuous performance instead of discrete attempts.

If the configured algorithm detects a failure state for either of the sensors, it implies that the corresponding edge of the wobbleboard is too high, and the other edge is too low (<5cm), meaning that the wobble board is imbalanced. Thus, two sensors can be used in tandem to detect balancing. To support the learner in keeping the wobbleboard balanced, the pneumatic pump can then inflate the cushion. Alternatively, if the learner balances the wobbleboard well, the pump deflates the cushion, thereby reducing the support and making the task of balancing harder.

6.3 Adaptive Bike: Synchronizing Actuators

We built and configured a bike with adaptive training wheels that supports learners in learning to balance the bike. The bike has one hall-effect sensor mounted on each of the training wheels to detect if the training wheel is being used, and one stepper motor on each of the training wheels to lower or raise them to provide more or less support in balancing the bike (Figure 8).

The adaptive bike, therefore, consists of a combination of two synchronized sensors and two synchronized actuators, i.e., two hall-effect sensors and two stepper motors. The hall-effect sensors are mounted on the axles of the training wheels and one magnet is mounted on the corresponding rim of each training wheel. If the hall-effect sensor detects the magnet, it implies the wheel is rotating and thus the training wheel is being used for riding. To

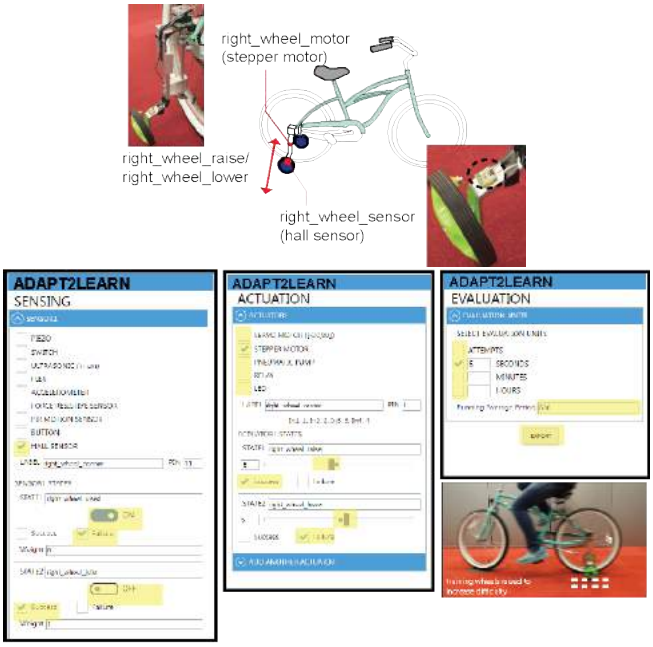

Figure 8: A bike with adaptive training wheels that supports learners in learning to balance the bike. Configuring the learning algorithm for the adaptive bike using *Adapt2Learn*'s user interface to set two synchronized hall-effect sensors and two synchronized stepper motors.

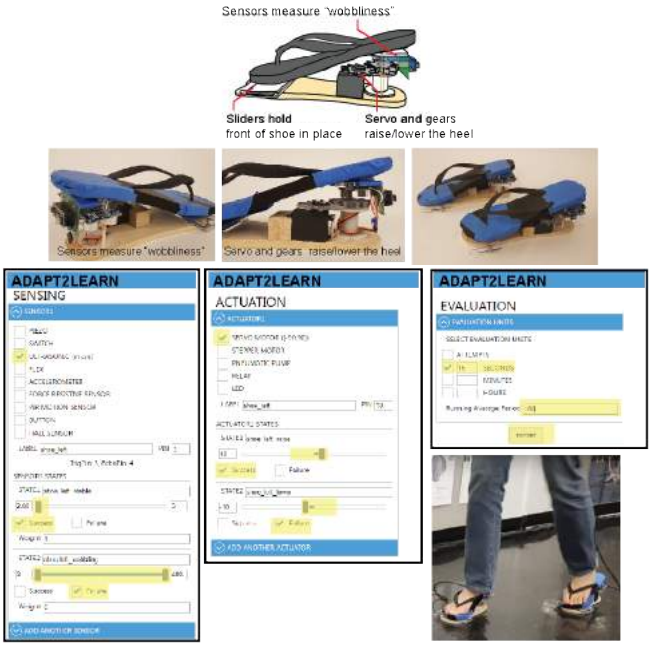

Figure 9: Adaptive heels that support learners in training to walk in high heels. Configuring the learning algorithm for the adaptive heels using *Adapt2Learn*'s user interface to set four synchronized ultrasonic distance sensors (two per shoe), and two synchronized servo motors.

set the hall effect sensor values, we configured (Figure 8) the two sensors' success states as *'true'* (*'wheel_idle'*) and failure states as *'false'* (*'wheel_used'*). Unlike the sensors used in the prior examples that detect a continuous value range, the hall-effect sensor detects boolean values. For the stepper motor, we configured the success state as -5 units (*'wheel_raise'*) and the failure state as +5 units (*'wheel_lower'*). For the evaluation unit, we set the *'time period'* to 5 seconds and the running average period to 600 seconds, thus measuring a longer performance window, i.e., performance over 120 data points.

If the configured algorithm detects a failure state too often for either of the sensors, it implies that the learner is unable to balance without the use of the training wheels. The stepper motors then lower the wheels further to provide more support to the learner. Since both the actuators are mapped to the same failure state, they both turn at the same time and by the same amount. In this way two sensor values can be mapped to two actuator values in combination.

6.4 Adaptive Heels: Synchronized Sensors and Actuators

In addition to the above examples, we also configured the learning algorithm for the studio-built adaptive heels that support learners in training to walk in high heels (Figure 9). The participant team mounted two ultrasonic distance sensors per shoe, one on each side of the heel to measure the balance of the learner while walking in the heels. One servo motor was mounted on each of the shoe to raise and lower the heel height while walking. Thus, the adaptive heels had a combination of 4 synchronized sensors and 2 synchronized actuators. This ensures that both the heels were synchronized in their adaptation. Thus only when all the four sensor values detected success states, the servo motors actuated to raise the heels and increased the difficulty of walking. Note that the 3D printed spindle that increased and decreased its height by the servo motor at the base of the heel supported the weight of the learner while walking.

In the same way that we used our user interface to configure the examples above, the user interface can be used to configure other examples from the studio, such as the adaptive skateboard, dartboard, fencing, jumprope, and cornhole prototypes (Figure 2) that use similar sensor-actuator combinations.

However, we also encountered two challenges for which we could not yet configure the learning algorithm using our user interface. The first challenge occurs when the success state is coupled with a specific timing, such as when hitting a note on time for playing piano. For instance, the adaptive piano used a switch sensor to sense if a key was pressed at the right time and then actuated the servo motor under the key to provide feedback to the learner on which key to press next. Since our user interface does not support time-based sensing, we were not able to configure the learning algorithm for this adaptation. The second challenge occurs when additional processing on the sensor data is needed. For example, both the adaptive pitching machine and the adaptive juggling used a camera to detect the learner's position, which requires computer vision techniques that go beyond the sensor value thresholding that our user interface currently supports.

In summary, we demonstrated the use of *Adapt2Learn* for configuring the learning algorithm for a variety of applications that ranged from single sensor-actuator combinations (adaptive golf-arm band), multiple synchronized sensors (adaptive wobbleboard), multiple synchronized actuators (adaptive bike), to multiple synchronized sensor-actuator combinations (adaptive heels).

7 IMPLEMENTATION

The *Adapt2Learn* user interface is a web application (HTML, CSS, and JavaScript) packaged as a stand-alone application. The visualization tool is implemented in Python and uses the Matplotlib library. The visualization tool communicates with an adaptive tool in real-time by reading from the tool's microcontroller (Arduino) serial port via Bluetooth.

Currently, *Adapt2Learn* supports 8 different sensors (piezo, switch, ultrasonic, flex, accelerometer, force resistive, PIR motion, and hall-effect sensors) and 4 different actuators (servo motor, stepper motor, pneumatic pump, and relay). These components were also most frequently used in the adaptive training tool prototypes (10 out of 16 teams) in our formative study. Sensors and actuators that follow common pin and code conventions (e.g., a switch) work with our user interface regardless of the specific manufacturer model. Components that read/write analog values, such as ultrasonic sensors and motors may require adjusting the mapping values with different models based on their specifications. Components that require special libraries or setup code, such as non-generic motor controllers, are currently not supported but can be added to the user interface in the future by providing separate code for them.

8 DISCUSSION

We illustrated how *Adapt2Learn* supports designers in configuring the learning algorithm for their custom adaptive training tools. *Adapt2Learn's* built-in visualization tool then supports designers in assessing the learner's performance and the tool's adaptation. The interface also allows designers to update the learning algorithm without re-programming the microcontroller code. We next discuss the limitations of our toolkit and provide directions for future work:

Extending the Range of Supported Components: *Adapt2Learn* currently supports 8 sensors and 4 actuators, which can be used in multiple sensor-actuator combinations, as seen in our examples. However, as discussed earlier, providing more components would further extend the range of adaptive tools for configuring the learning algorithm. For the future, we plan to integrate components that require more processing, such as depth sensors and cameras. Additionally, adding time-based sensing and custom components to the user interface is an important direction for future work.

Configuring the Algorithm in Real-time: While currently, our system provides real-time visualization of the learner's performance and tool's adaptation, it does not allow for real-time reconfiguration of the learning algorithm. The designers currently have to reconfigure the values, re-export the microcontroller script, and then deploy it again onto the adaptive tool. In future work, we plan to support designers to update the configuration of the learning algorithm in real-time in the context of the learning situation.

Evaluating the toolkit through user studies: While we demonstrated that *Adapt2Learn* can be used for configuring various adaptive training prototypes, evaluating the use of toolkit through user studies with designers and testing it in different phases of the design process is a part of our future work.

Visualizing the Learning Trajectory: While not the focus of our work, the visualization tool may also help assess how long the learner takes to transition from a low difficulty level to a high difficulty level, and predict the time needed to reach the highest skill level. Additionally, the visualization tool may also allow comparing the learning trajectory of multiple learners and gain more insights into that motor skill's learning.

Comparing Different Tool Designs: When building an adaptive training tool, designers have different options for sensing the learner's performance and adapting the task difficulty. For instance, when designing the adaptive basketball, instead of only detecting board and basket hits with a piezo sensor and switch, a camera can be used to sense the ball's trajectory, which provides more information. However, it is unclear which sensing-adaptation method leads to the best results. Providing a way to compare the adaptation of different designs for the same training tool could allow designers to choose their designs appropriately.

Supporting Multiple Learners: Many skills involve learning as a group where individuals may have varying skill levels. While currently the exported microcontroller script from our user interface and our visualization tool monitor a single learner's performance, a future direction for research could be to extend both the user interface and the visualization tool to support multiple learners.

9 CONCLUSION

In this paper, we presented *Adapt2Learn*, a toolkit that supports designers in creating adaptive training tools that maintain the task difficulty at the optimal challenge point. Our formative study showed that designers needed support, particularly in configuring the learning algorithm and assessing the tool's adaptation. We demonstrated that *Adapt2Learn* addressed these two challenges through its user interface and its visualization tool. We showed that *Adapt2Learn's* user interface supports configuring the learning algorithm by first registering the sensors and actuators of the adaptive tools, then mapping their values to success/failure states, and finally exporting the auto-generated micro-controller script, which can be deployed onto the micro-controller integrated with the tools. Furthermore, we showed how *Adapt2Learn's* built-in visualization tool supports designers in assessing if the learning algorithm maintains the task difficulty at the optimal challenge point during training by visualizing the learner's performance and the tool's adaptation. We demonstrated *Adapt2Learn's* use to configure the learning algorithm for five different adaptive tools with various sensor/actuator combinations, such as an adaptive basketball, armband for golf, wobbleboard, bike, and adaptive heels.

ACKNOWLEDGMENTS

We thank the 32 students at MIT, who took the 6.810 course and participated in the study. We thank Junyi Zhu for his help with the viztool. We also thank Or Oppenheimer and Christian De Weck for their contribution in building the prototypes. This work is supported by MIT Learning Initiative and the National Science Foundation under Grant No. 1844406.

REFERENCES

[1] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. Trigger-action-circuits: Leveraging generative design to enable novices to design and build circuitry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 331–342. https://doi.org/10.1145/3126594.3126637

[2] Arnold Baca and Philipp Kornfeind. 2004. Real time detection of impact positions in table tennis. *Journal of The Engineering of Sport* 5 (2004), 508–514. https://doi.org/10.1109/MPRV.2006.82

[3] Peter Blank, Thomas Kautz, and Bjoern M Eskofier. 2016. Ball impact localization on table tennis rackets using piezo-electric sensors. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*. 72–79. https://doi.org/10.1145/2971763.2971778

[4] Aaron Bloomfield and Norman I Badler. 2008. Virtual training via vibrotactile arrays. *Journal of Presence: Teleoperators and Virtual Environments* 17, 2 (2008), 103–120. https://doi.org/10.1162/pres.17.2.103

[5] Xiang'Anthony' Chen, Stelian Coros, Jennifer Mankoff, and Scott E Hudson. 2015. Encore: 3D printed augmentation of everyday objects with printed-over, affixed and interlocked attachments. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 73–82. https://doi.org/10.1145/2787626.2787650

[6] Xiang'Anthony' Chen, Jeeeun Kim, Jennifer Mankoff, Tovi Grossman, Stelian Coros, and Scott E Hudson. 2016. Reprise: A design tool for specifying, generating, and customizing 3D printable adaptations on everyday objects. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 29–39. https://doi.org/10.1145/2984511.2984512

[7] Ruta Desai, James McCann, and Stelian Coros. 2018. Assembly-aware design of printable electromechanical devices. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 457–472. https://doi.org/10.1145/3242587.3242655

[8] Hassan Ghasemzadeh, Vitali Loseu, Eric Guenterberg, and Roozbeh Jafari. 2009. Sport training using body sensor networks: A statistical approach to measure wrist rotation for golf swing. In *Proceedings of the Fourth International Conference on Body Area Networks*. 1–8. https://doi.org/10.4108/ICST.BODYNETS2009.6035

[9] Heng Gu, Kai Kunze, Masashi Takatani, and Kouta Minamizawa. 2015. Towards performance feedback through tactile displays to improve learning archery. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*. 141–144. https://doi.org/10.1145/2800835.2800893

[10] Mark A Guadagnoli and Timothy D Lee. 2004. Challenge point: a framework for conceptualizing the effects of various practice conditions in motor learning. *Journal of motor behavior* 36, 2 (2004), 212–224. https://doi.org/10.3200/JMBR.36.2.212-224

[11] Daniel James, David Rowlands, and James Lee. 2014. Visualization of wearable sensor data during swimming for performance analysis. *Sports Technology* 6 (02 2014). https://doi.org/10.1080/19346182.2013.867965

[12] Yoonji Kim, Youngkyung Choi, Daye Kang, Minkyeong Lee, Tek-Jin Nam, and Andrea Bianchi. 2019. Heyteddy: Conversational test-driven development for physical computing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (2019), 1–21. https://doi.org/10.1145/3369838

[13] Jiahao Li, Jeeeun Kim, and Anthony Chen. 2019. Robiot: A Design Tool for Actuating Everyday Objects with Automatically Generated 3D Printable Mechanisms. 673–685. https://doi.org/10.1145/3332165.3347894

[14] Stina Nylander, Mattias Jacobsson, and Jakob Tholander. 2014. Runright: real-time visual and audio feedback on running. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems (CHI'14)*. 583–586. https://doi.org/10.1145/2559206.2574806

[15] Craig O'Neil, Mark D Dunlop, and Andrew Kerr. 2015. Supporting sit-to-stand rehabilitation using smartphone sensors and arduino haptic feedback modules. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*. 811–818. https://doi.org/10.1145/2786567.2793705

[16] Hyung Kun Park and Woohun Lee. 2016. Motion Echo Snowboard: Enhancing Body Movement Perception in Sport via Visually Augmented Feedback. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS'16)*. 192–203. https://doi.org/10.1145/2901790.2901797

[17] Hyung Kun Park, HyeonBeom Yi, and Woohun Lee. 2017. Recording and sharing non-visible information on body movement while skateboarding. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17)*. 2488–2492. https://doi.org/10.1145/3025453.3025476

[18] Sebastiaan Pijnappel and Florian 'Floyd' Mueller. 2014. Designing interactive technology for skateboarding. In *Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction (TEI'14)*. 141–148. https://doi.org/10.1145/2540930.2540950

[19] Otniel Portillo-Rodriguez, Oscar O Sandoval-Gonzalez, Emanuele Ruffaldi, Rosario Leonardi, Carlo Alberto Avizzano, and Massimo Bergamasco. 2008. Real-time gesture recognition, evaluation and feed-forward correction of a multimodal tai-chi platform. In *International Workshop on Haptic and Audio Interaction Design*. Springer, 30–39. https://doi.org/10.1007/978-3-540-87883-4_4

[20] Raf Ramakers, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2016. Retrofab: A design tool for retrofitting physical interfaces using actuators, sensors and 3d printing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 409–419. https://doi.org/10.1145/2787626.2787650

[21] Dishita Turakhia, Yini Qi, Lotta-Gili Blumberg, Andrew Wong, and Stefanie Mueller. 2021. Can Physical Tools that Adapt their Shape based on a Learner's Performance Help in Motor Skill Training?. In *Proceedings of the 15th International Conference on Tangible, Embedded and Embodied Interaction (TEI'21)*. https://doi.org/10.1145/3430524.3440636

[22] Yu-Chian Wu, Te-Yen Wu, Paul Taele, Bryan Wang, Jun-You Liu, Pin-sung Ku, Po-En Lai, and Mike Y Chen. 2018. Activeergo: Automatic and personalized ergonomics using self-actuating furniture. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI'18)*. 1–8. https://doi.org/10.1145/3173574.3174132

[23] Amit Zoran, Nan-Wei Gong, Roy Shilkrot, Shuo Yan, and Pattie Maes. 2015. Cutting Edge Vision: Metal Embedded Optics for Smart Knives. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. 1223–1228. https://doi.org/10.1145/2702613.2732495