Mobiot: Augmenting Everyday Objects into Moving IoT Devices Using 3D Printed Attachments Generated by Demonstration

Abul Al Arabi
HCIED Lab, Texas A&M University
abulalarabi@tamu.edu

Jiahao Li
UCLA HCI Research
ljhnick@g.ucla.edu

Xiang 'Anthony' Chen
UCLA HCI Research
xac@ucla.edu

Jeeeun Kim
HCIED Lab, Texas A&M University
jeeeun.kim@tamu.edu

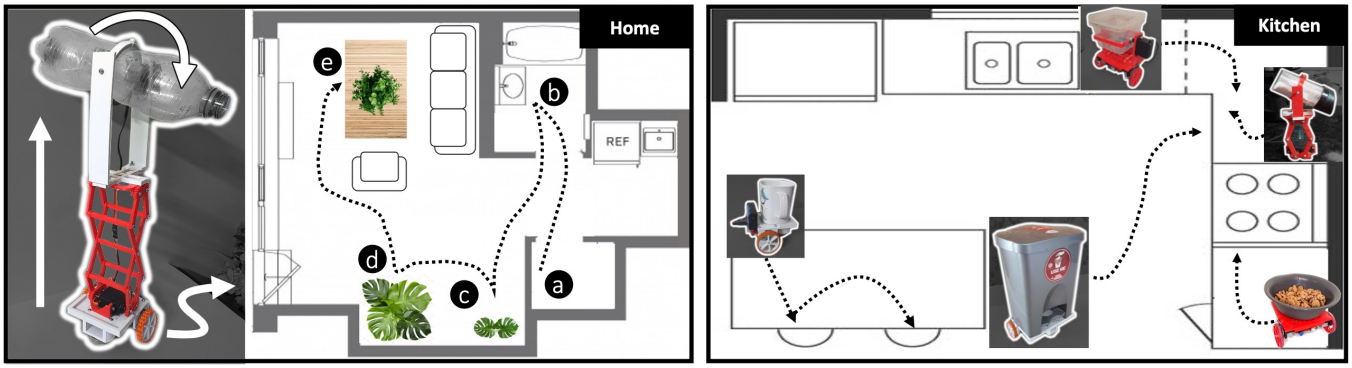

Figure 1: Using Mobiot, a user can add actuation mechanisms to mobilize legacy objects using 3D printing, for example, (left) a bottle to water plants while the user is away for vacation. An automated plant-watering mechanism can navigate the floor to (a) fetch at a scheduled time, (b) refill water from the tap, then (c-e) water the pots by rotating the bottle on the top and adjusting the height to reach the one on the coffee table. (right) In a kitchen, a user can build smart cooking aids with collective agent objects, a bowl fetching water, spice cans sprinkling salt and peppers, and a dish and a mug serving ready-to-eat foods as well as moving a trash bin next to a counter closer to where the food scraps are.

ABSTRACT

Recent advancements in personal fabrication have brought novices closer to a reality, where they can automate routine tasks with mobilized everyday objects. However, the overall process remains challenging- from capturing design requirements and motion planning to authoring them to creating 3D models of mechanical parts to programming electronics, as it demands expertise.

We introduce Mobiot, an end-user toolkit to help non-experts capture the design and motion requirements of legacy objects by demonstration. It then automatically generates 3D printable attachments, programs to operate assembled modules, a list of off-the-shelf electronics, and assembly tutorials. The authoring feature further assists users to fine-tune as well as to reuse existing motion libraries and 3D printed mechanisms to adapt to other real-world objects with different motions. We validate Mobiot through application examples with 8 everyday objects with various motions applied, and through technical evaluation to measure the accuracy of motion reconstruction.

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04…$15.00
https://doi.org/10.1145/3491102.3517645

CCS CONCEPTS

• **Human-centered computing → Interactive systems and tools**.

KEYWORDS

Personal Fabrication, Home-automation, Motion planning

ACM Reference Format:
Abul Al Arabi, Jiahao Li, Xiang 'Anthony' Chen, and Jeeeun Kim. 2022. Mobiot: Augmenting Everyday Objects into Moving IoT Devices Using 3D Printed Attachments Generated by Demonstration. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3491102.3517645

1 INTRODUCTION

Recent development in IoT systems is accelerating the transformation of the smart assistive home from fiction into reality. While we observe plentiful sensors and controllers that facilitate a new home with ubiquitous systems, such as the intelligent thermostat making

the room habitable automatically, mobile agents have not permeated our everyday life. Everyday objects with mobility can ease and automate daily routine tasks [2, 58], such as self-organizing workspace, moving the trash away, self-serving kitchen [10], ambulating items to different places, and beyond. Automotive agents can also support accessible homes, catering elderly care, such as assisting indoor navigation, auto-serving medicines, foods, drinks at the preset schedule, and many more [2].

The question *"Why are we not living yet with robots?"*[31] could be answered with possible reasons such as: (1) we did not domesticate robots; (2) in reality, substituting all the legacy objects with commercial robotic platforms is not feasible, (3) end-users tools are lacking to support designing and customizing smart agents, such as capturing real-world design requirements, (re)configure them for fabrication upon varying use-case scenarios using interactive systems at hand. While prior works tackled robotic-locomotion by off-the-shelf platforms[13, 30], they are often expensive and designed for domain experts. Also, the rigidity of commercial robotic platforms does not offer personalizations unless compensated with an expensive or sophisticated but versatile set of end-effectors. For example, to create a simple self-shaking spice jar, a user needs to purchase a multi-DOF robotic arm and perform kinematics programming, while there barely exists any accessible support for end-users to program ambulation from place to place.

The concert of the latest research in end-user programming and personal fabrication aims at transforming users into makers [39, 53] by assisting end-users with the process of augmenting personal belongings [34], building personal IoT devices actuated from legacy objects [37], auto-generating robotic movements in existing 3D objects [36], achieving easy programming of smart home appliances [4], and more. Recent advances have brought the story of *"we can start with today's existing devices and slowly add on intelligence, manipulative ability, and function"* [42] upfront, leading towards the future of end-user robotics. These unveil the chapter of ubiquitous robotics and potential participation of end-users in everyday robotics projects afforded by personal fabrication that scaffolds personal creative endeavors [33]. Nonetheless, end-users, especially novices, are not yet introduced to a direct or easy interaction space that supports the design and fabrication of personal robotics. The chronicle of adding mobility portrays three salient challenges:

- Most everyday objects are passive in nature, yet existing solutions (e.g., IoT devices or robotic platforms) are either not accessible or unaffordable;
- Every individual has different needs, from the physical dimension of personal objects to needed motions to accommodate unique lifestyles and support creative set-ups of personalized smart agents;
- Capturing individual design requirements to fabricate and program personal smart objects is challenging as it implies that an end-user must obtain custom specifications about the target object, including its shape, physics, movement paths, as well as electronics.

We present Mobiot, an end-user toolkit to mobilize everyday objects by auto-generated 3D printable attachments from a single demonstration. Incorporating an end-to-end pipeline for fabricating mobile agents with desired motions (roaming, lifting, and rotating) and using a commercial smartphone as a medium for both designing & action authoring, Mobiot automatically captures all design requirements and decomposes into 3D printable attachment designs and associated electronics specifications. The user takes a photo of a target object then mimics the required motion with a smartphone to record the demonstration. The input image and the captured data from the cellphone's IMU sensor then serve to extract three critical pieces of information: *(i)* geometry requirements, *(ii)* mechanisms to obtain the desired mobility, and *(iii)* motion planning. Based on these, Mobiot automatically generates 3D models that fit the target object & reflect mechanism parameters, displays a set of circuitry & driving components needed, and finally produces the code to operate assembled mechanisms with the intended motion. Following the instructions provided by Mobiot, the user can print and assemble all the components and load the program that finally transforms a legacy object into an automotive one. Everyday objects can achieve reasonably complicated maneuvering tasks by combining translation (2D roaming on a horizon), rotation, and lifting motions. Hence, in the context of this work, the word *'mobile'* or *'mobility'* refers to the one or composition of these motions in series, applied to various target objects. Mobiot contributes,

- An enabling toolkit for individual home users to mobilize passive everyday objects at low investment.
- An end-to-end pipeline that enables end-users to build custom robotic IoT devices from one-shot real-world demonstration by automatically capturing design requirements and generating the 3D printable mechanical components, a list of electronics, and the program to robotize the objects.
- End-users can easily re/author the motions upon varying needs without expert knowledge and append, reuse or repurpose fabricated mechanisms with flexibility, to step forward to the future of end-user programming of personal robotics.

2 RELATED WORK

2.1 Interaction with Personal Robotics (HRI)

Computational design of physical entities in the craft culture can be found as early as 1998 [53, 66], while recent advances in robotics and personal fabrication propelled research on the fabrication of personal robotic devices. Interaction with social robots can exert a strong engagement with the users [8], and such interactions are more preferred when the user has control over it [12]. Blossom [57] discussed three design considerations for end-user robotics, *accessibility*: rapid assembly and extension by the user, *flexibility*: customizable by the user, and *expressiveness*: the ability to program without knowledge barrier. We reflect these provisions, the achieved movements are defined and executed by the user-selves through the physical environment on-demand. With 3D printing for accessible and low-cost fabrication, Mobiot incorporates personal fabrication and end-user programming of robotics for *"destruction, remixing, repair, or modification of existing artifacts"* while they are being used [33]. Towards the future that anyone can fabricate their personal projects using accessible and affordable maker technology to adapt to unique needs [6], we aim to actualize "fabrication can happen throughout the entire lifetime of creative work"[33] by empowering end-users to design and fabricate personal robotic IoT devices based on live interactions with their real-world environment.

2.2 Personalizing Smart Systems for Automated Everyday Lives & Accessibility

Supporting users to build personal smart agents can promote the domestication of robotic systems that can bring the future of automated accessible homes to reality. As seen from the prior works, futuristic home can provide personal health assistance [28], support independent living [26], or assist in domestic works [19]. iRobot Roomba, for instance, can aid the daily cleaning routine. Another scope of smart objects is in the area of *accessibility*. Mobile platforms and smart objects can remind about daily routine activities, such as taking medicines or supporting the use of the bathroom by assisting with navigation (e.g., Pearl[49]), fetching diaper/medicine box or laundry basket, adjusting a display height to the wheelchair position, and more.

Many personal robotics and automated systems rely on off-the-shelf platforms [13, 30, 32], which are often unaffordable, hard to program for trivial but varying routine tasks per individual, such as tailoring roaming-paths to deliver a medicine basket from the kitchen to bedroom in varied floor plans. Moreover, such platforms are not flexible in size and shape to adapt to geometric dissimilarities of everyday objects. Mobiot augments objects with custom active mechanisms that adapt a fair range of geometric deviations.

2.3 Motion Planning by Demonstration

For an end-user, one main challenge to automate or actuate everyday environments lies in capturing motion requirements and authoring those actions. V. Ra [13], in this context, provides an AR-based platform for task authoring using a cellphone. Blockly [27], and Vipo [30] present visual programming methods to ease authoring, while Learning from Demonstration (LfD), as explored by Billard et al. [7] and Argall et al. [3], allows to capture demonstration and transfer the extracted information to execute the learned action. Gesture recognition is another considered method of capturing such demonstration [47, 61]. Hand-held devices and wearable sensors have been often used to record demonstration and program the actions of a robotic system, such as Neto et al. [40] utilized accelerometer, Chacko and Kapila [14] used a smartphone with AR, while Ehrenmann et al. [24], Fischer et al. [25], and Dillmann [22] used camera and on-body sensors. Aleotti et al. [1] utilized a 3D tracker to demonstrate tasks in a virtual environment and actuate a robotic manipulator in real life. Similarly, Wang et al. [63] utilized wearable sensors to mimic and reauthor the action of a robotic manipulator in 3D space. In sum, prior works found programming-via-demonstration accessible for end-users, where smartphones could be a solution to aid their demonstrations to capture the desired motion of active objects. We undertake a similar approach, by using a smartphone to enable easy-capturing of the desired motion from demonstration then generate key components for fabrication and programming.

2.4 Reality-based Augmentation of the Physical World by Personal Fabrication

Several prior works have begun to tackle the challenge of end-users to augment everyday objects' functionality using fabrication upon reality-based design constraints. Deriving information from the real-world scenario and synapsing the extracted data to augment physical entities has been practiced by exploiting different tools, such as AR/VR and 3D scanning. RoMA [48] presents an AR-based platform to design and 3D-print artifacts or extend an existing item on-the-fly. Printy [5] embeds circuitry into a 3D model allowing novice users to fabricate functional objects to augment real-world objects. Encore proposes three 3D printed attachments methods on the irregular surfaces of existing, or pre-printed objects [16]. AutoConnect lets users connect two real-world objects, such as a cup and chair in the desired arrangement [34] while retrofitting existing objects to extend its capacity has been explored in several ways with different motivations such as for accessibility purposes, and more [17, 21, 37, 51]. In a similar vein, Reprise allows users to specify, generate, customize, and fit adaptations to everyday objects to enhance usability [17].

More recent works also focused on augmenting objects with active mechanisms to support physical tasks. Robiot [37] and Romeo [36] are the closest related works. However, the domain and purpose of the motions of ours and prior works are non-overlapping. Robiot actuates a part of an object with 1-DOF at a time, mostly a short travel between two fixed points. Yet, the actuation applies to objects that have actuatable or moving parts. For example, an adjustable table lamp can be actuated, whereas a coffee mug with no moving parts cannot be animated. Mobiot, on the other hand, adds navigation and locomotion capability to the whole body. Romeo adds re-configurable capacity into a 3D object, embedding transformable mechanisms to perform localized robotic arm actions, while Mobiot offers to combine both non/localized actions. Also, Romeo relies on a digital CAD tool for path planning that may present real-world conflicts, while we capture real-world demonstration at scale.

3 MOBIOT: AN END-TO-END TOOLKIT TO TRANSFORM EVERYDAY PASSIVE OBJECTS INTO SMART MOBILE OBJECTS

We present Mobiot, a tool for non-expert users to capture design requirements simply from the demonstration of the motion and fabricate, assemble, and code mechanisms to mobilize their passive everyday objects, as overviewed in Figure 2. We set three design goals inspired by the classic HCI design principles [52]:

- **Low Threshold:** Mobiot can enable average users to design mechanisms that can be attached to everyday objects to add mobility without expertise and at a low cost. Mobiot takes the real-world demonstration as a one-shot input then automatically translates it to mechanical entities.
- **Wide wall:** Mobiot can support the creation of mobility mechanisms that fit a wide range of everyday objects. Objects with different sizes, shapes, and weights can be captured for describing mobility requirements and transformed into robotic devices according to the user's demonstration.
- **High ceiling:** The user can create fairly complex motion by combining different degrees of freedom and sequencing actions consisting of three discrete motions at various amalgams. For example, a self-watering bottle can travel in a 2D plane, lift, rotate, and then travel to different execution points. Using several robotic devices together, a user can automate a fairly complex daily task, such as assistive cooking.

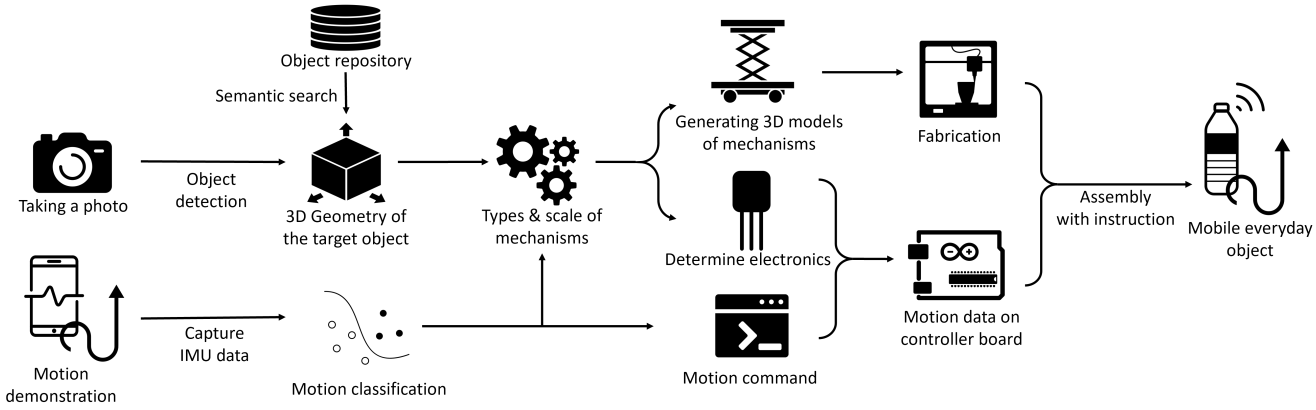

Figure 2: Overview of the Mobiot toolkit. A user demonstrates desired motions of an input target object to get ready-to-3D print mechanisms and controller board components (code and list of electronics) to add mobility to legacy objects

3.1 System Overview

Mobiot links the user's demonstration to the functional 3D-printable mechanisms and a program to actuate the needed electronics. As illustrated in Figure 3, a user starts with *Designing from Scratch* or *Reusing the Existing*. An image of the target object, captured by the user, is used to run a semantic search through a repository to retain the 3D model of it. In the second segment of the input phase, the user runs a provided app on the cellphone to record the demonstration. The user can put the cellphone on the target object while mimicking the large-scale movements or simply move the phone by hand imitating the desired motion. The initial inputs are exploited to attain three information used to generate 3D printable mechanisms and a program to run on electronics: *(i)* geometry requirements, *(ii)* mechanisms to realize the desired mobility, and *(iii)* motion planning information (detailed in Section 4).

The user's demonstration is categorized into three motion classes-translation, rotation, and lift, depending on the prime movements captured in different segments. Users can utilize the recognized motion sequence to compose tasks or fine-tune if necessary, as will be detailed in Section 4.2. Finally, the user gets ready-to-3D-print files of the mechanisms and a code to actualize the composed action. The toolkit also shows the required commercial electronics and purchase links that user can place an order (detailed in Section 4.3). The user then assembles everything to convert the inert object into a smart automated one. We will first introduce various use-case scenarios to create diverse examples in the following section.

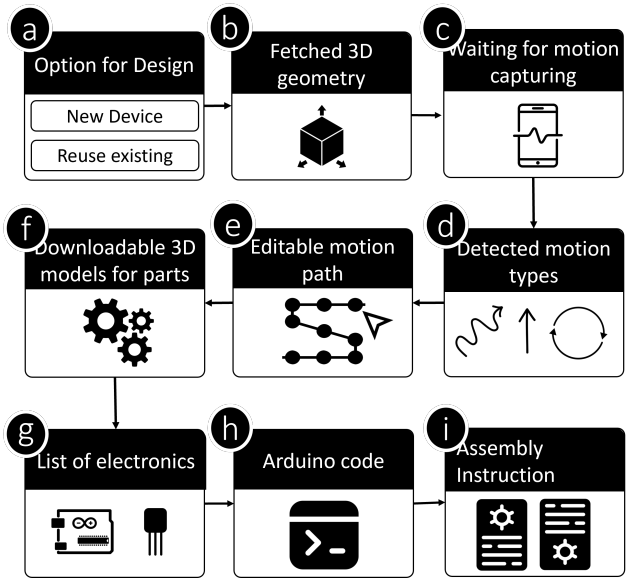

Figure 3: A step-by-step walk through of the Mobiot design toolkit. (a)-(e) requires user inputs for design and (f)-(i) generates outputs for fabrication and assembly.

3.2 User Workflow and Design Scenarios

In this section, we showcase a variety of everyday objects that are transformed to be active in motion to automate the home, following our design pipeline. Figure 3 shows the needed step-by-step walk-through of a user. Here we speculate a story of Nancy at home. Depending on her design scenarios for creating new mechanisms or editing the existing one to re-purpose, she may go through different design routes that streamline the required design steps.

3.2.1 Scenario 1. Creating a New Mobility Mechanism for a New Object: Trash bin. With hands full of food scraps in the kitchen, Nancy wants the trash bin to come near the sink during cooking. Yet, she does not want this to stay forever in the kitchen during usual days as it may be stinky. To make a trash bin that travels and returns to its origin near the door, Nancy first takes a photo of the trash bin with the Mobiot mobile app, which then automatically fetches a 3D model from the repository. Then she puts the cellphone on top of the trash bin, taps *"motion record"*, and drags it to the kitchen for demonstration. Mobiot records the motion using native IMU sensors, and after computing the trajectory, shows the recorded paths on her Mobiot editor. Being satisfied with the trajectory from the screen, she proceeds to the next step of generating outputs. Mobiot produces the 3D printable mechanisms to download parts and a list of electronics that she can order online. She copies the generated program to write to Arduino micro-controller, and the self-organizing trash bin is ready to use, as shown in Figure 8a.

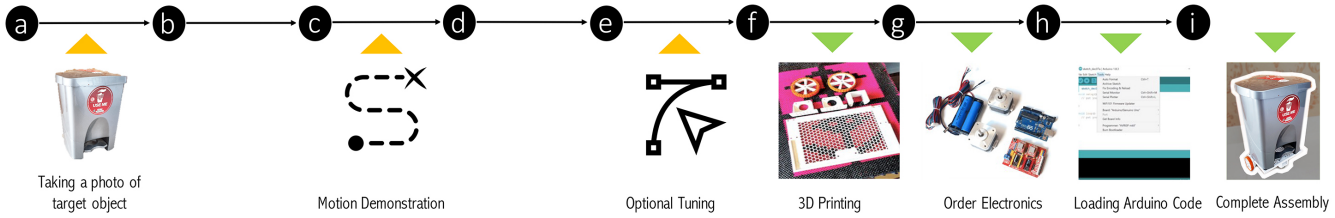

Figure 4: Walk-through to create mobility mechanisms for a trash bin and corresponding user actions. Yellow arrows indicate input and green arrows indicate output.

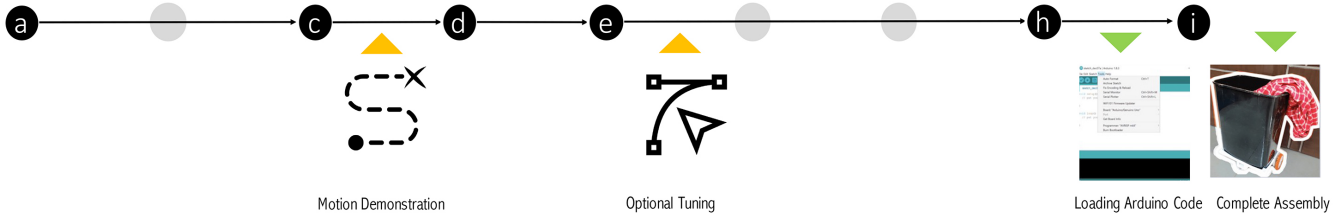

Figure 5: Re-using a the mechanism from trash bin example with new authored motion and new target object for laundry basket, which minimizes input making the design process is much simpler by skipping several stages

3.2.2 Scenario 2. Retargeting an Existing Mechanism with New Recording: Laundry Basket. Liking her self-organizing trash bin, Nancy intends to reuse the mechanism for her laundry basket as they have a similar scale & shape. She wants it to locate near the bathroom on a regular basis, wait to collect used towels, then move to the laundry room when it is loaded. She opens the Mobiot app but chooses the option *"Modify existing mechanism"* this time. She selects the *"Trash Bin"* from the library to load the prior information saved earlier. Nancy records a new motion by placing her phone on top of the basket while moving from door to door. The newly recorded demonstration appears on her editor, from where she slightly adjusts the trajectory by dragging anchors as she needs to offset the curve near the corner where a floor lamp sits. This time she does not need to print anything as there was no change in the 3D model or the type of motion but overwrites the program only, which is generated by Mobiot from the re-recorded motion (Figure 8b). Once loaded, her rolling mechanism now adapts the laundry basket instead and executes the new motion. Note that if the laundry basket is larger than the trash bin, Mobiot would generate a new chassis where the previously fabricated clamps and motor mountings could be attached as this Retargeting process is detailed in sections 4.4.1 and 4.4.2.

3.2.3 Scenario 3. Appending New Motions to Existing Mechanisms: Spice jars. Dreaming of a self-serving robot, Nancy wants spice jars on the table to be self-serving while her guests will be enjoying the Christmas dinner. As Nancy has the 3D model of the jar downloaded earlier, this time she taps the *"Upload"* button and chooses the downloaded STL file to upload it to Mobiot directly. Then, she places the spice jar on the smartphone running the Mobiot app backend, mimicking the desired rotation motion and iterating through the design process, respectively (Figure 6).

After the fabrication, she realizes that the jar needs to be lifted while peppering into a bowl, thus she opens the app to choose the option *"Modify existing mechanism."* Mobiot displays a list of prior mechanisms from where she selects the *"Spice jar."* Then she clicks *"Record a new motion"* to record lifting. The new motion appears in the Mobiot editor, from where she stacks this motion with the previous one. As the physical dimension of the target object remains the same while a new motion type is introduced, Mobiot generates new 3D parts to add lifting and suggests another motor to manipulate it from the motion data, as well as a new Arduino code to reflect changes (Figure 7). After attaching the scissor mechanism below the previously 3D printed attachments, the spice jars are ready to welcome guests, as shown in Figure 8f. The process of appending new mechanisms is detailed in section 4.4.3.

3.2.4 Scenario 4. Creating a Complex Series of Motion: Auto Plant-Watering Bottle. As Nancy plans for a long international trip to attend a friend's wedding, she wants to make an auto-watering system to keep home plants alive while she is gone. The water bottle needs to start its weekly journey from the faucet to fill water, move through several corners of the living room and to the balcony, tilt near a pot to pour water. As the heights and position of the different pots differ, the mechanism also needs to lift and lower the bottle to reach desired spots. Nancy records the series of motions using the Mobiot app running. The motion segments are captured accordingly, enabling Nancy to download all parts that constitute all three mechanisms for roaming, lifting, and tilting, and the code to operate these motions subsequently for the final assembly of the self-watering bottle as shown in Figure 8h. While she is away, she can also use her cellphone to trigger watering in addition to automating it at the scheduled time.

3.2.5 Scenario 5. Self-serving Kitchen with Collective Agents. During busy days, Nancy wants to have her breakfast ready while sitting in front of her laptop checking emails. She dreams of a futuristic kitchen where the liquid egg is to be poured into a bowl (Figure 8e), some spice to be shaken and sprinkled, then the self-mixing bowl to be self-poured to the pan. The breakfast table will serve

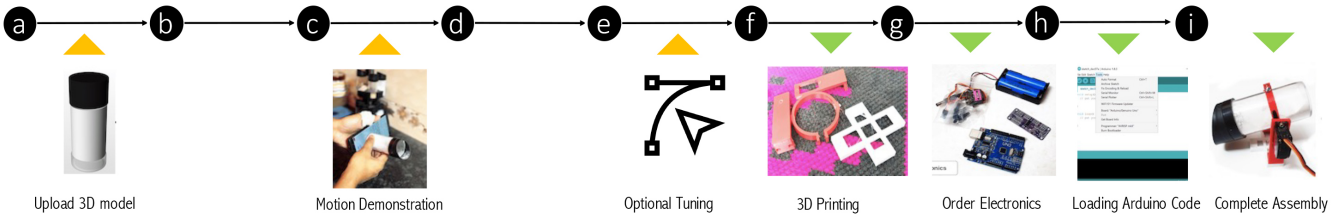

Figure 6: Creating a tilt motion for a spice can

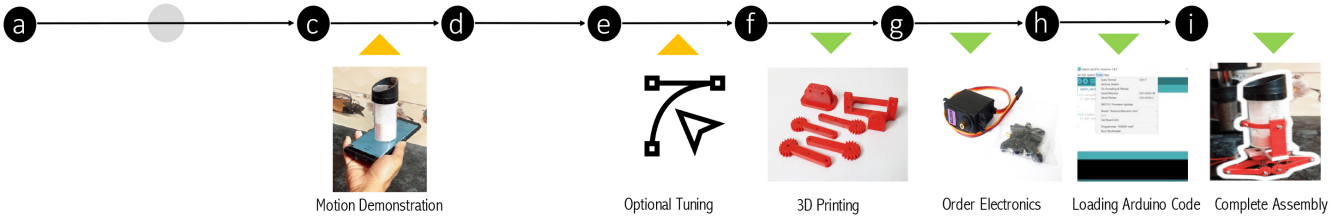

Figure 7: Appending a new motion (lift motion) to an existing mechanism that is added to a spice can

condiments and a milk carton and contain a self-organizing coffee cup (Figure 8c) and a platform to take her bread or salad bucket with self-served dressing on it. She records motions for each of the items, uploads them to Mobiot, collectively creating a smart cooking assistance that yields the 3D printable parts, a list of electronics, and the code for individual items (See section 4.2). She then 3D prints and assembles all the parts and attaches them to each target.

In the following section, we detail how the user inputs, in accordance with the possible use-case scenarios, are extracted to retrieve needed information to generate 3D printable mechanisms, the list of electronics, and the program to run them.

4 IMPLEMENTATION

Mobiot involves several steps for adding mobility to everyday objects, for example, extracting features to generate 3D printable parts using parametric modeling and programs.

4.1 Capturing Requirements by Demonstration

4.1.1 Geometry Information. To create 3D printable mechanisms that fit the target object, it is crucial to obtain geometry information for appropriate parametric design. Mobiot design tool contains a repository of 3D models, mainly derived from Shapenet [15]. When the user takes and uploads a photo of the target object, Mobiot passes the image through YoloV4 [9] object detection model. From the detected object class, the tool makes a semantic search in the repository and fetches a corresponding 3D model (Figure 3b). Optionally, a user can grab a 3D model of the target object from any public repository and provide it as input directly. As such open repositories are actively being populated with 3D models of many real-world objects, it will soon become a better approach to guarantee the accuracy of obtained 3D geometry.

4.1.2 Motion Information Classification. A user's motion demonstration is captured using the native IMU sensor of a smartphone, consisting of accelerometer and gyroscope, through a sensor data

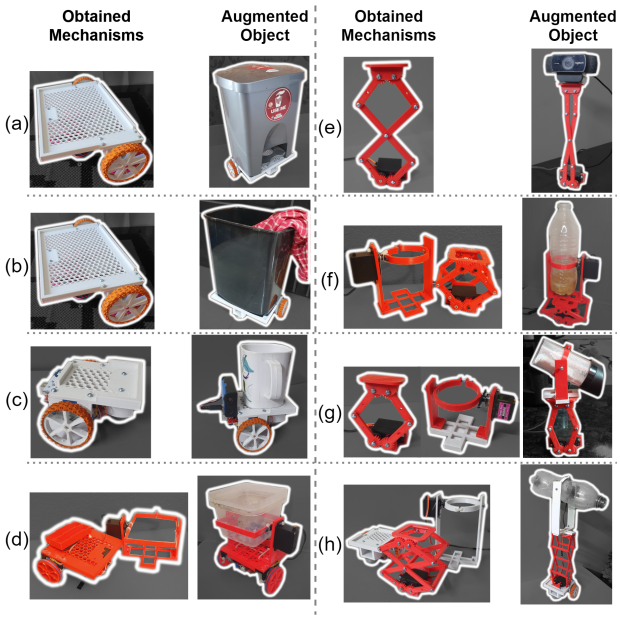

Figure 8: Examples generated by Mobiot: (a) Mobile trash bin, (b) Mobile laundry basket re-using the mechanisms from the trash bin, (c) Self-organizing coffee mug, (d) Assistive cooking bucket, (e) Adjustable webcam stand, (f) Self-serving beaten-egg bottle, (g) Self-shaking spice jar, (h) Auto plant-watering bottle

logger application on the Android platform [56]. The app is modified to capture the IMU data at a maximum possible refresh rate that typically varies from 100Hz to 500Hz depending on the sensor type. The required data types are gyro, accelerometer, linear acceleration, gravity, magnetic field, rotation vector, and timestamp. A user can upload the recorded motion data stored in the cellphone directly to the Mobiot by using the app or the user interface that will become the initial input for the mobility analysis (Figure 3c).

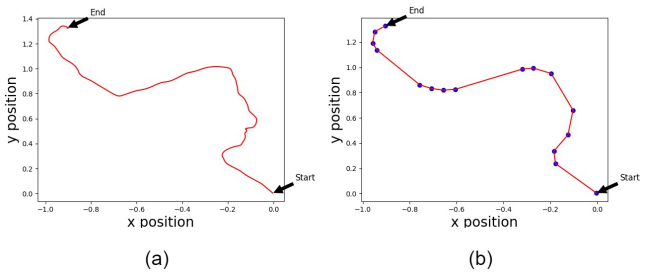

Figure 9: Processing of horizontal motion data (a) Raw positional data obtained using RONIN [29] (b) Moving average filtering and finding the anchor points.

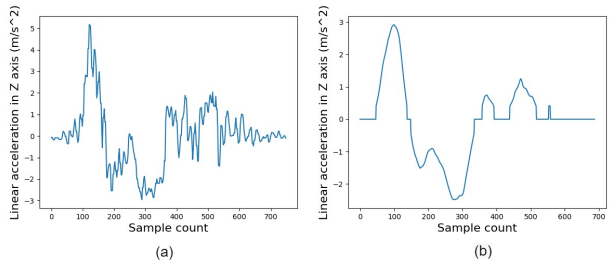

Figure 10: Segment of the motion data (a) Raw linear acceleration data along Z-axis (b) Filtering and thresholding.

The next step is to classify and quantify the demonstration (Figure 3d). Currently, Mobiot breaks down the demonstration into three most common motion types- translation, rotation, and lifting, by searching the axis where the most salient movement occurs, which is detailed as follows.

Translation. The translation motion is a 2D movement over a horizontal surface. We estimate the 2D trajectory using RONIN [29], a machine learning-based approach for dead reckoning. After calculating the total distance traversed and comparing this value with a threshold, the requirement for a horizontal motion is determined.

The raw trajectory ($\Gamma^r_{(x,y)}$) of the horizontal motion contains noises (Figure 9a) due to sensor accuracy and user's wavy hands. The first step of processing this trajectory is passing the data through a low-pass moving average filter (Figure 9b) as per equation 1 to obtain a smooth trajectory ($\Gamma^f_{(x,y)}$).

$$\Gamma^f_{(x,y)}[k] = \frac{1}{M}\sum_{i=0}^{M-1}\Gamma^r_{(x,y)}[k-i] \quad (1)$$

For our application, we consider window size (M) to be 5% of the trajectory data length. The second step of processing the trajectory data is fitting piece-wise linear segments as they can provide anchor points to allow fine-tuning. We consider a distance and tangential angle threshold for finding the anchor points from the smoothed trajectory ($\Gamma^f_{(x,y)}$). If any two consecutive line segments fall under this criteria, the corresponding points are considered to be anchor points as shown in Figure 9b. The obtained trajectory with the anchor points is then presented in the interface (e.g., Figure11d).

Lifting. Estimation of the lift motion from the demonstration is subjected to additional computation. We first grab the linear acceleration data along the Z-axis as shown in Figure 10a. It is obtained from Android's sensor fusion algorithm that removes the gravitational component from the acceleration data. This data is then passed through a moving average filter and thresholding (See Figure 10b). Afterward, we perform a double integration of the filtered data to estimate the amount of lifting height. Finally, we use a binary threshold to detect whether and where the lift motion was triggered. The amount of lifting height is presented as motion blocks in the user interface (See Figure 11b).

Rotation. From the IMU sensor, we obtain the rotation vector at a given time to estimate the roll or pitch of the cellphone. The rotation vector represents the cellphone's orientation about x, y, and z axes in the form of quaternions that we convert to Euler angles. If the estimated roll or pitch exceeds a threshold angle, it is classified as a rotation motion and presented as motion blocks in the user interface.

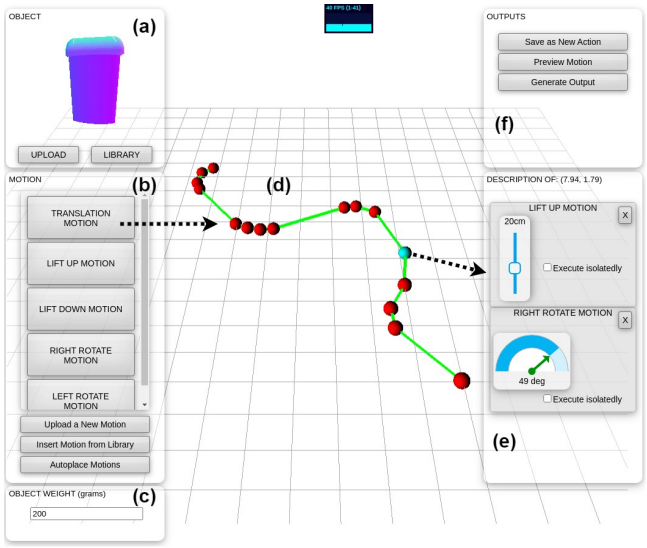

Figure 11: Mobiot interface consists of (a) fetched 3D model, (b) detected Motions' list as library, (c) additional information (e.g. weight), (d) task design and preview, (e) appended motions and parameter tuning and (f) options for output.

4.2 User Interaction and Task Design

After the classification and quantification of the demonstration, such as automatically segmenting the anchor points of the horizontal motion, this information is represented as different motion blocks in the interface (See Figure 11b). At this point (Figure 3e), the user can (i) import different motion blocks to the scene and manually design complex tasks, (ii) proceed with the motion sequence automatically recognized, or have a combination of both.

4.2.1 Manual Task Designing. The user can manually design a task from motion blocks, which represent individual motion types found in the user's demonstrations. Once a block is imported to the scene, the user can append other motion blocks if needed. For instance, the user can import a translation block into the scene, which displays anchor points of the trajectory to the user (e.g.,

(Figure 11d). Afterward, the user can tune anchor points if needed or add new anchor points to the trajectory. Then, other motion blocks such as lift or rotation can be appended to different anchor points (Figure 11d and e). The user can tune the amount of lift or rotation appended to each anchor point and drag-drop to organize their sequence, add pause or delay in between different actions, and choose to execute them either respectively or synchronously.

4.2.2 Automated Motion Sequence. During the motion classification, Mobiot keeps track of the spatial information of different motions. If the user author motions with the automatic recognition of sequence, the editor arranges motion blocks on the scene as they appear in the demonstration. This sequence may contain errors and noise generated by the natural movements of the body, requiring additional tuning. Once the sequence is auto-placed in the task design scene, Mobiot allows the user to optionally fine-tune parameters, add more motions blocks to different points, re-arrange them, remove any unwanted motion recognized by the process. The automatically imported blocks are by default executed sequentially, but can be also executed synchronously by selecting targets.

4.2.3 Action Module Designing. In our daily activities, many tasks consist of modular actions, such as mixing, pouring, etc. Design of such is enabled by integrating multiple low-level motions. Using Mobiot's interface, a user can arrange different motions in series to obtain a particular action and save it to the library. For instance, the user can sequence alternating clockwise and counterclockwise rotational motion to have a mixing action. Some other examples can be *synchronous lift + rotate = pouring, or lift then rotate multiple times = shaking*. Once saved as modular action, a user can re-use them and adjust parameters to modify their actions. In the future, these high-level action descriptions can also be brought under a repository to ease the task designing.

4.2.4 Additional Information about Target Object. The scale of the 3D printed mechanisms and electronics might vary upon the weight of a real-world object, as the heavier will need stronger motors. Mobiot asks a user to input the approximate weight of the object (Figure 11c) as the torque is computed based on it. Additionally, the weight also affects the parametric adjustment of 3D models, such as defining the size of motor mountings, the number of gears in the lifting mechanism, the thickness of the clamps.

4.3 Generating Outputs

The geometry of the object and the motion information are finally mapped to generate the outputs (Figure 3f to i). Mobiot provides the user with four outputs: (i) 3D printable parts of the mechanisms, (ii) a list of electronics, (iii) Arduino code to program the device, and (iv) assembly instructions.

4.3.1 Generating 3D Models. At this point, Mobiot acquires sufficient features to generate the 3D models of the mechanisms. The target object may be equipped with one or more mechanisms to replicate the desired mobility from the user demonstration. Mobiot uses a parametric design approach based on three main design libraries to reflect the acquired parameters into the 3D models. Figure 12 depicts three example parametric designs to illustrate the motion mechanisms and driving components.

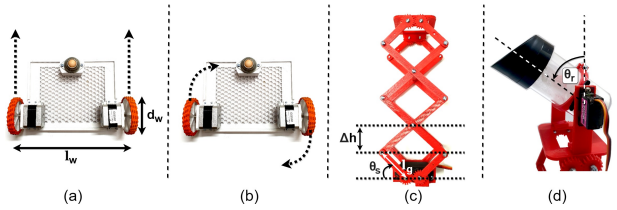

Figure 12: Realizing motions with three parametric designs. (a) Forward using the differential drive, (b) Taking a turn using the differential drive, (c) Adjusting the height of the lifting mechanism, (d) Rotating to a particular angle.

Differential Drive Mechanism. The translation motion is realized by a differential drive mechanism (Figure 12a). The geometry of the target object determines the size of the chassis, and the weight determines the required torque for mobilizing the platform. The torque is given by $\tau = r * (m/2) * g$ where m is the mass of the object, g is the gravitational acceleration, and r is the radius of the wheel. We match the required torque against a database built from the information acquired from the motors' dataset. Mobiot chooses motors with 20% more torque than required to incorporate other items within the mechanism. Afterward, supporting elements (e.g., motor clamps) are generated using Mobiot's parametric templates.

Scissor Lift Mechanism. The lifting mechanism is a modified version of the scissor lift system (figure 12c). Counter gears, driven by servo motor(s), are used for the scissor action. Additionally, parametric linkers are used to achieve the desired height. The selection between a single-sided or double-sided lifting mechanism and the strength of the linkers are determined based on the object's weight.

Rotation Mechanism. The rotation mechanism is also operated by a servo motor. A clamp holds the object while two supporting poles over a base (Figure 12d). The torque for rotating the object is calculated in a similar manner as described earlier. The clamping point is considered to be the center of gravity of the object, estimated from the object's 3D model. We borrowed Encore[16] to define clamps by circumference of a cross-section of the object. Once all the models are generated, ready-to-print STL files are created for downloading and 3D printing (figure 3f).

4.3.2 Electronics and Driving Components: After generating the 3D models, Mobiot rations required motors and controllers and presents the user with a list of items along with their store links (figure 3g). We use Arduino Uno, one of the most common open-source hardware, as the main controller. For the differential drive, we utilize stepper motors- NEMA17 (for mid and high torque applications) and 28BYJ (for small and lightweight operations). We choose the open-source Arduino-CNC shield, which is stackable to the Arduino board, to drive the steppers. For rotation and lifting mechanisms, we use servo motors (MG90, SG5010, and MG996R by Tower Pro) from low to high torque. For WiFi connectivity, we use a low-cost ESP8266 IoT-enabled module.

4.3.3 Transferring the Motion Information via Auto-Programming. As the 3D models are generated based on parametric designs, it covers all the critical information to process the motion data. Motion data is first parsed into motion commands then embedded into an Arduino Sketch code.

Parsing the anchor points. We iterate through all anchor points and calculate the distances and tangents from point to point. The number of motor rotations required to traverse a certain linear distance depends on the wheel diameter (d_w) as in Figure 12a, while for taking any turn, the distance between the wheels (l_w) is needed (See Figure 12b). For simplicity, we neglect the frictional coefficients and wheel slippage. Considering d_w, l_w, and steps per revolution of the motors, Mobiot calculates and sequences of steps required to proceed with linear distance or taking any turn to go from one anchor point to another.

Achieving the lifting height. The lifting mechanism comprises gear and linker stages as depicted in Figure 12c. The total lifting height (h_l) consists of small step heights (Δh). Considering the linker length (l_g) and adjusting the angle ($\theta_s = arcsin(\Delta h / l_g)$) of the servo, the desired lifting is achieved. This angle value is then parsed into a command string.

Executing the rotation motion. As servo motors can parse a given rotation angle (θ_r), it can be directly appended to the command. Mobiot then integrates the individual commands by encoding them with identifier keywords- go (g), turn (t), rotate (r), lift (l), delay (d), etc., to form the final command string. Mobiot patches this to a prebuilt code template that contains necessary functions to drive the motors. This can be copied and flashed directly to the Arduino board (Figure 3h) using the Arduino IDE. After programming and assembling the mechanism by following the provided instructions, the user needs to place it to the location where the demonstration was taken. Once a motion execution is done, the mechanism can be triggered to retain its original states, such as rendering the translation motion in reverse order to go back to the origin or adjusting the servo angle to reset the vertical position.

4.4 Reusability of Produced Mechanisms

After adding mobility to an object, Mobiot registers it to the library. Exploring the library, a user can alter the functionality of an existing mechanism to adapt it to different use-cases. Mobiot offers three ways of re-utilizing an existing mechanism- (i) reusing an existing mechanism with different motion settings, (ii) retrofitting a different object to an existing mechanism, and (iii) stacking new mechanisms to the existing.

4.4.1 Re-authoring Motions. To reauthor an existing mechanism, a user can record a new demonstration using Mobiot to convert it to machine commands. A laundry basket which initially designed to travel from the bathroom to the washing room can now roam from the bedroom in the early morning to the kitchen during breakfast to collect table clothes (e.g., Figure 5).

4.4.2 Retrofitting. Retrofitting lets a user attach a new object to an existing mechanism by replacing some parts of the mechanism. The process starts by capturing the geometry information of a new object that Mobiot offsets geometric changes from the previously stored information. If the new object is larger or smaller than the previous one, Mobiot generates new attachments or chassis corresponding to the changes. For example, if a rotation mechanism designed for a bottle is to be replaced with a spice jar in a shorter diameter, Mobiot will only generate a new clamp and base, assuming the weight does not significantly deviate from the earlier target.

4.4.3 Appending New Mechanism. As demonstrated in the earlier spice can example, a user can append new mechanisms to the existing one. Additional mechanisms are determined from a new demonstration by comparing it to stored information. The user can achieve this through two approaches, either by recording a full demonstration, or by recording the segment that needs to be appended. After classifying the motion types, Mobiot presents the user with new motion blocks, thus the user can design a new task or proceed with the automatically determined sequence. However, the stored motion information remains temporarily in the motion list until the user proceeds with the output phase. If a user choose to go with new motion recording only, he appends it to the previously stored motions manually through the user interface. As Mobiot sees the requirement for supporting a new motion, it provides additional 3D models, generates new motion commands from the new data, and lists additional electronics if added more.

4.5 Schematics and Constraints

In this section, we discuss configurations, schematics, and impossible cases by reviewing the scenarios introduced in Section 3.2.

4.5.1 Speed settings for the platforms (Scenario 1). In our test experiments, we observed a trade-off between the speed and torque of the stepper motors. To compensate variables for novices and ensure proper motion execution by the actuators, we use a predefined speed-set for motors. Thus, in scenario 1, changing the speed of the trash bin through the user interface is currently not possible. Presently, users have an opportunity to add latency or delay between different motion executions. Users with basic knowledge of Arduino sketch programming can change speed parameters by following the comments in the provided code.

4.5.2 Authoring new motions to retarget (Scenario 2). Mobiot interface offers two prime configurations of (i) adding new motions and (ii) modifying existing mechanisms to reuse, re-author, or retrofit. Currently, the interface does not allow appending multiple translation motion demonstrations. In this case, the user will need to record a whole new demonstration instead. Once a saved mechanism is updated with a new demonstration and generate new outputs, it replaces the prior information. While retrofitting a platform, the weight need to be similar. For example, in scenario 2 of section 3.2, if the weight of the laundry basket differed significantly from the trash bin, the desired motion would not be executed successfully.

4.5.3 Stacking mechanisms to append new motions (Scenario 3). The generated mechanisms can operate in combination or standalone. However, our current assembly hierarchy does not allow (i) appending a lifting mechanism on the top of a rotation mechanism due to the limitation of supported motion and authoring types, and (ii) the translation motion mechanism on the top of other mechanisms as it is not practical. Also, the same type of mechanism is not currently supported to be stacked. It leads to a total of 7 configurations or combinations for the mechanisms. Thus, in scenario 3, if Nancy stacked the lifting mechanism on the top of the rotation mechanism, it would not be possible to execute the intended motion.

4.5.4 Empty object vs. loaded object (Scenario 4). While entering the additional parameter- weight, the user needs to input the maximum possible weight of the object. For example, in scenario 4, entering the weight of the empty bottle in the interface may lead to malfunction or topple the mechanism when the motion is executed in real-life with a bottle full of water.

4.5.5 Availability of 3D models (Scenario 5). Currently, we fetch a 3D model from the 2D image from a pre-built repository built from the Shapenet [15] models mostly. Thus, the number of models is limited in our benchmark. Additionally, objects with the same taxonomy may vary in size and shape. Enriching the number of models in the repository and adding more classes to the machine learning model can resolve the issues. Alternately, users can download 3D models of the target object from any repository and feed them into the tool. Besides, users can use substitute models that are coherent in their geometry requirements. For example, if Nancy took a photo of a mayonnaise bottle in scenario 5 that does not exist in this benchmark, she could use the 3D model of a water bottle instead as they share identical shapes and sizes. Future solution to this constrain is discussed in section 6.3.

4.6 Limitations in Hardware Generation.

4.6.1 The relative scales and attachment mechanisms. Although the scale of the mechanisms is flexible depending on the target object's physical dimension, the minimal size is limited by motors and electronics to be assembled. The augmentation of the trash bin or laundry bucket seems natural, while attachments appear relatively bulkier for a spice jar or other small objects. With tiny commercial driving actuators such as Toio become more accessible, addressing the scale limitation could be a near-future work of Mobiot. Additionally, shells for electronics can be inherited from Retrofab [51] or Desai et al. [20]. To tightly hold the object on a platform, we use the clamps generated as in section 4.3, and a boundary across the translation chassis. For toppled or non-uniform objects, AutoConnect [34] could enhance the capacity.

4.6.2 Height accuracy retained from acceleration. In practice, step counting [44, 45] and double integration [38, 41] are popular methods of dead reckoning and trajectory estimation from IMU sensor. RIDI [67] shows a higher benchmark in the context of the double integration method, yet, it can not fully compensate for the accumulative error in double integration. Step-counting based positioning systems do not fit our application. Android Tango devices have an inbuilt positioning system, but such devices are not ubiquitous, which made us settle down to RONIN. However, it provides the 2D position of the audience with a non-tango device. As a result, the lifting height cannot be determined precisely with the existing methods. As a workaround, we use the linear acceleration data to make a rough estimation of the lifting height via the double integration method, which suffers from accumulative errors.

4.7 Design & Authoring Tool

We used Django, a Python-based web server for the backend. We developed the front-end as an web application using javascript, integrating Three.js for visualizing and rendering scenes and 3D models. OpenSCAD is used for parametric 3D designs.

5 EVALUATION

5.1 Technical Validation

Throughout the technical validation, we investigate how accurately the tool can reflect the demonstration, origin of different issues or inaccuracies, and how they can impact the usability of the mechanisms. Mobiot relies on three salient movements and makes different combinations to achieve various actions. Hence, we evaluate the involved cardinal motions: translation, rotation, and lifting, as we can segment a combined motion into these three categories.

Apparatus, Tasks, and Settings. (i) The trajectory estimation of the translation motion relies on RONIN[29], and hence we are focused on evaluating the parsing of the trajectory by the mechanisms. To find the accuracy of the parsed translation motion, we use a marker to draw the ground truth with known distances and angles and equip a translation motion mechanism with a marker (Figure 13a). Afterward, we upload the trajectory data manually to the Arduino board and let the mechanism parse the command data. The first type of horizontal traversal covered 200cm and two right-angle turns. Additionally, we let the mechanism traverse 400cm straight

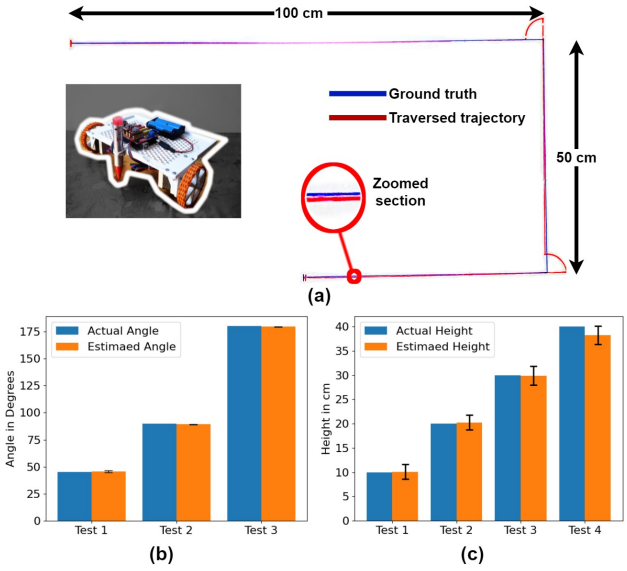

Figure 13: (a) Test setup to evaluate translation motion accuracy, ground truth and traversed trajectory (b) estimated angle compared to the actual angle for rotation and (c) estimated height compared to the actual height for lift motion.

with a rated load for the motors. For each criterion, we consider five iterations with two additional configurations- (1) repositioning to the origin before a trial and (2) letting the mechanism return to the origin on its own.

(ii) The rotation and lifting mechanisms use servos that parse the given angle with their feedback system, and hence, in these cases, we focus on evaluating the estimated parameters. For these criteria, we move a cellphone between two known points, record the motion, and let Mobiot quantify the demonstration. We recorded the motion of a cellphone by rotating it to known angles (45°, 90°, 180°) with ten trials for each angle. And, we lift the cellphone to

known heights (10cm, 20cm, 30cm, and 40cm) and recorded the motions of ten iterations for each height.

Results. Throughout five iterations of 200cm traversal, the mechanism lagged by 6mm to 9mm from the endpoint with a standard deviation of 1.36. With the object loaded, the displacement of the mechanism was found from 2cm to 8cm with a standard deviation of 2.24. When the mechanism was not re-positioned to the origin, the deviation accumulated. As a result, the platform can completely deviate from its origin point after multiple uses. These deviations originate from several points. One critical reason is the wheel-surface slippage. Currently, we use TPU or rubber-based tires for wheel-surface contact. But, as the controller estimates the traversed distance based on the step count and wheel geometry and the stepper motors do not have odometric feedback, the slippage can not be fully compensated. Besides, there is a tolerance limit of how accurate the 3D printer can print the models, such as wheels. Even a slight deviation in the diameter can result in larger dislodgement with long traversal. Hence, the platform needs to be re-positioned to its origin after several uses. We discuss this further with possible solutions in section 6.6.

Figure 13b compares the estimated angles (roll or pitch) of the recorded motion to the ground truths. The standard deviations for 45°, 90°, and 180° are found as 0.83, 0.20, and 0.21 with mean values 45.7°, 89.25°, and 179.53° respectively. Cellphone's sensor fusion algorithm provides highly accurate rotation vector data. In daily life activities, movements are more empirical than precise (e.g., shaking a spice jar), and such slight deviations are barely perceivable to have any substantive impact on usability.

As the accelerometer data contains heavy noise, integrating twice lets the noise accumulate and ultimately leads to a high deviation. As illustrated in Figure 13c, for 10cm, 20cm, 30cm, and 40cm heights, we observed standard deviations 1.52, 1.53, 1.94, and 1.87 with mean values 10.1cm, 20.24cm, 29.89cm, and 38.22cm. Such deviations can impact the usability of the platforms for subtle applications, such as the water bottle may over-spill water, while it can be relatively ignorable for serving foods where human intervention can compromise. Currently, the user interface provides the opportunity to tune the motion parameters, and hence, the user can compensate for the issue through trials. As commercial smartphones are getting equipped with better positioning sensors for AR and VR (e.g., Android Tango devices), we expect to have better benchmarks in 3D positioning in the future.

5.2 Experts Interview

We interviewed three experts to elicit their feedback, recruited from the first author's network, with four years of experience in fabrication (E1), four years of experience in the field of human-cognition analysis (E2), and eight years of experience in software & machine learning (E3). We introduced the toolkit overview with one example demonstration using the fully fabricated cooking bucket (Figure 8d). Interviews with E1-2 took place in person, and we let them use the user interface, whereas the interview of E3 took place online to whom we showed the demonstration of the user interface. Overall, all three experts appreciated the features and user-centered workflows, noting *"If the end-user only has to just connect, I think that's a pretty good approach. If you can assemble the furniture, you can also do this" (E1), "Like movies, [...] in a room there's magic happening everywhere, everything is coming to your hand. These things are not happening now, so we should help or we should be participating in those things so that, that future can be achieved faster" (E1).* All also revealed limitations and suggestions for future improvements towards their future visions for ubiquitous personal robotics as we summarize the key findings:

Cellphone as a Design and Authoring tool. All experts liked the use of the cellphone as a design and authoring tool, as *"I think it would be the easiest because we use cell phones every day" (E1).* As they all are aware, modern smartphones have a decent amount of sensors that can be utilized to capture various activity data just by one-shot demonstration, while they are not necessarily required to know all details about the types of sensors or data types they need to acquire for fabrication. With the ongoing development of the smartwatch and other unobtrusive smart wearables, we believe that the user's embodiment can also be a potential design input.

Fabrication cost. E2 expressed concerns about Mobiot's feasibility regarding the fabrication cost, thinking of the overarching goal of lowering the access of personal robotics fabrication. *"Is it also considering the cost of that product and suggesting?" (E2).* As the experts referred, the fabrication cost is one of the critical factors while developing the mechanisms, that we lower the cost by leveraging consumer-grade 3D printing as the main method to create the mechanisms. While the cost is subjective to the valuation of functionality and personal preference, we aim to portray the fabrication cost along with the bill of materials in the future user interface to let the user judge the cost.

Sensor integration. Experts also pointed to the necessity of sensor integration, as augmented objects using Mobiot will be operating in the real-life context where there could be many changes in the daily context, such as new obstacles placed in the pre-planned path or temporary obstructions in the trajectory. *"One of the major things about any human-robot interaction is safety. So you need something, like, some kind of safety sensors" (E2). "It would be nice if the system is smarter to get signals from my room. Like, the obstacles that you are going to avoid" (E3).* While the sensor integration can significantly improvise the functionality of the mechanisms, it also poses new design requirements to locate sensors without hampering the functionality of prior electronics. We further discuss the future integration plan for sensors in detail in section 6.4.

Validation and Simulation. Experts were curious about motion contextualization in real-world and validation, speculating the possibility and feasibility of simulation. *"Is there any verification process if the things will actually work or not?" (E2).* While we simulate the motion in the user interface, E3 suggested integrating floor plan or 3D model of the arena, *"You can import the map of the room into the software so that I can visually see the movements, like creating a 3D model of the room" (E3).* We also present a further discussion on this in section 6.5.

6 LIMITATION & DISCUSSION

6.1 Personal Robotics and Mobiot

While we observe the adoption of robots in industries and manufacturing, home with smart agents is still in speculation. Two important questions are to be addressed: *"what might a home robot do?" and "what would it look like?*[42]". As observed in autonomous

lawnmowers or vacuum cleaners, one core functionality of everyday robots is to assist in daily repetitive tasks making users focus on other productive activities. Instead of reckoning sophisticated humanoid robots or Androids that will magically carry out all of our works, we can visualize the future of everyday robots as tangible devices delegating routine tasks. While users are empowered with adding actuation (e.g., Robiot[37]), manipulative capacity to 3D printable objects (e.g., Romeo[36]), or control to mechanical buttons (e.g., IoTIZER[18]), Mobiot juxtaposes the addition of mobility. That being said, an integrated system soon will unfold the story of everyday personal robots being designed and fabricated by the end-users.

6.2 DIY Robotics and Mobiot

The discourse analysis of Roedl et al. [53] depicts a recent shift in HCI research, where the paradigm of end-user system design is increasingly focusing on *makers* rather than passive users. Wakkary and Maestri [62] portrays how family members as everyday designers augment, adapt, and modify their surroundings, which ground the motivation of supporting end-users with tools and technology that is also relayed by Buechley et al. [11] and Kuznetsov and Paulos [35]. From trusting the user's capability to design, assemble or repurpose their furniture, we are marching towards co-design and fabrication of custom products as already witnessed in DIY headphones [50], co-manufactured mechanical devices[46], custom assistive design of IKEA furniture [60], and more. The process of augmentation and customization adds personal value, attachment, and individual meaning to the items that elongate the duration of adoption and urges persistent care over time [43]. As a result, the cultivation of these acts contributes to the betterment of social and environmental aspects as well. Being inspired by these findings and gradually filling the blanks of users' demand for supporting various physical tasks, we also expect new and existing repositories (e.g., Thingiverse) to encompass everyday robots for mass users. Instead of being a *consumer waste* of the past, we envision today's everyday legacy objects being transformed into everyday robots of the future.

6.3 Automatically Recovering 3D Model Geometry from 2D Photo Input

In case the semantic search of the target object fails, we preliminarily integrated photogrammetry techniques to generate the 3D model of the target object. Works such as Pixel2Mesh++ [64], 3D model generation using GAN [68] provide prospective ways of generating 3D model of an object from 2D images. However, 3D models generated from such methods may not be precise and contain noises to create perfectly fitted 3D printable mechanisms. Besides, there are everyday objects that share identical shapes. Such as, the 2D geometrical mask of a water glass can be identical to the mask of a laundry basket. Thus, as a current solution, we utilize a machine learning technique instead to identify the object and run a semantic search for fetching the 3D model of the target object. Cellphones with lidar sensors (e.g., iPhone 12 Pro) prophesize a handy 3D scanning. We expect to re-integrate the automatic recovery of 3D mesh when advanced methods with higher benchmarks are available.

6.4 Sensing to Adapt to Dynamic Changes

Sensing enables users to add custom interactions to different interfaces. IoT-based sensors can allow remote sensing and action mappings, such as automatic lights and fans [55], smart doors [23], etc. Commercial IoT devices, such as Google Nest, allow users to control inter-connected home appliances via voice commands. LeapMotion or Kinect enables gesture sensing for custom action mapping. Platforms, such as Blynk or Adafruit IO, let novices add IoT control or sensing using opensource hardware. Currently, we utilize an IoT-based WiFi module (ESP8266) and a web interface for triggering the actions. Our choice of open-source hardware and WiFi module leaves the floor to the user for IoT-based sensors integration and action mapping. We assume users can eventually adopt these commercial sensing and triggering solutions. Considering such, sensing and custom action mapping is left out of the scope of this work. The inclusions of onboard sensors, such as LiDAR will make the mechanisms smarter, as they can facilitate safety and obstacle sensing. A native camera can facilitate visual monitoring, while smoke or CO sensor can provide an additional layer of safety and feedback. Moreover, recent 3D printable sensing systems (e.g., [54, 59, 65]) could be considered to embed such sensors within the mechanisms in future works.

6.5 Motion Contextualization

As also recommended by one expert, we plan to embed 3D scanned floor plans of the target areas that are increasingly easy to integrate with advances of native sensors in modern smartphones. It will help visualize the trajectories in a real-life context providing a more accessible design scenario to the user. The recent inclusion of lidar sensors in cellphones and AR-based apps to measure the area (e.g., Apple AR Ruler) pave the way for 3D scanning of a scene and generating the 3D floor plan of an arena. Simulation of the motion in a virtual/augmented reality environment can also be enabled in such a scenario so that end-users can validate the outputs before fabrication. Such integration can facilitate other accessibility applications, such as guided navigation for elderly care to navigate different places (e.g., bathroom, kitchen, etc.) from remote places by caretakers as well.

6.6 Limitations in Motion

Types & Authoring. Mobiot currently incorporates three types of motions. Actions of everyday objects may require more motion types, such as twisting, curved lifting, sliding, etc. Mobiot breaks down the demonstration into pieces, based on the salient motions observed in different axes. For example, a curved lifting will be reported as a lift motion and a separate rotation motion, thus, fitting them into *realizable* types. Additionally, any accidental motion falling under the conditions described in section 4.1.2 will be present in the interface. However, the user has the opportunity to remove or tune any unwanted motion, or re-record the motion otherwise. We aim to stretch the type of supported motions. Also, action triggering in compound motions (e.g., translation and rotation together) is accomplished only using spatial programming that can be extended to sensor or time-based triggering to allow inter-agent collaboration. As the triggering methods are out of the scope of this work, we leave this as a future improvement.

Initial placement & Deviation in Translation motion. As discussed earlier in section 5.1, the translation mechanism needs to be repositioned to the origin after multiple uses due to possible deviations over time. The frequency of repositioning a platform depends on several factors, such as the amount of distance traversed by it, its size, and path clearance. With a prolonged trip by the platform, the deviation accumulates higher. Adequate path clearance can allow the platforms to be in action even with some degrees of divergence. The deviation is also subjective to the platform size, such as a 5cm deviation for a mobile trash bin is less apprehended compared to a mobile coffee mug. Without any feedback sensors, the mechanisms are not self-aware of their positions and use the starting point as the reference. Thus, changing the initial orientation of the mechanism can lead to a wrong trajectory. Additionally, the mechanisms require re-authoring if they are relocated to a different place. In the future, an onboard IMU sensor can help recognize the deflection from the position, while the addition of positioning sensors and odometric feedback can extend Mobiot's capability in the wild.

7 CONCLUSION

Personal robotics is receiving concentrated focus by researchers from several domains. In addition to actuation and sensing, mobility is another critical element for domestic robotic gadgets. Yet, unlike 3D-printing passive artifacts, the process of adding mobility through personal fabrication is convoluted as it demands domain-specific expertise. Mobiot advocates a barrier-free tool for end-users to augment everyday objects with mobility through personal fabrication. It captures design requirements from the physical world and allows users to mobilize everyday objects without expert knowledge. We believe Mobiot can add one more step towards democratizing robotics for end-users. Mass participation of end-users in personal robotics can accelerate the arrival of the future smart home.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable feedback. We also thank the experts for their participation.

REFERENCES

[1] Jacopo Aleotti, Stefano Caselli, and Monica Reggiani. 2004. Leveraging on a virtual environment for robot programming by demonstration. *Robotics and Autonomous Systems* 47, 2-3 (2004), 153–161.

[2] Ron Alterovitz, Sven Koenig, and Maxim Likhachev. 2016. Robot planning in the real world: Research challenges and opportunities. *Ai Magazine* 37, 2 (2016), 76–84.

[3] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57, 5 (2009), 469–483.

[4] Jordan Ash, Monica Babes, Gal Cohen, Sameen Jalal, Sam Lichtenberg, Michael Littman, Vukosi Marivate, Phillip Quiza, Blase Ur, and Emily Zhang. 2011. Scratchable devices: user-friendly programming for household appliances. In *International Conference on Human-Computer Interaction*. Springer, 137–146.

[5] Daniel Ashbrook, Shitao Stan Guo, and Alan Lambie. 2016. Towards augmented fabrication: Combining fabricated and existing objects. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 1510–1518.

[6] Alexander Berman, Francis Quek, Robert Woodward, Osazuwa Okundaye, and Jeeeun Kim. 2020. *"Anyone Can Print": Supporting Collaborations with 3D Printing Services to Empower Broader Participation in Personal Fabrication.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3419249.3420068

[7] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. 2008. *Survey: Robot programming by demonstration.* Technical Report. Springrer.

[8] Elin A Björling, Emma Rose, and Rachel Ren. 2018. Teen-robot interaction: A pilot study of engagement with a low-fidelity prototype. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. 69–70.

[9] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934* (2020).

[10] Paul Bovbel and Goldie Nejat. 2014. Casper: An assistive kitchen robot to promote aging in place. *Journal of Medical Devices* 8, 3 (2014).

[11] Leah Buechley, Daniela K Rosner, Eric Paulos, and Amanda Williams. 2009. DIY for CHI: methods, communities, and values of reuse and customization. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*. 4823–4826.

[12] Maya Cakmak, Crystal Chao, and Andrea L Thomaz. 2010. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development* 2, 2 (2010), 108–118.

[13] Yuanzhi Cao, Zhuangying Xu, Fan Li, Wentao Zhong, Ke Huo, and Karthik Ramani. 2019. V. ra: An in-situ visual authoring system for robot-iot task planning with augmented reality. In *Proceedings of the 2019 on Designing Interactive Systems Conference*. 1059–1070.

[14] Sonia Mary Chacko and Vikram Kapila. 2019. An augmented reality interface for human-robot interaction in unconstrained environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3222–3228.

[15] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. *ShapeNet: An Information-Rich 3D Model Repository.* Technical Report arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago.

[16] 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.

[17] 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.

[18] Hyungjun Cho, Han-Jong Kim, JiYeon Lee, Chang-Min Kim, Jinseong Bae, and Tek-Jin Nam. 2021. IoTIZER: A Versatile Mechanical Hijacking Device for Creating Internet of Old Things. In *Designing Interactive Systems Conference 2021*. 90–103.

[19] Matei Ciocarlie, Kaijen Hsiao, Adam Leeper, and David Gossow. 2012. Mobile manipulation through an assistive home robot. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5313–5320.

[20] 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.

[21] Alexander Dijkshoorn, Patrick Werkman, Marcel Welleweerd, Gerjan Wolterink, Bram Eijking, John Delamare, Remco Sanders, and Gijs JM Krijnen. 2018. Embedded sensing: Integrating sensors in 3-D printed structures. *Journal of Sensors and Sensor Systems* 7, 1 (2018), 169–181.

[22] Rüdiger Dillmann. 2004. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems* 47, 2-3 (2004), 109–116.

[23] Ohsung Doh and Ilkyu Ha. 2015. A digital door lock system for the internet of things with improved security and usability. *Advanced Science and Technology Letters* 109, Security, Reliability and Safety 2015 (2015), 33–38.

[24] M Ehrenmann, R Zollner, O Rogalla, and R Dillmann. 2002. Programming service tasks in household environments by human demonstration. In *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*. IEEE, 460–467.

[25] Kerstin Fischer, Franziska Kirstein, Lars Christian Jensen, Norbert Krüger, Kamil Kukliński, Maria Vanessa aus der Wieschen, and Thiusius Rajeeth Savarimuthu. 2016. A comparison of types of robot control for programming by demonstration. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 213–220.

[26] David Fischinger, Peter Einramhof, Konstantinos Papoutsakis, Walter Wohlkinger, Peter Mayer, Paul Panek, Stefan Hofmann, Tobias Koertner, Astrid Weiss, Antonis Argyros, et al. 2016. Hobbit, a care robot supporting independent living at home: First prototype and lessons learned. *Robotics and Autonomous Systems* 75 (2016), 60–78.

[27] Google. 2019. Blockly. https://developers.google.com/blockly/.

[28] Horst-Michael Gross, Steffen Mueller, Christof Schroeter, Michael Volkhardt, Andrea Scheidig, Klaus Debes, Katja Richter, and Nicola Doering. 2015. Robot companion for domestic health assistance: Implementation, test and case study under everyday conditions in private apartments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 5992–5999.

[29] Sachini Herath, Hang Yan, and Yasutaka Furukawa. 2020. RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, & New Methods. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3146–3152.

[30] Gaoping Huang, Pawan S Rao, Meng-Han Wu, Xun Qian, Shimon Y Nof, Karthik Ramani, and Alexander J Quinn. 2020. Vipo: Spatial-Visual Programming with Functions for Robot-IoT Workflows. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[31] Frédéric Kaplan. 2005. Everyday robotics: robots as everyday objects. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*. 59–64.

[32] Tarik Keleştemur, Naoki Yokoyama, Joanne Truong, Anas Abou Allaban, and Taşkin Padir. 2019. System architecture for autonomous mobile manipulation of everyday objects in domestic environments. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*. 264–269.

[33] Jeeeun Kim, Haruki Takahashi, Homei Miyashita, Michelle Annett, and Tom Yeh. 2017. Machines as co-designers: A fiction on the future of human-fabrication machine interaction. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 790–805.

[34] Yuki Koyama, Shinjiro Sueda, Emma Steinhardt, Takeo Igarashi, Ariel Shamir, and Wojciech Matusik. 2015. AutoConnect: computational design of 3D-printable connectors. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–11.

[35] Stacey Kuznetsov and Eric Paulos. 2010. Rise of the expert amateur: DIY projects, communities, and cultures. In *Proceedings of the 6th Nordic conference on human-computer interaction: extending boundaries*. 295–304.

[36] Jiahao Li, Meilin Cui, Jeeeun Kim, and Xiang'Anthony' Chen. 2020. Romeo: A Design Tool for Embedding Transformable Parts in 3D Models to Robotically Augment Default Functionalities. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 897–911.

[37] Jiahao Li, Jeeeun Kim, and Xiang'Anthony' Chen. 2019. Robiot: A Design Tool for Actuating Everyday Objects with Automatically Generated 3D Printable Mechanisms. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 673–685.

[38] Chuanhua Lu, Hideaki Uchiyama, Diego Thomas, Atsushi Shimada, and Rinichiro Taniguchi. 2019. Indoor positioning system based on chest-mounted IMU. *Sensors* 19, 2 (2019), 420.

[39] Catarina Mota. 2011. The rise of personal fabrication. In *Proceedings of the 8th ACM conference on Creativity and cognition*. 279–288.

[40] Pedro Neto, J Norberto Pires, and A Paulo Moreira. 2010. High-level programming and control for industrial robotics: using a hand-held accelerometer-based input device for gesture and posture recognition. *Industrial Robot: An International Journal* (2010).

[41] Pedro Neto, J Norberto Pires, and Anónio Paulo Moreira. 2013. 3-D position estimation from inertial sensing: Minimizing the error from the process of double integration of accelerations. In *IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 4026–4031.

[42] Donald A Norman. 2005. Robots in the home: what might they do? *Interactions* 12, 2 (2005), 65.

[43] William Odom, James Pierce, Erik Stolterman, and Eli Blevis. 2009. Understanding why we preserve some things and discard others in the context of interaction design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1053–1062.

[44] Lauro Ojeda and Johann Borenstein. 2007. Non-GPS navigation with the personal dead-reckoning system. In *Unmanned Systems Technology IX*, Vol. 6561. International Society for Optics and Photonics, 65610C.

[45] Lauro Ojeda and Johann Borenstein. 2007. Personal dead-reckoning system for GPS-denied environments. In *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*. IEEE, 1–6.

[46] Ortak. [n.d.]. Ortak 3D Printable Products - edelkrone. https://edelkrone.com/collections/ortak (Accessed on 03/09/2022).

[47] Mikkel Rath Pedersen and Volker Krüger. 2015. Gesture-based extraction of robot skill parameters for intuitive robot programming. *Journal of Intelligent & Robotic Systems* 80, 1 (2015), 149–163.

[48] Huaishu Peng, Jimmy Briggs, Cheng-Yao Wang, Kevin Guo, Joseph Kider, Stefanie Mueller, Patrick Baudisch, and François Guimbretière. 2018. RoMA: Interactive fabrication with augmented reality and a robotic 3D printer. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–12.

[49] Martha E Pollack, Laura Brown, Dirk Colbry, Cheryl Orosz, Bart Peintner, Sailesh Ramakrishnan, Sandra Engberg, Judith T Matthews, Jacqueline Dunbar-Jacob, Colleen E McCarthy, et al. 2002. Pearl: A mobile robotic assistant for the elderly. In *AAAI workshop on automation as eldercare*, Vol. 2002. 85–91.

[50] Print+. [n.d.]. print+ DIY products. https://www.print.plus/ (Accessed on 03/09/2022).

[51] 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.

[52] Mitchel Resnick, Brad Myers, Kumiyo Nakakoji, Ben Shneiderman, Randy Pausch, Ted Selker, and Mike Eisenberg. 2005. Design Principles for Tools to Support Creative Thinking. http://www.cs.umd.edu/hcil/CST/Papers/designprinciples.htm. (Accessed on 03/27/2021).

[53] David Roedl, Shaowen Bardzell, and Jeffrey Bardzell. 2015. Sustainable making? Balancing optimism and criticism in HCI discourse. *ACM Transactions on Computer-Human Interaction (TOCHI)* 22, 3 (2015), 1–27.

[54] Corey Shemelya, Fernando Cedillos, Efrian Aguilera, E Maestas, J Ramos, D Espalin, D Muse, R Wicker, and E MacDonald. 2013. 3D printed capacitive sensors. In *SENSORS, 2013 IEEE*. IEEE, 1–4.

[55] Himanshu Singh, Vishal Pallagani, Vedant Khandelwal, and U Venkanna. 2018. IoT based smart home automation system using sensor node. In *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 1–5.

[56] Steppschuh. [n.d.]. Steppschuh/Sensor-Data-Logger: Android Wear sensor data plotter. https://github.com/Steppschuh/Sensor-Data-Logger (Accessed on 03/09/2022).

[57] Michael Suguitan and Guy Hoffman. 2019. Blossom: A handcrafted open-source robot. *ACM Transactions on Human-Robot Interaction (THRI)* 8, 1 (2019), 1–27.

[58] Ja-Young Sung. 2011. *Towards the human-centered design of everyday robots*. Georgia Institute of Technology.

[59] Carlos E Tejada, Raf Ramakers, Sebastian Boring, and Daniel Ashbrook. 2020. AirTouch: 3D-printed Touch-Sensitive Objects Using Pneumatic Sensing. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–10.

[60] ThisAbles. [n.d.]. Assistive Technology Devices For Ikea Products. https://www.universaldesignstyle.com/thisables-assistive-technology-devices-for-ikea-products/. (Accessed on 03/09/2022).

[61] Panagiota Tsarouchi, Athanasios Athanasatos, Sotiris Makris, Xenofon Chatzigeorgiou, and George Chryssolouris. 2016. High level robot programming using body and hand gestures. *Procedia Cirp* 55 (2016), 1–5.

[62] Ron Wakkary and Leah Maestri. 2007. The resourcefulness of everyday design. In *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition*. 163–172.

[63] Weitian Wang, Rui Li, Yi Chen, Z Max Diekel, and Yunyi Jia. 2018. Facilitating human–robot collaborative tasks by teaching-learning-collaboration from human demonstrations. *IEEE Transactions on Automation Science and Engineering* 16, 2 (2018), 640–653.

[64] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. 2019. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1042–1051.

[65] Karl Willis, Eric Brockmeyer, Scott Hudson, and Ivan Poupyrev. 2012. Printed optics: 3D printing of embedded optical elements for interactive devices. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 589–598.

[66] Thomas Wrensch and Michael Eisenberg. 1998. The programmable hinge: toward computationally enhanced crafts. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*. 89–96.

[67] Hang Yan, Qi Shan, and Yasutaka Furukawa. 2018. RIDI: Robust IMU double integration. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 621–636.

[68] Jing Zhu, Jin Xie, and Yi Fang. 2018. Learning adversarial 3d model generation with 2d image enhancer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.