Mobiot: Augmenting everyday objects into moving IoT devices using 3D printed attachments generated by demonstration

ANONYMOUS AUTHOR(S)

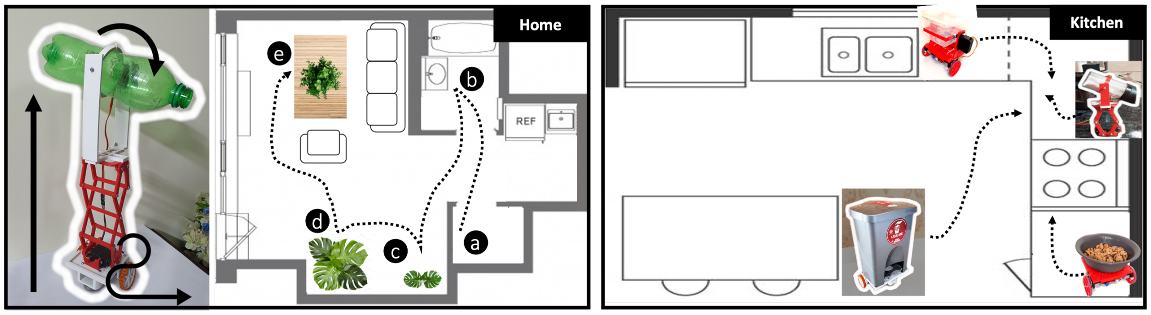

Fig. 1. Mobiot is an end-to-end pipeline that helps home users easily automate routine tasks by mobile smart objects. In the automated home, user can add actuation mechanisms to a bottle to (a) fetch itself, maneuver to (b) fill in water from the tap, then (c-d) travel to water plants at different location, including (e) adjusting the height to reach one on the coffee table. In a kitchen, a user can build smart cooking aids with objects collectively operating with various motions, fetching a bowl delivering water, spice cans sprinkling salt and peppers, and a dish to serve ready-to-eat foods, as well as moving a trash bin next to a counter closer to where food scraps are.

Recent advancements in personal fabrication made the story of novices automate routine tasks with mobilized everyday objects not too far away from today. 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-to-end pipeline 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 with 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 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.

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

Additional Key Words and Phrases: 3D printing, Personal Fabrication, Home-automation, Motion planning

ACM Reference Format:

Anonymous Author(s). 2018. Mobiot: Augmenting everyday objects into moving IoT devices using 3D printed attachments generated by demonstration. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The ascendancy of Industry 4.0 is propelling the utilization of automation in our everyday life, from industries to office spaces to households. The merge of the cyber-physical system and IoT connectivity is bringing the smart home [23] from incubator to actuality. Yet, the lion's share of the everyday objects does not comply with the provisions of automation, as many legacy entities are either inert or passive in nature. While home-users can easily achieve setting automatic turning on/off of a device (e.g., lights and thermostats) upon environmental sensing using commercial solutions (e.g., Alexa switch), actuation and mobility of an object has been under the unexplored area. As found in robotic vacuums that freely travel on the floor [20], assistive kitchen robots [8] and many more, the home with mobile agents that are capable to roam in larger space is the next key for automated home. The addition of mobility to everyday objects can ease and automate various daily routine tasks, such as self-organizing workspace, helping in food preparation, ambuling items to different places, etc. Moreover, mobilized agents can facilitate elderly care and aid accessibility based applications, such as indoor assistive navigation, serving medicines, foods, drinks, etc. However, such motions are not always localized and executed in between two fixed points, as found in most off-the-shelf IoT devices (e.g. SwitchBot[1]) or works such as Robiot [31]. Everyday objects can achieve fairly complicated maneuvering tasks by a combination of translation (2D roaming on a horizon) in a small/large space, rotation, and lifting motions. Hence, in the context of this work, the word *'mobile'* or *'mobility'* refers to the *'composition'* of these motions in series, applied to various target objects.

Yet, there remain three key challenges for novices to actualize this near-future:

- Most legacy objects are passive in nature, and existing IoT solutions do not support adding motion.
- Capturing design requirements are challenging for novices, such as type of motions and paths, shape of target objects, needed electronics, and transferring and decomposing those information into components for mechanism generation, 3D models for fabrication, assembling of electronics and programming. And,
- Authoring, re-purposing or re-using the existing mechanisms to obtain desired actions and adapt to different use-case scenarios.

In reality, it is not possible to replace all the legacy objects, while commercial robotic solution is either un-affordable or need advanced knowledge to be able to program for various actions. Although prior works tackled solutions for novices to turn passive legacy objects to be actuatable using attachment mechanisms [30, 31, 45], the range of motions of these active objects has been fairly limited, while personalizing other robotic solutions to freely travel a wider space with desired motion planning requires expert knowledge in robot programming (eg., [22, 35]) or too expensive (eg., [19]) to barely afford one, which is not enough to automate multiple everyday objects at home.

Recent advances in personal-fabrication along with Maker's movements [37] have promised a story of non-experts to develop their everyday automated systems (e.g., [26]), paving the way toward co-design and fabrication of highly custom products using desktop printers at home, as witnessed in DIY headphones by Print+[2] and co-manufactured mechanical devices[3] and more. Users buy some pre-built components then 3D-print the rest to make the whole product fit individual needs and tastes. Users also can print different mechanisms and attachments for legacy objects to stretch their range of usage[31]. Yet, it still presents a barrier to end-users as it demands expertise from capturing real-world requirements to modeling to final fabrication.

We present Mobiot, an end-to-end pipeline to mobilize everyday objects by demonstration, using 3D printed attachments. By demonstrating a desired motion with a commercial smartphone as a tool for both designing and action

[1]https://www.switch-bot.com/
[2]www.print.plus
[3]https://edelkrone.com/collections/ortak

authoring, Mobiot automatically captures mobility requirements and decompose it to create 3D printable mechanisms and a list of electronics, and produces step-by-step instructions for assembly, as well as program to operate modules to replicate user-specified motions. The user takes a photo of a target object, then either puts the device on it and performs the motion demonstration by mobilizing it or keeps it on hand to mimic the required motion. The captured data using IMU sensors and the input image then serve to extract three critical piece of information at a single shot: *(i)* geometry requirements, *(ii)* mechanisms to obtain the desired mobility, and *(iii)* motion planning information. Based on this, the system automatically generates 3D models that fit the target real-world object and reflects mechanism parameters, displays a set of circuitry needed & driving options, and produce codes to operate mobilization using assembled mechanisms and circuitry. Following the instructions provided by the system, the user can print and assemble all the components and load the program to finally transform a legacy object into a smart automotive object.

Mobiot contributes,

- An enabling system with an end-to-end pipeline for home users, to mobilize passive everyday objects at low investment.
- From simple real-world demonstrations by the user, our system automatically captures design requirements at one-shot and generates the mechanical components to be 3D printed, list of electronics and program to robotize everyday objects to reflect the demonstration.
- The end-user can also easily author the motions of the fabricated mechanisms for fine-tuning and reuse or re-purpose, not demanding expertise about programming and motion planning.

2 RELATED WORK

2.1 Reality-based Augmentation of the Physical World by 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. The practice of deriving information from the real-world scenario and synapsing the extracted data to augment physical entities has been traversed by exploiting different tools such as augmented reality, virtual reality, 3D scanners, etc. RealitySketch transforms a drawn sketch into a overlayed visualization that can interact with the physical world [53]. DART augments a storyboard into the working experience [34] . RoMA [43] presents an AR-based platform to design and 3D-print different artifacts or extend an existing item on-the-fly, Printy [4] embeds circuitry into a 3D model allowing novice users to fabricate functional internet-connected objects. Encore enables creation of 3D printed attachment parts on the irregular surfaces of existing objects [11]. AutoConnect lets users connect two real-world objects, such as a cup and chair in the desired arrangement [28], while Retrofitting existing object to extend its capacity has been explored in several ways with different motivations such as for accessibility purpose, and more [12, 15, 31, 45]. In a similar vein, Reprise allows users to specify, generate, customize, and fit adaptations to everyday objects to enhance usability [12]. Patching 3D Objects utilizes a depth camera and a milling bit to a 3D printer to enable bi-directional on-the-fly design [55].

A handful amount of prior works focus on developing active mechanisms for augmenting physical world, especially for novice users. Robiot [31] and Romeo [30] closely relates to this context. However, the domain and purpose of the motions of ours and prior works are non-overlapping. Robiot actuates a part of object with 1-DOF at a time, mostly a short travel between two fixed points. Moreover, actuation operates on the entities, as specific portion of an object is moved, pushed, elevated, or triggered by the actuation mechanism. Mobility, on the other hand, adds navigation and locomotion capability to the whole body. Romeo adds re-configurable capacity into a 3D object, creating 'robotic arm' mechanisms to perform localized actions, while Mobiot offers to combine both non/localized actions. Also, Romeo relies on digital CAD tool for path planning that may present real-world conflicts, while we capture real-world demonstration at scale.

2.2 Personalizing Smart Systems for Automated Everyday Lives & Accessibility

Domestication of smart automatic systems can provide with health assistance [22], support independent living [19], or assistance in domestic works [13]. For example, iRobot Roomba [4] and K¨archer RoboClean [5], for instance, can aid the daily cleaning task. In later works (e.g., [50, 51]), the impact of customization on the personalization of smart systems has captivated our attention. Another application of smart objects cuts the area of *accessibility*. Mobile platforms and smart objects can serve the prepared foods (Fuzzy-Bot[47]) and drinks, remind about daily routine activities, such as taking medicines, using bathroom and assist in navigating elderly people (Pearl[44]), fetch diaper/medicine box or laundry basket, adjusting a display height to the wheelchair position, and many more.

Many of the concept of personal robotics and automated systems relies on off-the-shelf robotic platforms [9, 25, 27], which often not affordable, hard to program for trivial but varying tasks per individual, such as tailoring roaming plan to deliver medicine basket from kitchen to room 2. Moreover, platforms are limited by the size of the platform to adapt a variety of everyday objects, in the scale of the mechanisms and shape to adapt their geometric dissimilarities. Mobiot focuses on augmenting by 3D printing of custom active mechanisms that adapt fair range of geometry deviations. High-level input from the user is captured to extract the required low-level information to translate it into desired mobility mechanisms, and the examples are developed to facilitate the repeated daily life applications, such as, assisting in cooking, bringing a water bottle, fetching diaper/medicine box or laundry basket, adjusting a display height to the wheelchair position, etc.

2.3 Motion/Action Planning & Authoring by Demonstration

For an end-user, the main challenge to automate everyday environments comes from capturing motion requirements and authoring the actions of the fabricated active mechanisms. V. Ra [9], in this context, provides an AR-based platform for task authoring using a cellphone. Blockly [21] and Vipo [25] present visual authoring methods to ease authoring, while Learning from Demonstration (LFD) [3] is a technique to capture user demonstration and transfer the extracted information to execute the learned action. The imitation technique can utilize sensor-on-teacher to record the user's action and harvest the data to author a robotic system. Billard et al. [6] explores different prospects of robot programming via demonstration. Gesture recognition [36] is a considered method of capturing such demonstration. Gesture Coder [33] allows users to record multi-touch gestures to invoke different applications. It is known that hand-held device is an efficient way to record demonstration and program the actions of a robotic system using accelerometer [39], voice, handwriting, and gesture [29], camera and other sensors Dillmann [16], Ehrenmann et al. [18]. A later work utilizes a 3D tracker to demonstrate tasks in a virtual environment and actuate a robotic manipulator in real life Aleotti et al. [1]. Similarly, wearable sensors are used to mimic and reauthor the action of a robotic manipulator in 3D space Wang et al. [58]. In sum, prior works found programming via demonstration accessible for end users, in which smartphone could be a solution to aid their demonstrations to capture desired motion of active objects. We undertake a similar approach, use a smartphone to enable easy capturing of desired motion by demonstration then generate key components for fabrication and programming.

[4]https://www.irobot.com/roomba
[5]https://www.kaercher.com/int/home-garden/robocleaner.html

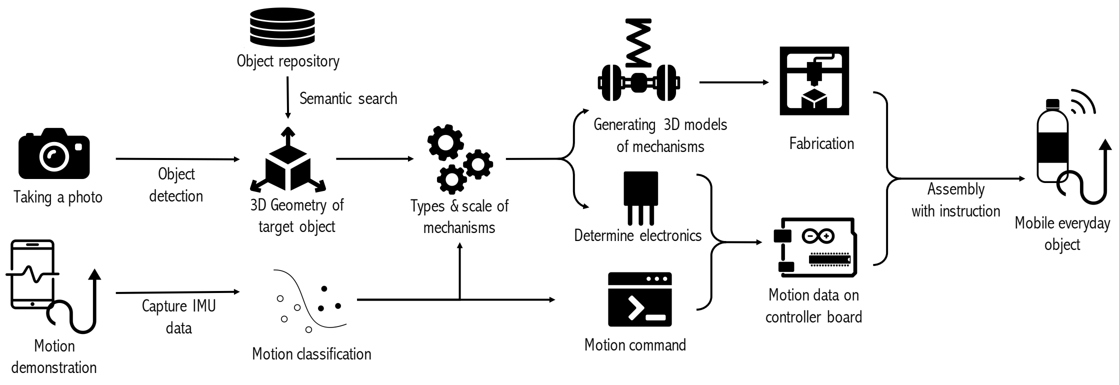

Fig. 2. Overview of the Mobiot system. A user demonstrates desired motions of an input target object to get ready-to-3D print mechanisms and controller board components (code and list of electronics) to add mobility to legacy objects

3 MOBIOT: AN END-TO-END TOOLKIT FOR AVERAGE USERS TO TRANSFORM EVERYDAY PASSIVE OBJECTS INTO MOBILE OBJECTS

We present Mobiot, an end-to-end system for non-expert users to mobilize their passive everyday objects, capture design requirements simply by demonstrating the motion and fabricate, assemble, and code mechanisms using low-cost 3D printers as in Figure 2. We set three design goals taken from the classic HCI design principles [46]:

- **Low Threshold:** The system can enable average users to design mechanisms to add mobility, that can be attached to everyday objects without expertise, at low cost. Mobiot takes the real-world demonstration as an one-shot input then automatically translate it to the mechanical entities. Leveraging a cellphone both as a design and authoring tool, automatic creation of list of required electronics with custom assembly instruction let the user easily convert a legacy object into a smart mobile objects using low cost 3D printers at home.
- **Wide wall:** The system can support creation of mobility mechanisms that fit a wide range of everyday objects. Objects with different sizes, shapes, and weight can be captured for mobility requirements description to be translated into robotic device design, according to the user's demonstration.
- **High ceiling:** The user can create fairly complex motion, combining different degrees of freedom and a series of actions consisting of three discrete motions at various combinations. For example, a self-watering bottle can travel in 2D plane, lift, rotate, then travel again to different execution points. Also, using a number of robotic devices together, a user can also automate fairly complex daily task, such as assistive cooking.

3.1 System Overview

Mobiot links the user's demonstration to the functional 3D-printable mechanisms and program to actuate needed electronics. As illustrated in Figure 3, a user starts with *Designing from Scratch* or *Reusing the Existing*, to provide an image as input. This image is then used to obtain the name of that object to run a semantic search through repository to retain 3D geometry of target object. In the second segment of the input phase, the user runs a provided app on the cellphone to record IMU sensor data to recover movement information of the demonstration. A user can put the cellphone on the target object while moving it mimicking the large scale movement, or simply move the phone by hand imitating a desired motion. The initial inputs are uploaded to the system to attain three information used to create 3D printable mechanisms and program to run on electronics: *(i)* geometry requirements, *(ii)* mechanisms to realize the desired mobility, and *(iii)* motion planning information, which will be detailed in the Section 4.1.

User's demonstration is categorized into three motion classes of translation, rotation, and lift, depending on the prime movements captured in certain segments. Users can view the required mechanisms, utilize the recognized motion sequence to compose task or fine-tune if necessary, as will be detailed in Section 4.2. Once everything is complete, the user gets ready-to-3D print files of mechanism and a program code to operate this mechanism upon motion descriptions. The system finally shows the required commercial electronics such as motors and purchase link to the user based on the types of mechanisms and object weight that user can simply place an order, as will be 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.

3.2 User Workflow and Design Scenarios

Recent smart home assistants' motions include carrying spices from shelves (Robot Kitchen), pouring ingredients (CafeX), serving foods by patrolling (robot waitress), feeding pets (Cat Mate), watering plants (Aquarius), fetching laundry (Stretch RE1), taking photos by cam adjustment (Temi). These mostly comprise a combination of translation, rotation, and lift, that we attempted to decompose these requirements to replicate in 'need-based' examples. In this section, we showcase a variety of everyday objects that are turned to be active in motion to automate home, following our design pipeline.

The below Figure 3 shows the step by step walkthrough of a user. Here we speculate a story of Nancy at home. Depending on her design scenario for creating new motions or editing the existing to re-purpose, she may go through different design routes that simplifies the required design steps.

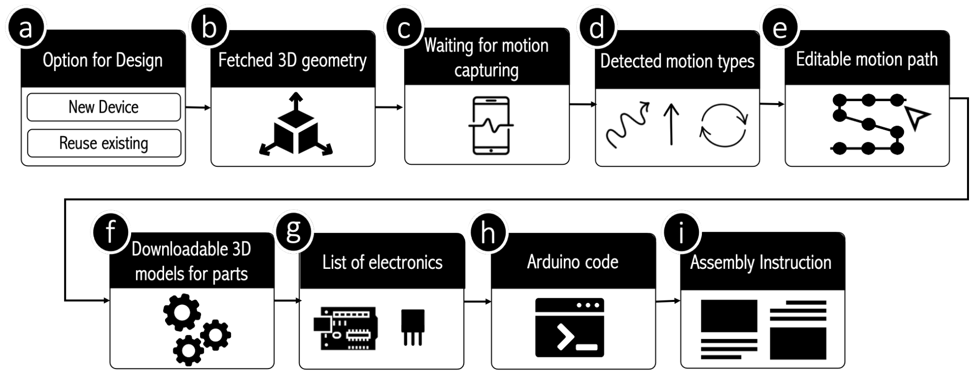

Fig. 3. A step-by-step walkthrough of the design system. (a)-(e) requires user inputs for design to proceed to the next step, and (f)-(i) generate outputs for a user to assist fabrication and assembly

3.2.1 Scenario 1. Creating a New Mobility Mechanisms to a New Device: Trash bin. Nancy at the kitchen with full hands of food scraps wants the trash bin comes near the sink while cooking. Yet, she may not want this stay forever in the kitchen during normal days as it may be stinky. To make the trash bin that travels and returns back to its original location near the door. Nancy first took a photo of the trash bin using her smartphone running the Mobiot app, then puts the device on top of the trash bin and dragged in to the kitchen for demonstration. Satisfied with the motion and its trajectory from the screen, she proceeds to the next step to download mechanical parts for 3D printing and order electronics for assembly. She copied the generated program to write to Arduino, and the self-organizing trash bin is ready for use as shown in Figure 1a.

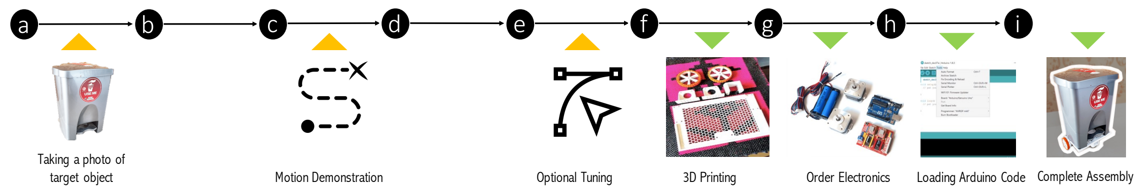

Fig. 4. Walkthrough to create mobility mechanisms for a trash bin and corresponding user actions. Yellow arrows indicate input and green arrows indicate output.

3.2.2 *Scenario 2. Re-target an Existing Motion Mechanisms with Newly Recorded Motion: Laundry Basket.*

Liking her self-organizing trash bin a lot, she bought a smart one. Instead, as the scale and shape of the laundry basket is similar to her trash bin, Nancy would like to reuse the mechanism generated for it because there was no auto-rolling basket available on market yet. She wants her basket to locate near the bathroom during daytime collecting dirty towels, then move to the laundry room during the night.

She opened the app but chose an option for "Reuse the existing mechanism" this time, to record new motion by demonstrating it using her app which is placed on top of basket while moving from door to door. She slightly adjusted the trajectory by dragging the anchor shown on the Mobiot display, as she needs to offset more the curve near the corner where floor lamp sits. This time she does not need to print or buy anything, overwrite the program only. Once loaded, the rolling platform now adapts her laundry basket instead, as depicted in Figure 1b.

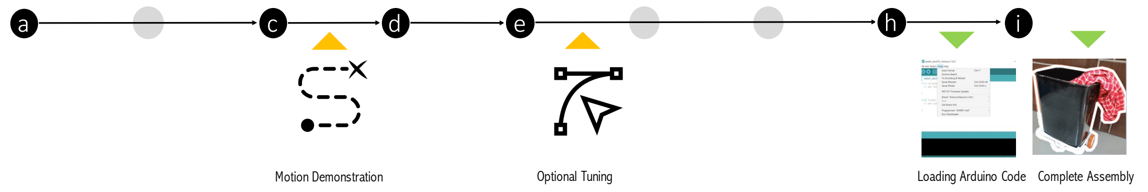

Fig. 5. Re-using a the mechanism from trash bin example with new authored motion and new target object for laundry basket, which minimizes input making the design process is much simpler by skipping several stages

3.2.3 *Scenario 3. Appending New Motions to Existing mechanisms: Spice can.*

Dreaming of a serving robot in the future of life, Nancy made her spice jars on the table self-serve her guests over Christmast dinner and reposition to the origin. She went through the same design scenario but by uploading the 3D model of her 3D printed spice jar, placing it on the smartphone to mimic the rotation then iterate the same process for fabrication, as illustrated in Figure 6.

She then realized that the jar needs to be lifted while peppering into a bowl, thus opened the app to choose the option for "appending movement" to the existing mechanism. She recorded lifting by demonstrating it, tuned the height to fit her dinning table. Mobiot generated new 3D parts to add lifting and listed another motor to manipulate it, as well as a new Arduino program to reflect changes, as shown in Figure 7. Attaching the scissor mechanism below the previous 3D printed attachment, her salt and pepper jars are ready to welcome guests on the table, as the result shown in Figure 1d.

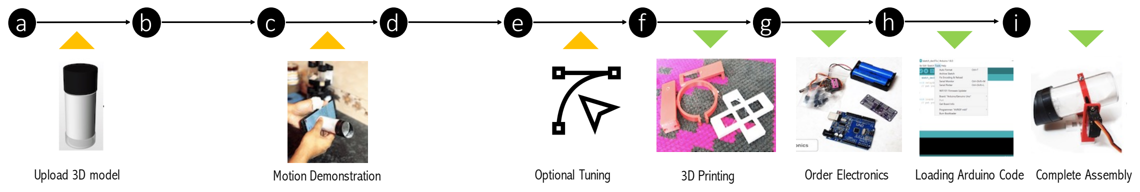

Fig. 6. Creating a tilt motion for a spice can

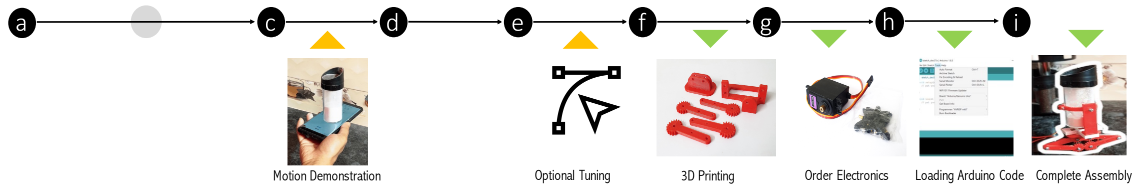

Fig. 7. Appending a new motion (lift motion) to an existing mechanism that is added to a spice can

3.2.4 *Scenario 4. Creating a Complex Series of Motion: Auto-planting Water Bottle.*

As Nancy plans for a long international trip to attend friends' wedding, she wants to design a self-watering bottle to keep her plant pots live while she is gone. The water bottle needs to start its weekly journey from near her auto facet to fill the water, move to her balcony and water a pot by tilting to pour water, move to the next pot for watering another plant. As the height of two pots differ, the mechanism needs to lift the bottle to reach the different spot. Nancy recorded the series of motion subsequently, starting from lifting to reach water tap, moving to the balcony, watering the first pot by tilting, relocating to the next pot and lifting, then tilting to water it. The list of motion segments are captured accordingly, then she designed the pouring action by combining the lift and rotation together using the Mobiot interface, downloaded all parts that constitute all three mechanisms and the program to operate these motions subsequently for final assembly of self-watering bottle as shown in Figure 1e. While she is away, she can use her cellphone to trigger watering.

3.2.5 *Scenario 5. Assisting Cooking.*

During busy days, Nancy gets a handful amount of time for preparing breakfast. She wants to have her breakfast readied while sitting in front of her laptop or checking emails. In the kitchen space, the liquid egg is to be poured into a bucket, some salt and spice to be shook into the bucket, prepare the mixture and then pour it to the pan, and give the bucket an initial wash. The breakfast table has to have different mayonnaise, sauce, & pepper spilling bottles, a milk carton, a self-organizing coffee cup, and a platform to take her bread or salad bucket and pour sauce or dressing on it. She recorded motions for each of the items and fed them to Mobiot individually. She then 3D printed and assembled all the parts generated from Mobiot and attached them to the target objects. She designed the mixing, pouring, and shaking actions using the Mobiot platform and uploaded the generated code to the mechanisms. Now she just needs taps on the cellphone for the omelet to be prepared, and with different sequences of taps, she prepares breakfast with different flavors. Using a *future version of Mobiot* she adds sensing ability to them, and now from one fingertip, she would be able to enjoy her breakfast ready while focusing on her other busy morning routine.

4 IMPLEMENTATION

The Mobiot pipeline involves several steps for adding mobility to everyday objects. From taking user input to generating 3D printable mechanisms we require different features and states regarding the object and the motion. The geometry and motion features are extracted and mapped systematically for generating the mobility mechanisms and programming the device. A step by step implementation is described as following.

4.1 Capturing Requirements by Demonstration

4.1.1 Geometry Information. To create 3D printable mechanisms that fit the target real-world object, it is crucial to obtain the required geometry information for appropriate parametric design, which is captured using a camera, . Mobiot contains a repository of 3D models derived mostly from Shapenet [10]. When user uploads a photo of the target object, Mobiot passes the image through YoloV4 [7] object detection algorithm. From the detected object class, the system makes a semantic search in the repository and fetches a 3D model corresponding to the object class. Optionally, a user can grab a 3D model of the target object from any public repository and provide it as an input directly. As such open repository has been populated with the 3D models of many real-world objects, this would soon become a better approach to guarantee accuracy of obtained 3D geometry.

4.1.2 Motion Information Classification. User's demonstration is captured via cellphone IMU sensor that consists of accelerometer and gyroscope data using an android application [6]. The app is configured 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. User can upload the recorded data, stored in the cellphone, directly to the Mobiot interface. The captured motion data acts as initial input for the mobility requirements.

The next step is to classify motion types, so we can segment motion trajectories upon the type of motions. Currently, the system classifies the three most common motion types- translation, rotation, and lifting motion, by searching the axis where the most salient movement occurs, which is detailed as following,

Translation. The translation motion is a 2D movement over a horizontal surface. The 2D position of the cellphone can be obtained from the recorded IMU sensor data using dead-reckoning, double integration or using the state-of-art machine learning approach. We estimate the 2D trajectory using RONIN [24], a machine learning based approach. From the trajectory information, we calculate the total amount of distance traversed. Comparing this value with a threshold, the system determines the requirement of a 2D horizontal motion.

Lifting. RONIN does not provide the 3D trajectory information from non-tango devices. Hence, estimation of lift of the cellphone during the demonstration is subject to additional computation. We find the lift via a double integration approach from the acceleration data along Z-axis. Afterwards, using binary threshold the system detects if a lift motion was triggered in the demonstration.

Rotation. From the IMU sensor, we can obtain the rotation vector at a given time. The rotation vector is highly accurate due to the android sensor fusion that combines gyroscope and accelerometer data to compensate for errors and biases. If roll or pitch, obtained from this rotation vector, exceeds a threshold angle, the system classifies this as a rotation motion.

[6]https://github.com/Steppschuh/Sensor-Data-Logger

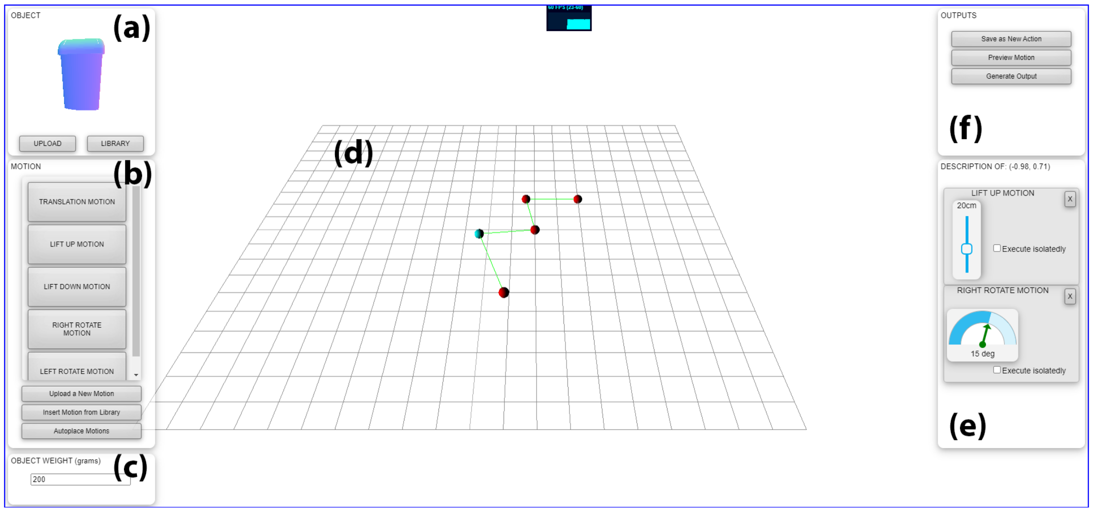

Fig. 8. A screenshot of the user interface (a) Fetched 3D model, (b) Detected Motions' list and Motion tools, (c) Additional information (e.g. weight), (d) Task design scene for importing and previewing motions, (e) Appending motions and adjusting parameters (f) Animating or Generating outputs, saving custom actions

4.2 User Interaction and Task Design

After the classification and quantification of the motions, they are represented as different blocks in the interface. At this point, the user can import different motion blocks to the scene and manually design complex tasks or proceed with the motion sequence automatically recognized by the system, or have a combination of both of them.

4.2.1 Manual Task Designing. The user can manually design a task from the represented motion blocks. These motion blocks 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. For instance, the user can import a translation motion block to the scene. This presents the user with the anchor points on the trajectory. Afterwards, the user can drag the positions of the anchor points if needed, or additionally add new anchor points to the trajectory. Other motion blocks, such as lift or rotation, can be appended to different anchor points, as can be seen in figure 8. User can tune the amount of lift or rotation appended to each anchor points and can drag-drop to organize the sequence of motions, add pause or delay in between different actions, and choose to execute them either isolatedly or synchronously. During the isolate execution, one motion at a time is rendered.

4.2.2 Action Module Designing. In our daily activities, many of everyday tasks consist of modular actions, such as mixing, pouring, etc. Action module designing involves multiple low level motions to have one action. Using our interface, the user can arrange different motions in series to obtain a particular action, name it and save to the library. For instance, the user can record one rotation motion, sequence leftwards and rightwards rotation in series multiple time to have a mixing action module. It provides the opportunity of different actions to be incorporated to the main task without re-designing it. Some other examples can be *synchronous lift + rotate = pouring, lift then rotate multiple times = shaking*, etc. During the task design, the user can re-utilize these modules, adjust their ranges to obtain the desired actions with ease. In future these high-level action descriptions can also be brought under a repository to further aid the task designing.

4.2.3 *Automated Motion Sequence.*

During the the motion classification, the system also determines the point of execution of different motion types. If the user proceeds with the automatically recognized motion sequence, the system arranges the blocks on the scene as they appear in the demonstration. However, the recorded motion and recognized sequence may contain human errors, noises originating from the natural movements of the body, or adjusting upon the environmental changes. Thus, in a similar manner to the manual task designing, the user can fine tune the parameter if necessary, add more motions blocks to different points, re-arrange them, etc. The user can also remove any unwanted motion, recognized by the system. The imported motions by the automated system are by default executed isolatedly. However, the user can select different blocks to be executed synchronously as well.

4.2.4 *Additional Information about Target Object.*

The scale of 3D printed mechanisms and the electronics specification might vary upon the weight of the object, as the heavier real-world object will need larger but stronger motor. Mobiot lets the user to input the approximate weight of the object whether it will be heavier than 200g. The torque for mobilizing the target is computed based on this weight which will be determined under/over this threshold. Additionally, the weight plays role while generating different 3D models, such as defining size of motor mountings, number of gears in the lifting mechanism, thickness of the clamps in the rotation motion, etc.

4.3 Generating Outputs

4.3.1 *Generating 3D Models and Required Circuitry of Mechanisms.*

At this point, the system acquired all sufficient features to generate the 3D printable mechanisms. The target object may be equipped with one or more mechanisms to replicate the desired mobility that user recorded.

Translation. Geometry of target 3D object determines the size of the platform to install wheels for 2D travels, and weight determines the required torque for the driving motors. 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' datasets. Additionally, we choose motors that have 20% more torque than required to incorporate other items within the mechanisms. We choose between two models of stepper motors- 28byj48 and NEMA 17 variants to support objects' weight under 200grams and beyond. After the motor-type is selected based on the required torque, wheels, and motor clamps are then chosen from pre-built parametric templates as library that we provide, using OpenSCAD[7] scripts. We add a boarder on the chassis edges to prevent the object from falling off while moving, while the user can choose to use glue if necessary.

Rotation. The rotational mechanism uses servo motors where the required torque for rotating is calculated as previously described. The mount for the servo motor and the supporting parts are added based on the height of the object. The clamping point is considered to be the center of gravity of the object, which is estimated from the analysis of the object's 3D model. For generating the clamps to hold the object, we use the method of Chen et al. [11] for computing the circumference of a cross-section of the object. Thickening the boundary and extruding it, we obtain the required clamp.

Lifting. The lifting mechanism is a modified version of the scissor lift system. Counter gears, driven by a servo motor, are used for the scissor action and adjustable linkers are used to achieve the desired height. The lifting height is

[7]https://www.openscad.org

obtained from the motion demonstration, from where the system determines the required linkers to achieve that height. The selection between a single-sided or double-sided lifting mechanism is determined based on the object's weight.

4.3.2 Circuitry and Driving Components: We use Arduino Uno, one the most common controller, for the mechanism operation. Arduino boards are easy to program and offer a vast range of pre-built libraries. For the driving elements, we use servo and stepper motors. From different variants of servo motors, three popular models are utilized- MG90, SG5010, and MG996R by Tower Pro. The required torque determines the selection of the servo motors, controlled from the Arduino Uno using a PCA9685 12-bit PWM I2C servo driver. It has a total of 16 port for driving up to 16 servo motors. On the other hand, for 2D mobility, we utilize bipolar stepper motors. The NEMA 17 variants offer a vast range of torque. Besides, the 12V version of the 28BYJ-48 stepper motor is suitable for small & lightweight operations. For driving the steppers, the popular & open-source Arduino-CNC shield is utilized, which is stackable to the controller board. For WiFi connectivity, we use low-cost ESP8266 IoT enabled module.

Not all above components for fabrication and electronics assembly are always required, for example, if only rotation and lifting mechanisms are captured from user demonstration, the system will choose the components and circuitry for these motions only. The required components and circuitry along with their store links are saved to be presented before providing the assembly instructions.

4.3.3 Obtaining Parameters to Program Controller Board. The recorded motion information needs to be converted to machine commands. All the 3D models are generated by the system based on parametric designs, thus the system possesses all the required information for translating the motion. First, we determine the motion equation and then parse the information into machine instructions. The processing of data for each of the motion types is described as following:

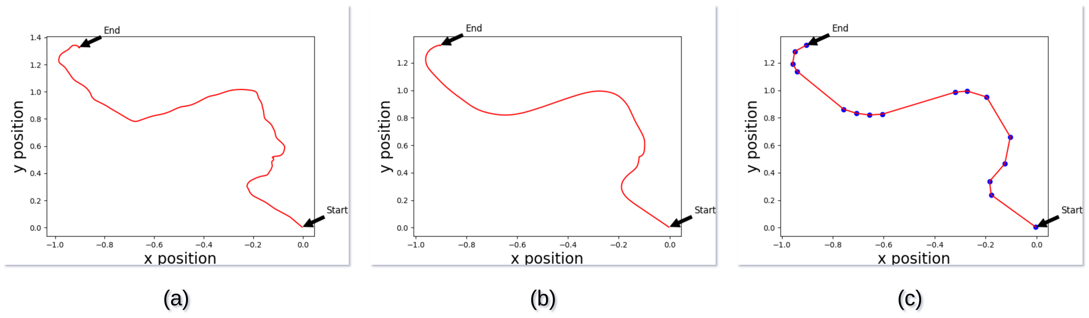

Fig. 9. Processing of Horizontal Motion Data (a) Raw positional data obtained using RONIN [24] (b) Moving average filtering (c) Finding the anchor points.

Translation. The raw trajectory ($\Gamma^r_{(x,y)}$) of the 2D motion contains noises (figure 9 (a)) 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 9 (b)) as per equation 1 to obtain a smooth trajectory ($\Gamma^f_{(x,y)}$).

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

The window size (M) of this moving average filter depends on the length of the trajectory, and for our application, we consider 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 as shown in Figure 9c. We iterate through the points on the smoothed trajectory data ($\Gamma^{f}_{(x,y)}$) and calculate the distances and tangents from point to point. We consider a distance threshold and tangential angle threshold for finding the anchor points. If any two consecutive line segments fall under this criteria, the corresponding points are considered to be anchor points. After determining all the anchor points, the system calculates the distances and angles between these anchor points. This information is then utilized to generate the motion commands, how far to travel, when to rotate to switch the proceeding direction.

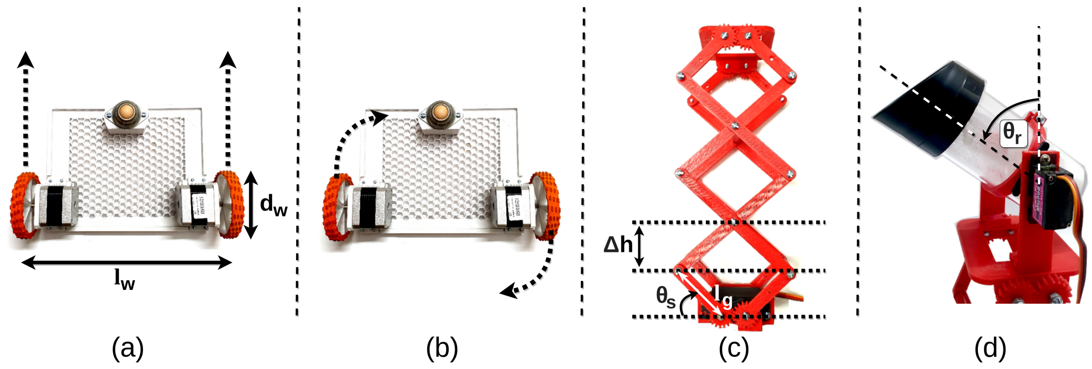

Fig. 10. Realizing motions from parameters with three parametric design as library (a) Forward motion using the differential drive (b) Taking a turn using the differential drive (c) Setting the height of the lifting mechanism (d) Rotating an entity to a particular angle, in our pepper can example.

The Figure 10 illustrates three parametric designs to design motion mechanisms and drive them with assembled electronics. 2D translation is achieved utilizing a differential drive mechanism (Figure 10a,b). By adjusting the rotation of the motors, the mechanism can reach desired locations. The number of rotations for a motor, required to traverse a particular linear distance, depends on the wheel diameter (d_w) as in Figure 10a. Whereas, for going across an arc, two parameters are necessary- wheel diameter (d_w) and distance between the wheels (l_w) (See Figure 10b). For simplicity, we neglect the frictional coefficients and wheel slip. The stepper motors are driven based on the number of steps. Considering these parameters and the steps per revolution of the stepper motor, the number of steps required for covering a particular linear distance (See Figure 10a) are computed.

Lifting. The initial phase of getting the height of the lift motion is processing the accelerometer data along Z-axis. For this, we first grab the linear acceleration data along the Z-axis as shown in Figure 11a. This data is computed by the Android sensor fusion algorithm that removes the gravitational acceleration from the Z-axis acceleration data. Afterwards, the data is passed through a moving average filter and thresholding (See Figure 11b). Afterwards we perform a double integration of the filtered data to estimate the amount of lifting height (h_l) (See Figure 11c). The lifting mechanism comprises gear stages and linker stages as depicted in Figure 10c above. The total lifting height (h_l) can be considered to consist of small step heights (Δh). Considering the linker length (l_g) and adjusting the servo angle ($\theta_s = arcsin\Delta h/l_g$) of the servo, the desired lift is achieved.

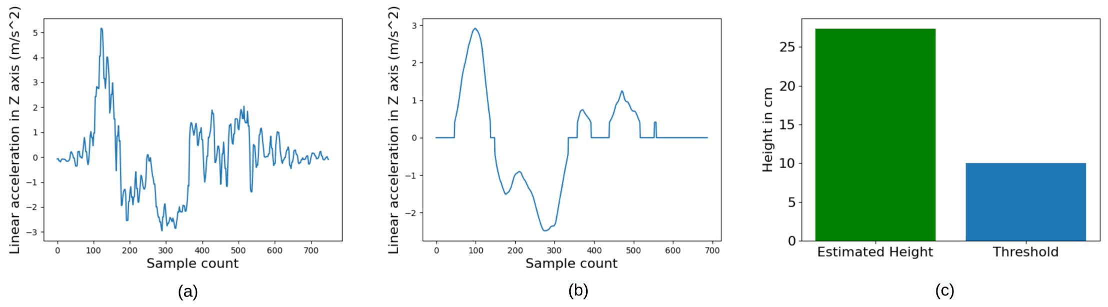

Fig. 11. Estimation of Lifting Height (a) Raw linear acceleration data along Z-axis (b) Filtering and thresholding (c) Estimated lifting height from double integration.

Rotation. The rotation vector represents the cellphone's orientation about x, y, and z axes and is expressed in the form of quaternions. For practical usage, we first convert the quaternions to Euler angles [2] (roll, pitch, and yaw) by:

$$\begin{bmatrix} roll \\ pitch \\ yaw \end{bmatrix} = \begin{bmatrix} arctan(\frac{2(w*x+y*z)}{1-2(x^2+y^2)}) \\ arcsin(2(w*y-z*x)) \\ arctan(\frac{2(w*z+x*y)}{1-2(y^2+z^2)}) \end{bmatrix} \tag{2}$$

Where, x, y, z are the rotation vector components along corresponding axes, and w is the scalar component of the rotation vector. We check the maximum values of roll and pitch to estimate the rotation angle. This angle value can be directly parsed by the servo motor of the rotation mechanism (figure 10 (d)).

4.3.4 Transferring the Motion Information via Auto-Programming. Having all parameters that will be used to program driving motors for actuation, the next step is to translate them into the writable Sketch program and transfer to the board. Arduino IDE is a handy platform to program all the Arduino boards. After all the motion data is translated into motor values, Mobiot generates a string containing the machine commands. The machine commands consist of identifier keywords- go (g), turn (t), rotate (r), lift (l), delay (d), etc. Each of the keywords is followed by its corresponding values. Mobiot patches this command string with a pre-built code template which contains codes for driving the motors based on the command string. This code can be copied and flashed directly to the Arduino board via the Arduino IDE or the Arduino online code editor [8].

4.4 Re-authoring Motions and Re-usability of Produced Mechanisms

After adding mobility to an existing object, the system creates an entity consisting of the extracted geometry and motion information then registers to the system as library. By exploiting this stored information, a fabricated mechanism can be re-authored and re-targeted. Mobiot presents three methods of re-utilizing an existing mechanism- using the existing mechanism but with different authored motion, retrofitting other objects to existing mechanism, and adding new mobility mechanisms with the existing one.

4.4.1 Re-authoring Motions. Mobiot offers the user to edit the motion of an existing mechanism. The user can re-record the motion and use the Mobiot to convert it to machine commands, allowing a laundry basket which initially travels

[8]https://create.arduino.cc/editor

from bathroom to the washing room into a new path that ravels from bedroom in early morning to the kitchen during breakfast to collect table clothes. Afterward, the new motion can be uploaded to the Arduino board or transferred over WiFi if an ESP8266 is used.

4.4.2 Retrofitting. Retrofitting lets the user to attach a new object, such as one designed for spice jar to adapt water bottle, to the existing mechanism by generating necessary 3D printable parts. The process starts with capturing the geometry information of the target object. Afterwards from the previously stored information the system rations the geometry of the new object to determine the changes. If the new object is larger or smaller than the previous one, the system generates new attachments or chassis corresponding to the new geometry. For example, if a previous bottle is equipped with a rotation mechanism and a new bottle has a larger diameter, the system will generate a new clamp to be replaced in the mechanism to hold the new one. However, we assume the weight of the new object doesn't deviate significantly from the previous one.

4.4.3 Appending New Mechanism. The user can also append new mechanisms with existing one as we demonstrated in spice can example in Section 3.1. For this, the user can capture a new motion demonstration only. Following, the system determines the types of motions, and hence, the required 3D models of the mechanisms. The system generates new motion commands based on the newly recorded motion data. Moreover, the system also manifests if there is any requirement for additional circuitry. The user then can 3D-print additional parts and assemble them with the existing.

4.5 Software

Mobiot is implemented using Python for back-end using django[9] (Python based web framework). JavaScript and custom CSS styling build the front-end. jQuery[10] has been used to handle user responses, and three.js[11] helps visualizing the scene and 3D models. We used OpenSCAD[12] for parametric design of template that are automatically reconfigured upon design parameters that are obtained through user input and analysis from back end.

5 EVALUATION

To evaluate Mobiot's performance, **JK: add the summary about accuracy test, the goal, method, etc.**

We use the pre-trained model of ResNset18 provided by RONIN for motion recover from IMU data, and hence, for the accuracy of estimated trajectory, we refer to RONIN, where the Absolute Trajectory Error (ATE) has been reported 1.62 and Relative Trajectory Error (RTE) has been reported 1.67 for unseen datasets. For the technical evaluation of Mobiot, we further set up experiments for estimating the accuracy of the translation motion parsed by the mechanism.

Figure 12 shows performed evaluations on different motion types. We use a marker to draw the ground truth of a trajectory with known distances and angles for motion recording and equip a translation mechanism with a marker (Figure 12a and b). Afterward, we upload the command data manually on the Arduino board, and let the mechanism parse the command data. After a total traversal of 200cm and two right-angle turns, we found an approximate deviation of 6mm to 9mm from the end point. The lateral displacement is found negligible. We also load an object on the mechanism and let it traverse 400cm straight. In this traversal, the deviation of the mechanism from the end-point was found ranging between 2cm to 8cm. It might manifest the mechanism's initial placement needs to be adjusted more frequently for larger motions than smaller ones, although, in most cases larger motion is required by larger objects and the relative

[9]https://www.djangoproject.com/
[10]https://jquery.com/
[11]https://threejs.org/
[12]https://www.openscad.org/

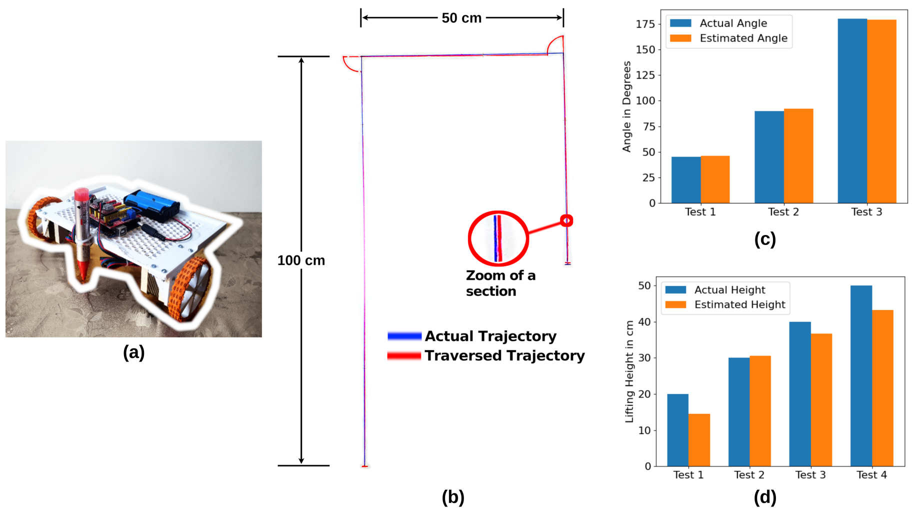

Fig. 12. Evaluation of Motion Accuracy (a) Test Setup to Evaluate Translation Motion (b) Ground Truth and Traversed Trajectory for Translation Motion (c) Estimated Angle vs Actual Angle for Rotation Motion (d) Estimated Height vs Actual Height for Lift Motion.

displacement might not be very significant. Incorporating odometry and wheel encoders could further improve for more frequent usage that need more accurate movement.

Figure 12c compares the recorded motion by cellphone's roll or pitch angle in action and the estimated angle by the system. For this criteria, we record the motion of the cellphone by rotating it to a known angles (45^o, 90^o, 180^o). Afterwards, the system estimates the roll and pitch from the recorded motion data. A maximum deviation of 3 degree has been observed but is negligible in practical usecase scenarios for spice jars for example.

Evaluation of estimation of lifting heights is illustrated in Figure 12d. We lift the cellphone to known heights while recording the motions. Accelerometer data contains heavy noise. Since we use double integration method to estimate the vertical displacement from the accelerometer, the noise accumulates and leads to deviations. The relative deviation is within the range of 3cm, which could highly impact for water bottle that may spill over water, while relatively ignorable for serving foods that human intervention can compromise. With further improvements in 3D pose estimation from IMU data, the lifting height can be estimated more accurately. Additionally, training a regression model with the IMU data to estimate the vertical displacement can be another solution to this.

6 DISCUSSION & FUTURE WORK

In this section, we discuss remaining challenges found from Mobiot with potential future work which can be easily extended to a broader scope of research about end-user enabling interactive systems.

6.1 Mobiot vs. Multi-agent Robotics

Mobilizing multiple objects brings the concept of multi-agent robotics forward. Many prior works, for example Hermits [38] in this context, focused on mechanical shell design, borrowing the swarm robot approach. This work utilizes a locomotive toy- Toio for mobility, and also incorporates passive mechanical shells for its reconfigurable architecture. However, Hermits allow custom mechanical shells to be actuated by the locomotive device which requires special arrangements and mechanisms, such as attachments for magnetic coupling and hollow shells for attaching the locomotive device. These shells are required to be designed specifically to allow a designated motion translations via different gear mechanisms. Further, ShapeBots [54] uses a vision-based trajectory planning for a multi-agent system, where each robot contains a Fiducial marker. An interface allows users to configure their positions and actuators. This allows the robots to act as physical displays or in-situ assistants, wipe debris off a table, perform tangible gaming, create dynamic fences, and move physical objects. Such requirements are not abundant to adapt everyday objects' various geometry, however, utilizing small mobility drivers used in such works would be a promising future work for us to minimize the scale of the mechanisms in the future.

In a large scale, RoomShift presents a haptic environment for virtual reality [52]. It deploys Roomba robots with scissor lift mechanisms to lift and re-locate furniture in order to augment virtual scenes with physical objects. It exposes a potential way of augmenting daily life objects with mobility to realize automated home providing haptic feedback for further accessibility. As the type of mechanisms only support objects that can be lifted and the system complexity does not allow end-users to utilize such a system for daily life activities in various scale, Mobiot could be a complementary solution for transforming individual objects into mobile agents. We aim at exploring the inclusion of multi-agent robotics to interconnect different mechanisms from different objects for automatic task completion in future works.

6.2 Automatically recovering 3D model geometry from 2D photo input

In case the semantic search of target object from ShapeNet fails, we preliminarily integrated photogrammetry techniques to generate the 3D model of the target object. Works such as Pixel2Mesh [57], Pixel2Mesh++ [59], 3D model generation using GAN [62], etc., provides 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 a perfectly fit 3D printable mechanisms. We expect to re-integrating automatic recovery of 3D mesh using this method when there are more advances and benchmark 3D models available to improve accuracy in retained model soon in the future.

6.3 Limitations of the number of 3D models

Currently, we fetch a 3D model from the 2D image from a pre-built repository, mostly built from the Shapenet [10] models. Hence, the number of available models in this benchmark adds a limitation. Moreover, object detection and its classes solely depend on YoloV4 [7]. Enriching the number of models in the repository and adding more classes to the machine learning can resolve the issues. Alternately, users can download 3D models of the target object from any repository and feed them to the system. Also, users can use substitute models that are coherent in their geometry requirements (eg. using a laundry bucket model instead of a trash bin with similar geometry).

6.4 The relative scales and attachment mechanisms

Although the scale of the mechanisms is flexible depending on the target object's physical dimension, the minimal dimension is limited by motors and electronics. Hence, for the trash bin or laundry bucket example, the mechanisms seem coherent, although for spice can or water bottle it seems a bit bulkier and obtrusive. With the tiny commercial driving actuators such as Toio and advances of new electronics and motors that will be much smaller and more powerful and become available in the market, we will investigate how to address the scale limitation. Additionally, we currently do not use shells for electronics, but can be inherited from works such as Retrofab [45] or Desai et. al. [14]. To ensure the object put firmly on a platform, we utilize the clamps generated as described in section 4.3.1, and the boundary across the translation motion chassis in addition to gluing keeps the objects from falling apart. However, for toppled and non-uniform objects, works such as AutoConnect [28] can be utilized.

6.5 Height accuracy retained from acceleration

For the positioning purpose, we rely on RONIN [24]. In practice, dead reckoning based on step counting [5, 41, 42] and double integration [32, 40] are popular methods of positioning. In the context of double integration, RIDI [61] shows promising performance. Step-counting based positioning system does not fit our application. Apart from that double integration method also suffers from the accumulative error that eventually leads towards a huge deviation in positioning. Android Tango devices have an inbuilt positioning system. However, such devices are not ubiquitous which made us settle down to RONIN. However, it only provides the 2D position of the audience with non-tango devices. As a result, the lifting height cannot be determined precisely with the existing algorithms. As a workaround, we use the acceleration data and find whether and where the lifting is triggered, and make a rough estimation using double integration of the accelerometer data. Yet, we let the user adjust the height in the fine-tuning step. In the future, if any better algorithm comes, then it may be integrated along.

6.6 Limitations in Motion: Types & Authoring

Our system currently incorporates three types of motions. However, actions of everyday objects may require additional motions, such as twisting, curved lifting, sliding, etc. Also, action triggering in compound motions (eg. translation and rotation together) is accomplished only using spatial programming. In future work, we will incorporate different triggering types.

6.7 Translation motion & initial placement

The differential drive mechanism for the translation motion does not incorporate the more complex condition than flat home floor such as terramechanics,e.g. wheel slippage, frictional co-efficient, etc. To have a firm grip in the wheels, we currently utilize TPU based tires, generated automatically based on the wheel parameters, or the user can also rubber bands of specific thickness. The initial position and orientation is a crucial factor for mobilizing an object to the target destination. Apart from that, anchor points are determined based on binary thresholding, which may lead to deviation of the traversed trajectory from the authored or actual trajectory. Additionally, changing the initial orientation of the mechanisms can lead to wrong trajectory. The future addition of a closed-loop system, gyro sensor, and adaptive thresholding can eradicate such issues for extension of Mobiot in-the-wild. Moreover, the mechanisms require re-authoring once the origin point is changed (e.g. moving the origin point of a trash bin from living room to dining space). A library of authored motion can help resolving the issue from where the user can trigger a particular trajectory. However, as the triggering methods are out of scope of this work, we leave this as a future improvement.

6.8 Custom interactions and sensing

Sensing enables users to add custom interactions to different interfaces. IoT-based sensors provide the opportunity for remote sensing and action mappings, such as automatic turning on and off lights and fans [49], smart door opening and closing system [17],etc. Off-the-shelf IoT devices, such as Google Nest[13], allow users to control inter-connected home

[13]https://store.google.com/us/product/nest_audio

appliances via voice commands. Using LeapMotion[14] or Kinect[15] devices, users can use gestures for mapping custom actions. We assume users can easily adapt these commercial sensing and triggering solution, assuming such sensing and custom action mapping are out of the scope of this work. Currently, we utilize an IoT-based WiFi module (ESP8266) and a JavaScript based interface for triggering the actions. The inclusion of the WiFi module leaves the floor to the user for the inclusion of IoT-based sensors integration. Platforms, such as Blynk [16], let novices add IoT control or sensing to different devices. Unification of Adafruit IO[17] and IFTTT[18] can allow users to control devices over IoT directly from Facebook chat or Whatsapp messengers.

Inclusions of other sensors on-board (e.g. IR, Lidar, Sonar for obstacle sensing) can capacitate the outcome, and we intentionally chose open source Arduino and IoT devices to provide easy integration. Moreover, recent developments in 3D printable sensing system (eg. [48, 56, 60]) unleash the potentiality of embedding such sensors within the mechanisms in another future works.

6.9 Motion Visualizing and Embedding Floor-Plan

Currently we utilize the three.js platform to animate the composed motions with the target object. As a future work, we aim at embedding the floor plan of an apartment or target area in our system. This will help the visualization of the trajectories. Recent inclusion of lidar sensors in cellphones (e.g. Iphone 12 Pro) paves the way of 3D scanning of a scene and generating the 3D floor plan of an arena, which can be embedded in the system in near future for detailed visualization. Such integration can also facilitate other accessibility applications, such as guided navigation for elderly care to navigate to different places (e.g. bathroom, kitchen, etc.).

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] Simon L Altmann. 2005. *Rotations, quaternions, and double groups.* Courier Corporation.

[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] 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.

[5] Stephane Beauregard and Harald Haas. 2006. Pedestrian dead reckoning: A basis for personal positioning. In *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication*. 27–35.

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

[7] 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).

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

[9] 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.

[10] 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.

[11] 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.

[14]https://www.ultraleap.com/
[15]https://developer.microsoft.com/en-us/windows/kinect/
[16]https://blynk.io/
[17]https://io.adafruit.com/
[18]https://ifttt.com/

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] Jodi Forlizzi and Carl DiSalvo. 2006. Service robots in the domestic environment: a study of the roomba vacuum in the home. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. 258–265.

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

[22] 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.

[23] Richard Harper. 2006. *Inside the smart home.* Springer Science & Business Media.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] Jennifer L Leopold and Allen L Ambler. 1997. Keyboardless visual programming using voice, handwriting, and gesture. In *Proceedings. 1997 IEEE Symposium on Visual Languages (Cat. No. 97TB100180)*. IEEE, 28–35.

[30] 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.

[31] 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.

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

[33] Hao Lü and Yang Li. 2012. Gesture coder: a tool for programming multi-touch gestures by demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2875–2884.

[34] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*. 197–206.

[35] Wim Meeussen, Melonee Wise, Stuart Glaser, Sachin Chitta, Conor McGann, Patrick Mihelich, Eitan Marder-Eppstein, Marius Muja, Victor Eruhimov, Tully Foote, et al. 2010. Autonomous door opening and plugging in with a personal robot. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, 729–736.

[36] Sushmita Mitra and Tinku Acharya. 2007. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37, 3 (2007), 311–324.

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

[38] Ken Nakagaki, Joanne Leong, Jordan L Tappa, João Wilbert, and Hiroshi Ishii. 2020. HERMITS: Dynamically Reconfiguring the Interactivity of Self-Propelled TUIs with Mechanical Shell Add-ons. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*.

882–896.

[39] 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).

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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).

[47] HS Selaka, MAWT Deepal, PDR Sanjeewa, HP Chapa Sirithunge, and AGBP Jayasekara. 2018. Fuzzy-Bot: A Food Serving Robot as a Teaching and Learning Platform for Fuzzy Logic. In *2018 Moratuwa Engineering Research Conference (MERCon)*. IEEE, 565–570.

[48] 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.

[49] 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.

[50] JaYoung Sung, Rebecca E Grinter, and Henrik I Christensen. 2009. " Pimp My Roomba" designing for personalization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 193–196.

[51] JaYoung Sung, Rebecca E Grinter, and Henrik I Christensen. 2010. Domestic robot ecology. *International Journal of Social Robotics* 2, 4 (2010), 417–429.

[52] Ryo Suzuki, Hooman Hedayati, Clement Zheng, James L Bohn, Daniel Szafir, Ellen Yi-Luen Do, Mark D Gross, and Daniel Leithinger. 2020. Roomshift: Room-scale dynamic haptics for vr with furniture-moving swarm robots. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–11.

[53] Ryo Suzuki, Rubaiat Habib Kazi, Li-Yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. 2020. RealitySketch: Embedding Responsive Graphics and Visualizations in AR through Dynamic Sketching. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 166–181.

[54] Ryo Suzuki, Clement Zheng, Yasuaki Kakehi, Tom Yeh, Ellen Yi-Luen Do, Mark D Gross, and Daniel Leithinger. 2019. Shapebots: Shape-changing swarm robots. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 493–505.

[55] Alexander Teibrich, Stefanie Mueller, François Guimbretière, Robert Kovacs, Stefan Neubert, and Patrick Baudisch. 2015. Patching physical objects. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 83–91.

[56] 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.

[57] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 52–67.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] 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.