

UCLA

UCLA Electronic Theses and Dissertations

Title

Smartening Legacy Objects: Transforming Legacy Physical Objects into Smart Entities through Robotic Augmentations

Permalink

<https://escholarship.org/uc/item/61b1d7c8>

Author

Li, Jiahao

Publication Date

2024

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Smartening Legacy Objects:

Transforming Legacy Physical Objects into Smart Entities through Robotic Augmentations

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mechanical Engineering

by

Jiahao Li

2024

© Copyright by
Jiahao Li
2024

ABSTRACT OF THE DISSERTATION

Smartening Legacy Objects:
Transforming Legacy Physical Objects into Smart Entities through Robotic Augmentations

by

Jiahao Li

Doctor of Philosophy in Mechanical Engineering

University of California, Los Angeles, 2024

Professor Dennis W. Hong, Co-Chair

Professor Xiang Chen, Co-Chair

Recent digital technologies have envisioned a future ecosystem where objects are not merely passive entities but actively participate in and respond to the interaction between the user and the environment. While the majority of existing physical objects remain *unsmart*, researchers and practitioners have made efforts to incrementally upgrade legacy objects to perform digital tasks. However, there has been limited support for the automation of physical tasks, which is essential for users within certain demographics (e.g., the elderly and people with disabilities) or in scenarios of situational impairments.

To bridge this gap, my thesis research aims to augment physical legacy objects with the ability to perform physical tasks automatically. My research explores two different types of augmentations: *(i)* involves augmenting the physical objects to perform tasks without external assistance, and *(ii)* augmenting the objects to be more manipulable by robots. Specifically, I have introduced innovative methods for designing mechanical mechanisms that either empower objects to actuate on their own or enable actuation through an external robotic arm. A crucial aspect of enabling such robotic manipulation is a novel data collection technique I proposed, aimed at accurately estimating the pose of objects whose appearance changes during manipulation. Furthermore, given the diverse range of physical

objects present in our daily lives, a central goal of my research is to democratize the design process, making it more accessible for augmenting these legacy items.

In summary, my thesis aspires to enhance physical objects with the approach of robotic augmentations, smartening them to perform physical tasks for a more interactive and responsive future.

The dissertation of Jiahao Li is approved.

Veronica Santos

Yang Zhang

Dennis W. Hong, Committee Co-Chair

Xiang Chen, Committee Co-Chair

University of California, Los Angeles

2024

To my parents and Iris the Mermaid.

TABLE OF CONTENTS

List of Figures	xii
List of Tables	xxi
Acknowledgments	xxii
Curriculum Vitae	xxiv
1 Introduction	1
1.1 From the Conventional to the Smart Realm	1
1.2 Challenges and Research Questions in Smartening Legacy Objects with Physical Capabilities	2
1.2.1 Enabling Individuals in Customizing Augmentations	3
1.2.2 Enabling Augmentations to Adapt to Complex Tasks	4
1.3 Research Goals	5
2 Background and Literature	7
2.1 History of Smart Home and Related Technologies	7
2.1.1 Early Explorations of Home Automation	8
2.1.2 The Explosion of Modern Smart Homes	8
2.1.3 The Gap of the Current Smart Home Technology	9
2.2 Research Trends in Smartening Legacy Objects	11
2.2.1 General-Purpose Personal Robots for Home Automation	11
2.2.2 Augmenting Physical Objects	12
2.3 Democratizing the Smartening of Legacy Objects	15

2.3.1	Design Tools for Personal Fabrication	15
2.3.2	Design Tools for Robotic Motion	16
2.4	Summary of the Related Work	16
I		18
3	Enabling Users to Design and Generate Add-on Mechanisms to Actuate Legacy Physical Objects	19
3.1	Related Work	21
3.1.1	Personal Robotics	21
3.1.2	Reality-based Design Tools	22
3.1.3	Designing and Prototyping Functional Objects	23
3.2	Robiot: Mechanisms to Actuate Everyday Things	25
3.2.1	Overview of Robiot’s Workflow	25
3.2.2	Examples Generated by Robiot	27
3.3	Implementation	29
3.3.1	Preprocessing	29
3.3.2	#1 Extracting an object’s 2D and 3D representations	30
3.3.3	#2 Determining the type of joint	31
3.3.4	#3 Finding maneuverable parts and grounds	33
3.3.5	#4 Generating Actuation Mechanisms & Instructions	35
3.3.6	Software and Hardware	37
3.4	Design Sessions	37
3.4.1	Participants	37
3.4.2	Apparatus, Tasks and Procedure	38

3.4.3	Results and Findings	41
3.5	Discussion	43
3.5.1	Future Technical Work	43
3.5.2	Scale of Mechanisms	44
3.5.3	Camera Angle to Capture Desired Motion	44
3.5.4	Designing Motion Beyond Given Affordances	44
3.5.5	Sensing and Designing Custom Interactions	45
3.5.6	Performing Ungrounded Action	45
4	Enabling Users to Design and Generate Embedded Mechanisms to Robot- ically Augment Default Functionality for Legacy Objects	47
4.1	Introduction	47
4.2	Related Work	49
4.2.1	Computational Design of Shape-Changing Objects	50
4.2.2	Reality-based Design Tools	51
4.2.3	Interactive Design of Robotic Characters	52
4.3	Overview of Design and Fabrication Process	53
4.3.1	Preprocessing	53
4.3.2	Segmentation of the Transformable Part	60
4.3.3	Searching for Robotic Arm Configurations	61
4.4	FABRICATION AND SOFTWARE IMPLEMENTATION	70
4.5	Examples Generated using Romeo	72
4.6	Design Sessions	73
4.6.1	Participants	73
4.6.2	Apparatus, Tasks and Procedure	78

4.6.3	Results	79
4.6.4	Observations & Findings	80
4.7	Discussion, Limitations & Future Work	82
II		85
5	Making Physical Objects Robotically Manipulable with 3D-Printable Add-on Mechanisms	86
5.1	Introduction	86
5.2	Categorizing Challenges of Handheld Objects for Robotic Manipulation	90
5.2.1	Objects Manipulated as a Whole (Static Objects)	92
5.2.2	Objects Manipulated by Constituent Parts (Dynamic Objects)	94
5.3	Related work	95
5.3.1	Augmenting Generic Robotic Interfaces and Everyday Objects	95
5.3.2	Relationship to Robotic Research	96
5.4	Examples: Roman Makes Handheld Objects Robotically Manipulable	98
5.4.1	Manipulating Objects as a Whole	99
5.4.2	Manipulating Objects by Constituent Parts	101
5.5	System Overview of Roman	107
5.6	Hardware Implementation	108
5.6.1	A Magnetic Gripper to Attach to, Recognize, and Transfer Motion to an Object’s Add-on Mechanism	108
5.6.2	Mechanisms for Manipulating Handheld Objects with Different Motion Profiles	110
5.7	Software Implementation: A Tool for Robotic Engineers to Specify Custom Motion Profiles for Object-Specific Mechanisms	115

5.7.1	Custom Motion Profiles	116
5.7.2	User Interface	116
5.8	Validation with Expert interview	118
5.8.1	Participants & Procedure	119
5.8.2	Results and Feedback	120
5.9	Limitations, TradeOffs, future work	122
5.9.1	Performance testing	122
5.9.2	Mechanisms interfering with normal use	122
5.9.3	Incorporating sensing & perception	123
5.9.4	Generalizability of the design	123
5.9.5	Trade-off between torque and the size/complexity of mechanisms	123
5.9.6	Manipulating objects with multiple movable parts or multiple consecutive manipulations	124
5.9.7	Opportunities to support novice users	125
5.9.8	Areas of hardware improvement	125
6	Enabling Pose Estimation of Appearance-changing Physical Objects	126
6.1	Introduction	126
6.2	Related Work	130
6.2.1	Object pose estimation	130
6.2.2	Pose data collection	130
6.3	Appearance-changing objects	133
6.3.1	Deformation	133
6.3.2	Viewing-angle dependent	133
6.3.3	Articulated	134

6.4	RoCap pipeline	134
6.4.1	Eye-to-hand camera calibration	135
6.4.2	Data collection	136
6.4.3	Data labeling	139
6.4.4	Data processing and augmentation	140
6.5	Evaluation	141
6.5.1	Implementation of pose estimation pipeline	142
6.5.2	Quantitative evaluation	142
6.5.3	Qualitative evaluation	144
6.6	Discussion	145
6.6.1	Limitations	145
6.6.2	Handling Occlusion	147
6.6.3	Automatic Changing of Mechanical States	148
6.6.4	Leveraging Robots for Large-Scale Data Collection	148
7	Summary	150
7.1	Limitations and Future Work	152
7.1.1	Enabling More Complex Tasks	152
7.1.2	Adapting to Humans and Environments	153
7.1.3	Studying Human Behavior	153
	Bibliography	155

LIST OF FIGURES

3.1	Robiot enables end-users to take a video and automatically generates mechanisms that can actuate everyday objects to perform simple physical tasks. All the four examples were designed by participants in our design session: lowering a lamp while busy soldering (a), waving a hand up to open a trashcan afar (c), setting up a scheduled house plant water spraying system (e), and asking the drawer to open itself when having no extra hand (g).	20
3.2	Overview of Robiot’s end-to-end pipeline for generating actuation mechanisms from a user’s input video.	25
3.3	A screenshot of the user interface provided to the user once Robiot generated a set of candidate mechanisms from a given video demonstration: the corresponding 3D model (a), the generated mechanisms (b), and sliders to filter the list by range of motion, torque and speed (c).	26
3.4	Automatic stapler is one common office appliance, that Robiot can robotize from a cheap manual stapler	27
3.5	Robiot mechanism attached on top of a soap bottle performs pressing to squeeze liquids without touching	27
3.6	A manual faucet can automatically turn to release water by attaching a mechanism to pull the handle.	28
3.7	Automatically adjusting mirror can aid a user who wears make-ups and contact lens using both hands	28
3.8	Optical flow analysis shows distinct difference between prismatic (a) and revolute (b) joints by comparing the least-square fitted radius (prismatic joint’s fitted radius is much larger, extending beyond the camera view).	32
3.9	Technical sketches of the three mechanisms showing how they are actuated. . . .	32

3.10	Some objects due to their geometry and texture provide very few feature points. To solve this problem, we incorporate a user’s operating hand whose motion correlates to the manipulated object’s.	33
3.11	Three mechanisms and their parameters (gear rack appears twice for both types of joints). For revolute joints, different mechanisms can be considered and modeled after the same set of parameters – l_1, l_2, l_3 and α	34
3.12	Two cases where gear rack does not apply	35
3.13	Participants in our study took videos of them manipulating everyday objects (a), whereby Robiot generated corresponding actuation mechanisms for participants to explore on our user interface (b)	39
3.14	Robiot generates an instruction for assembly (a), then a participant can assemble and install following that instruction (b), to interact with the robotic things using gesture (c) lowering the lamp height to work with optimal brightness when soldering (d)	40
4.1	Romeo enables a user to transform a Minion model into a coin-stealing piggy-bank by selecting the mid-section of the Minion as the transformable part (a), specifying motion path for the transformed part to pick up a coin and place it in the bank (b), generating the corresponding transformable robotic arm (c) and 3D printing and installing the result (d-e).	48
4.2	Existing examples of objects with multiple functionalities by transforming its geometry: a stealing coin piggybank (a), a transformable cup holder (b), and an iPad cover folded into a stand (c).	49
4.3	A screenshot of Romeo’s user interface: a) target object; b) reference object; c) functional buttons, from left to right: selecting transformable part, specifying motion points and action, generate embedded robotic arm, animation, export and d) button to restart the current step	54

4.4	Romeo enables selecting part of an object to transform by sweeping cross-sectional area along X/Y/Z axis.	55
4.5	Users can specify the motion points for the end-effector to follow in order (abc) and the corresponding action to be taken (d).	57
4.6	Romeo allow users to first specify the 2D position on a reference plane (ab), and then the third dimension along a perpendicular reference line (cd). A spherical widget is used for specifying the end-effector orientation (d). Lack of orientation specification may result in bad result (fg).	58
4.7	Romeo lets a users to specify the target surface for attaching action (a) and further define the orientation of the surface (b)	59
4.8	End-effector can be on different part of the object: static part of spatula (a) or end of the transformable part of Minion piggybank (b)	61
4.9	Two types of segmentation, (a) slender (b) non-slender shapes.	62
4.10	Four parameters to translate the coordinates. The figure is borrowed from Wikipedia: DH Parameters (Accessed 5/5/2020)	63
4.11	Visualized workspace (a). Double clicking makes the motion points outside, highlighted in red (b), snaps to the closet workspace (c).	65
4.12	Corresponding robotic arm representatives and the joint placement for three distinct cases.	66
4.13	If the steering joint locates at the first joint while the transformable part lies in the middle, both cases of b and c will cause collision between geometries.	67
4.14	Matching the orientation by aligning the x-axis of the end-effector with the arrow direction of the spherical widget control. The coordinate system represent the target orientation of the end-effector (a). Black arrow indicates the specified orientation (b).	69
4.15	Two types of joint connectors.	71

4.16	Romeo generates screw holes (a) and gripper (b).	72
4.17	By robotizing a paper towel holder, a user can pull one sheet without contaminating the others.	74
4.18	A conventional stamp can be robotized by Romeo to become a self-inking stamp.	74
4.19	Romeo can robotize a spatula to automatically stir the pot when unattended.	75
4.20	A tissue box can be stick to the door and wipe the doorknob every time someone opens the door.	75
4.21	A cup holder can extend the lower part of it to attach to the chair.	76
4.22	A flowerpot can be robotized to automatically water the plant according to a predefined schedule.	76
4.23	Different 3D objects can be augmented into a stealing coin piggy bank using Romeo. The examples are made by the participants in the design sessions: a dodo bird (a) and a Tesla cybertruck (b)	77
4.24	A participant installs the printed components (a) following a provided instruction (b).	79
4.25	Participants' rating on the Romeo assessment. DQ refers to questions asked in the design session with 8 participants and AQ refers to questions asked in the assembly session with 4 participants	81
4.26	Participants' design of coin-stealing piggy bank of their chosen 3D objects: (a) piggy bank, (b) baby yoda, (c) minion, (d) rilakkuma, (e) polar bears, (f) stitch, (g) Tesla cybertruck and (h) dodo bird. (Original 3D models are designed by Thingiverse users: layerone, MarVinMiniatures, sota919, Anthonylu, MakerBot, Erinfezell, wov, stargatedalek	82

5.1	Roman is a novel robotic design making everyday handheld objects more robotically manipulable, <i>i.e.</i> , easier to be manipulated by a conventional robotic arm. This figure shows a sequence of a Roman-enabled robotic arm picking up a wire cutter on the desk and performing a wire cutting task collaborating with a human (a-d). Roman provides a magnetic gripper (e) for the robotic arm to easily attach and augment the wire cutter with Roman mechanism (f). Snapping in the mechanism using magnets (g), the gripper can actuate the gear-rack movement on the wire cutter (h) to perform the cutting task by squeezing the cutter’s handles (i).	87
5.2	A human hand manipulating a variety of objects that present challenges for robotic arms such as using a whisk (a), rotating a screwdriver (b), squeezing a wire cutter (c), and opening a jar lid (d).	91
5.3	Why manipulating a screwdriver might require range of motion beyond a robotic arm’s capability: (a) normal working grasp/distance; (b) when the distance increases, the grasp changes and the end-effector can no longer rotate around the screwdriver’s axis; (c) in some edge cases, the manipulation might be interfered by the physical surroundings, <i>e.g.</i> , the ground.	91
5.4	Everyday handheld objects are often not manipulable by a generic robotic arm with a common parallel gripper: when manipulating an object as a whole, speed and range of motion are two main limitations and; when manipulating an object by its constituent parts, the dexterity required to both grasp and perform a range of manipulation (squeezing/releasing, twisting and pumping) is the main challenge.	93
5.5	Configuration of the pin-in-slot mechanism (a) to produce periodic up and down motion with a knife (b) and an alternative configuration of the pin-in-slot mechanism (c) to produce periodic side to side motion for spreading peppers from a spice bottle (d).	100

5.6	Configuration of the bevel gear mechanism to produce rotational motion at an angle on a screwdriver (a), allowing for two attachment configurations (b, c); the spur gear mechanism with a reverse reduction gear ratio produced higher rotation speeds for a whisk (d) and tilting it by a small angle further expands the range of motion at its end (e).	101
5.7	Configuration of the gear rack mechanism to produce linear motion (a) in a single direction to squeeze the handle of a spray bottle (b).	102
5.8	Configuration of the spur gear mechanism to produce rotational motion to squeeze together the tips of chopsticks (a), and a practical demonstration of the chopsticks being manipulated with the mechanism attached (b).	103
5.9	Configuration of the spur gear mechanism with a reduction gear ratio to produce rotational motion with high torque in order to rotate the door knob (a), and a practical demonstration of the door knob being twisted with the mechanism attached (b, c).	104
5.10	Two examples of the bevel gear mechanism to produce rotational motion at an angle to unscrew the lid (a, b) or to rotate the pepper grinder (c, d).	104
5.11	Configuration of the gear rack mechanism to produce bi-directional linear motion in order to squeeze a bottle of hand sanitizer (a), and a practical demonstration of the bottle of hand sanitizer being squeezed with the mechanism attached (b, c).105	
5.12	Combination of two separate mechanisms with a reduction gear ratio to produce high torque: a spur gear to cut around the can (c), and a gear rack to pierce the can (also using spur gears to increase torque) (d). A ratchet mechanism (f) is used to maintain the position of the piercing mechanism (<i>i.e.</i> , keep the handles squeezed), which allows the gripper to switch to the other mechanism (c) for cutting the can open.	105

5.13	Configuration of the gear rack mechanism to produce bi-directional linear motion in order to actuate a pump (a), and a practical demonstration of the pump being actuated with the mechanism attached (b, c).	106
5.14	Configuration of the gear rack mechanism to produce linear motion in order to depress the spray button (a), and a practical demonstration of the button being squeezed with the mechanism attached (b, c).	106
5.15	Overall hardware structure of Roman	107
5.16	Exploded view of the Roman gripper.	108
5.17	Meshing (a) and detaching (b) operations of Roman gripper.	109
5.18	Overview of the four types of mechanisms used to achieve target motions: spur gears for rotational motion (a), bevel gears for rotational motion at an angle (b), gear rack for linear motion (c), and a pin-in-slot mechanism to achieve periodic motion (d).	111
5.19	Breakdown of the configuration of the gearbox, showing how to produce higher speeds at the expense of torque (a) and to produce higher torque at the expense of speed (b), as well as the physical configuration of the gears (c) and the design of the gearbox (d).	112
5.20	Demonstration of the two types of periodic motion that could be accomplished with a pin-in-slot mechanism: angular (a), and up and down (b).	115
5.21	The Roman user interface used for selecting motion templates, authoring control programs, and uploading them to the robotic arm.	117
5.22	A sample motion profile used to control the wire cutter in order to cut a wire.	117

6.1	The RoCap pipeline is a robotic system designed to collect datasets for the purpose of pose estimation of appearance-changing objects, <i>e.g.</i> , a deformable plush toy (a). The system consists of a robotic arm and an RGB camera, which allows for data collection (c) of objects with appearance-changing features (b). Through data augmentation and training on off-the-shelf deep learning models using the collected data, the system can effectively estimate the pose of the plush toy during manipulation, even as it transitions through deformation (d).	127
6.2	3D reconstructed results for a transparent flask.	129
6.3	Example objects for each category that RoCap is focusing on, Viewing-angle dependent : (1) flask, (2) water bottle and, (3) pitcher, Deformable : (4) flexible frog and (5) stiff anpanman, Articulated : (6) scissors, (7) spray head and (8) clamp.	131
6.4	<i>Overview of RoCap</i> . RoCap pipeline consists of camera calibration (§6.4.1), data capturing (§6.4.2), data labeling (§6.4.3), data processing (§6.4.4) and data augmentation (§6.4.4). By training on an existing deep learning framework, RoCap achieves object segmentation, state classification and pose estimation for appearance-changing objects.	134
6.5	Illustration of the eye-to-hand camera calibration (a). The robotic arm grip a checkerboard and move to multiple positions and orientations for an accurate calibration (b).	135
6.6	Pose coverage in RoCap capturing pipeline.	136
6.7	RoCap captures the appearance-changing feature of deformable objects (a), viewing-angle dependent objects including transparent objects (b) and reflective objects (c), and objects with articulated features (d). Human operator is needed if the robotic arm is not able to change the states automatically (d).	138
6.8	Data processing of data collected in by RoCap. RoCap generates mask for each image (d) by prompting SAM with bounding box (b) and points (c).	139

6.9	Quantitative evaluation setup. The green bounding box represents the ground truth and the red bounding box represents the predicted pose.	143
6.10	Qualitative evaluation of the eight objects.	145
6.11	Failure case for a highly deformable cloth.	146

LIST OF TABLES

3.1	Robiot goes beyond prior work with active mechanisms to actuate a range of everyday objects.	24
3.2	Selected statements with survey scores, counts in each cell indicate how many participants rated their scores	42
6.1	Quantitative evaluation result. The numbers indicate the average precision at 20° azimuth error.	143

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Xiang ‘Anthony’ Chen, who introduced me into the field of Human-Computer Interaction (HCI) and has been an incredible mentor along my Ph.D. journey. Anthony has done a great job in guiding me through the process of conducting research, writing papers, and preparing for my thesis defense. I am grateful for his patience, encouragement, and support. I will not become a researcher without any of his help. I am forever grateful to him and I would like to send huge congratulations to him for being promoted to Associate Professor with tenure at UCLA.

I would like to thank my committee members, Dr. Veronica Santos and Dr. Dennis Hong, for their insightful comments and suggestions. I am grateful for their time and effort in reviewing my work and providing valuable feedback.

I would also like to thank Dr. Yang Zhang for his support and guidance. As one of the only two HCI faculty members in UCLA during my Ph.D., I had a great time collaborating with him and his students on various projects. I am grateful for his support and encouragement.

I would like to thank my labmates at UCLA HCI LAB and HiLAB for their support and friendship. I am grateful for the opportunity to work with such a talented and diverse group of people. I would like to thank Ruolin Wang for her help in my RealityPlay project. Especially, I would like to thank Xingyu Bruce Liu as we share a similar research vision and our discussions have been very inspiring. We have also started a Podcast channel. And I would like to thank Xiaoying Yang and Shipeng Liu for hosting me during my temporary homeless time (every Friday) since I moved to Irvine.

During my Ph.D., I had the chance to meet and become friends with many talented researchers and students from other institutions all over the world. First and foremost, I am grateful to the HCI community. I have the chance to not only work with a lot of talented people but also become close friends with them. Specifically, I would like to thank Jiaju Ma and Zhuohao Zhang. Although we live in three different cities and only have chances to

meet during conferences, we have been supporting each other and discussing research and life. I am grateful for their friendship and support. I would like to congratulate Zhuohao for getting the Apple AI/ML fellowship, wish him the best of luck in his future career. I would also like to thank Bryan Wang from University of Toronto. Thanks for inviting me to DGP labs giving a talk and Bryan is definitely a great thinker, and I have learnt a lot from him. I wish him best luck in his future career. I would like to thank Peilin Jiang, Yining Cao and Fuling Sun from UCSD. I have visited them several times and had a great time hanging out. I would also like to thank Mingrui Zhang from Meta NYC, Xiaofei Zhou from University of Rochester, Jiannan Li from Singapore Management University, Yudai Tanaka from University of Chicago, Liang He from Purdue University, Toby Chong and Anran Qi from University of Toronto, and many others. I am grateful for their friendship and support.

I would also like to thank my mentors during my internships, including Takeo Igarashi, Li-Yi Wei, Rubaiat Habib Kazi, Stephen DiVerdi, Tovi Grossman, Yan Xu, Stephanie Santosa, Michelle Li, Erva Ulu, Nurcan Ulu and many others. I am grateful for their guidance and support.

Lastly, I would like to thank my parents, Bing Li and Liqun Cai, for their love and support. I would not be able to achieve anything without their support. And I would like to thank my wife, Iris Huang, affectionately known as my mermaid. I truly thank her for endless support, companionship, and deep love. Words fall short of expressing my gratitude towards her, and my love for her will forever endure.

CURRICULUM VITAE

2017 – Present	Ph.D. student in Mechanical Engineering, University of California, Los Angeles.
2023	Research Intern, Meta Reality Labs, Toronto, Canada.
2022	Visiting PhD Student, University of Tokyo, Tokyo, Japan.
2021	Research Intern, Adobe Research, San Jose, CA.
2019	Research Intern, Palo Alto Research Center, Palo Alto, CA.
2013 – 2017	B.S. in Naval Architecture and Ocean Engineering, Shanghai Jiao Tong University, Shanghai, China.

PUBLICATIONS

- [1] Jiahao Li, Jeeun Kim, Xiang ‘Anthony’ Chen, “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 (UIST ’19)*. (DOI: [10.1145/3332165.3347894](https://doi.org/10.1145/3332165.3347894)) (2019), .

- [2] Jiahao Li, Meilin Cui, Jeeun Kim, Xiang ‘Anthony’ Chen, “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 (UIST ’20)*. (DOI: [10.1145/3379337.3415826](https://doi.org/10.1145/3379337.3415826)) (2020), .

- [3] Abul Al Arabi, Jiahao Li, Xiang ‘Anthony’ Chen, Jeeun Kim, “Mobiote: Augmenting Everyday Objects into Moving IoT devices using 3D Printed Attachments Generated by Demonstration”, *In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI ’22)*. (DOI: [10.1145/3491102.3517645](https://doi.org/10.1145/3491102.3517645)) (2022), .

- [4] Jiahao Li, Alexis Samoylov, Jeeun Kim, Xiang ‘Anthony’ Chen, “Roman: Making Everyday Objects Robotically Manipulable with 3D-Printable Add-on Mechanisms”, *In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI ’22)*. ([10.1145/3491102.3517645](https://doi.org/10.1145/3491102.3517645)) (2022), .
- [5] Xiaoying Yang, Jacob Sayono, Jess Xu, Jiahao Nick Li, Josiah Hester, Yang Zhang, “MiniKers: Interaction-Powered Smart Environment Automation,” , *In Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT), Volume 6 Issue 3, September*. ([DOI: 10.1145/3550287](https://doi.org/10.1145/3550287)) (2022), .
- [6] Jiahao Nick Li, Yan Xu, Tovi Grossman, Stephanie Li, Josiah Santosa, Michelle Li, “OmniActions: Predicting Digital Actions in Response to Real-World Multimodal Sensory Inputs with LLMs,” , *In Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI ’24)*. (2024), .
- [7] Xingyu Bruce Liu, Jiahao Nick Li, David Kim, Xiang ‘Anthony’ Chen, Ruofei Du, “Human I/O: Towards a Unified Approach to Detecting Situational Impairments,” , *In Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI ’24)*. (2024), .

CHAPTER 1

Introduction

1.1 From the Conventional to the Smart Realm

In an era where digital technologies become ubiquitously embedded in our daily lives, the integration of physical objects into interconnected networks has long been extensively explored in the realms of ubiquitous computing [131] and the Internet of Things (IoT) [8], enhancing user efficiency and quality of life. This fusion has blurred the boundaries between the digital and physical worlds. For example, a user might establish a weekly routine wherein automated blinds close at 10pm and open at 7am, managing lighting and privacy. Similarly, a system could automatically detect and alert if a pair of scissors is not securely stored, thereby mitigating potential risks to children.

This envisions an ecosystem where objects are not merely passive entities but actively participate in and respond to the interaction between the user and the environment. Such objects, referred to as *smart objects* [100, 216], are embedded with integrated sensors, actuators, and computational capabilities, enabling them to communicate and exchange data and to be remotely monitored and controlled. However, a significant number of legacy objects remain untouched by this technology evolution, lacking inherent actuation and sensory capabilities. Replacing these legacy objects with new, smart-capable counterparts is costly, both financially and environmentally.

Therefore, a question arises: how can we ensure a seamless transition from the conventional to the smart realm, enriching individuals' daily lives by progressively introducing smart objects into the environment? By “progressively”, I refer to two primary strategies: (i) incrementally upgrading legacy objects with smart features (*e.g.*, replacing light bulbs

with smart bulbs such as Philips Hue¹) and (ii) adding previously absent, cost-effective smart devices, such as smart speakers, which not only function as audio devices but also serve as central hubs to control smart devices.

1.2 Challenges and Research Questions in Smartening Legacy Objects with Physical Capabilities

Objects with smart capabilities are expected to perform tasks autonomously, which consists of both digital and physical tasks. Digital tasks refer to operations executed electronically, such as data communication (*e.g.*, sending room temperature to central hub) or control parameters of smart device (*e.g.*, adjusting the brightness of a light bulb). Physical tasks, on the other hand, refer to tangible actions in the real world, such as movement, actuation, or structural modifications of physical properties (*e.g.*, opening an automatic door).

While researchers and practitioners have explored various way to augment legacy objects with digital capabilities (being able to perform digital tasks), such as replacing light bulbs with smart bulbs, or attaching smart tag for digital information storage, it is still challenging to augment legacy objects with physical capabilities. Physical tasks are crucial in aiding individuals in daily life, such as enhancing human labor efficiency, or assisting people with disabilities or elderly individuals. Product have been made available to public, such as smart switches² or smart blind motors³ which can be attached to legacy objects, enabling remote control functionalities. Researchers have also explored ways to retrofitting legacy objects with external mechanisms to perform a wider range of tasks on custom objects [39, 174]. For example, a custom toaster can be augmented to autonomously toast bread to perfection every morning [174].

Efforts in this domain predominantly employ the design of mechanical mechanisms, driven

¹Philips Hue.

²Smart light switches.

³Smart blind motors.

by actuators affixed to traditional physical objects, to facilitate the execution of physical tasks. However, the current landscape of technology for augmenting legacy objects with physical capabilities remains limited. Most of the prior work focuses on simple motions on interfaces, such as the toggling motion of a switch, or the linear motion of a blind. Furthermore, the scope of prior work rarely encompasses tasks involving multiple objects, such as cooking a meal or using handheld tools. Additionally, a distinctive challenge arises in augmenting objects with physical capabilities correlated to the challenges above: the need for customizing these augmentations to adapt to custom objects. These challenges align closely with the fields of human-computer interaction (HCI) and robotics. Consequently, addressing these issues gives rise to several pertinent research questions:

RQ1: From a user standpoint, how can we enable individuals to customize augmentations to their specific legacy objects to perform desired physical tasks?

RQ2: Considering the augmentation itself, how can we design and fabricate it to execute complex tasks or those encompassing multiple interactions?

Below, I elaborate on these research questions and outline the goals for addressing them.

1.2.1 Enabling Individuals in Customizing Augmentations

User customizability is pivotal to the adoption of such augmentations. Given the wide variety of objects each with their unique form and function, a one-size-fits-all solution is not feasible. Hence, it is essential to enable users to design augmentations based on their own needs. Furthermore, as the desired motion grows in complexity, the requisite design, encompassing robotic or electronic components, may demand a higher level of expertise from the user. This presents a considerable challenge for the general public. Within the domain of HCI, researchers have endeavored to develop tools that enable end-users to harness the potential of cutting-edge technologies. These tools, tailored to distinct applications and user requirements, underscore the importance of accessibility and adaptability. Consequently, when it comes to designing augmentations to enhance physical objects with physical capabilities,

there emerges a need for novel tools that highlight user customizability.

1.2.2 Enabling Augmentations to Adapt to Complex Tasks

There are two different strategies to design augmentations such that legacy objects are able to perform physical tasks in a smart manner: *(i)* augmenting legacy objects to perform physical tasks autonomously, with embedded or add-on mechanisms, and *(ii)* augmenting legacy objects to be more amenable to be manipulated by robotic systems to perform physical tasks.

Augmenting for Self-Task Automation In designing augmentations for objects to execute tasks autonomously, the task complexity is a pivotal factor to consider. Motions extending beyond basic linear or rotary actions may be necessary, especially when tasks involve actuating an entire component of an object or engaging in more intricate manipulations. These augmentations need to navigate the nuanced requirements of various tasks, potentially demanding multi-dimensional movement, precise control, and interaction with other objects or systems. Hence, the design process must account for these complexities to ensure effective and reliable task automation.

Augmenting for Robot-Task Automation Researchers have devoted significant efforts towards developing robotic systems that emulate human interaction with everyday objects to aid individuals. In the realm, numerous open research questions (*e.g.*, robotic manipulation [18], localization [163], and navigation [139]) need to be addressed before such systems can be smoothly integrated into our daily lives to assist with physical tasks—a challenging endeavor not easily achieved in the near future. Thus, the goal of augmenting legacy objects to be more amenable to be manipulated by robotic systems is to facilitate easier manipulation and interaction of objects with the robotic system, thereby creating a broader smart environment where everything works more smoothly.

1.3 Research Goals

To address the aforementioned challenges, my thesis research aims to explore methods for integrating legacy objects into the smart realm, leveraging add-on devices and mechanical mechanisms to perform physical tasks by themselves or robotic systems.

The research goals are *(i)* enhancing user customizability, making it easier for end-users to design and fabricate their own augmentations for their custom legacy objects, and *(ii)* designing robotic augmentations, such that objects can perform physical tasks either by themselves or by robotic systems. Specifically, this thesis presents

1. Several design tools for end-users to interactively express their intent of the target task and automatically generates and fabricate the design of the augmentation.
2. Support for augmenting the legacy objects to be manipulated by robotic systems, focusing on both actuation and sensing perspectives.

The outline of this dissertation is as follows:

- **Chapter 2** discusses the background and related work in the field, including the history and challenges of smart home environments and personal robotic systems, the state-of-the-art in smartening legacy objects, and the challenges in designing augmentations for robotic systems.
- **Chapter 3** starts with addressing the research question of user customizability, discussing the design and implementation of my paper – Robiot, which is a design tool for users to design external mechanical mechanisms to for legacy everyday objects to perform physical tasks autonomously, such as adjusting the angle of a desk lamp or opening a drawer. Users can express their design intent by simply recording a short video of how they manipulate the object and the system will automatically generate the design of the mechanisms that can be attached to the objects to perform the task.

- **Chapter 4** focuses on more complex and customized physical tasks for everyday physical objects, showcasing the potential to augment these physical objects with additional functionalities. Specifically, I present Romeo, which is a design tool enabling end-users to design and fabricate physical objects with embedded transformable parts to perform augmented tasks (*e.g.*, a spatula can automatically stir the pot). Users can specify the desired motion points of the target tasks in the user interface and the system will automatically generate the printable part of the objects.
- **Chapter 5** transitions to augment legacy objects to be manipulated by robotic systems. In Roman, I identified various robotic manipulability problems that are applicable to the physical objects and proposed mechanical design guidelines which can be utilized by any robotic arm to manipulate the physical handheld objects to perform physical tasks.
- **Chapter 6:** Unlike human, robotic systems may struggle to perceive the spatial information of the target objects, which is a core component in manipulating physical objects, which is not addressed in the previous chapter. In this chapter, I present a novel pipeline – Rocap, to augment the physical objects with enhanced perceptibility for robotic systems, focusing specifically on 6-DoF pose estimation. Data essential for training a deep learning model is collected through the Rocap pipeline, enabling the augmentation of physical objects traditionally deemed challenging for pose estimation (*e.g.*, deformable, transparent, or articulated objects). These objects can be augmented to be tracked via an RGB camera, establishing a crucial foundation for robotic systems to accurately manipulate the target objects.
- **Chapter 7** concludes this thesis by reflecting on the contributions and limitations of this work, and discussing future directions.

CHAPTER 2

Background and Literature

Prior to delving into the details of my work on smartening legacy objects, I first provide a contextual background through an overview of the related technology and state-of-the-art research in the related field. To start with, I discuss the historical evolution of smart homes, spotlighting cutting-edge products and technologies that have significantly impacted this domain.

Alongside the discussion on the general smart home landscape, I also delve into the academic research. The spectrum of the discussion ranges from research in smartening legacy objects to explorations in personal robotics aimed at executing physical tasks for users. Lastly, given that my research tackles specific challenges inherent in smartening legacy objects, such as enhancing user customizability for design and personal fabrication, enabling machine manipulability and vision, I also discuss related research in these areas.

2.1 History of Smart Home and Related Technologies

The concept of smart home has been around for decades, embodying the integration of common devices that control features of the home, such as environmental systems including lighting and heating [178]. Over time, the evolution of smart homes represents the convergence of technological innovations and changing societal needs. In this section, I discuss the historical journey of smart homes and the development of their associated technologies.

2.1.1 Early Explorations of Home Automation

ECHO IV (1966) ECHO IV¹, is a prototype of a home computer developed by Westinghouse Electric in the mid 1960s. While it is not technically a smart home system, it is recognized as one of the first computer systems designed for home use. It was designed to control home temperature, automate appliances and even managing household inventory. This system was working in the creator's house until 1976.

X10 (1975) Features the Beginning of Home Networks X10², a pioneering protocol for home automation developed in 1975, facilitated communication between compatible devices over existing electrical wiring in the home, allowing for remote control of devices from a central system. Released to consumers in the late 1970s, X10 was an early indicator of the potential for integrated home automation.

The Echelon Corporation's LonWorks (1990) Echelon's LonWorks platform³, an early multi-device communication platform, allowed home devices to interface and be governed from a single user point, advancing the pursuit of comprehensive home automation systems.

2.1.2 The Explosion of Modern Smart Homes

The advent of wireless technologies opens a new era in smart home capabilities, obviating the need for cables to connect different devices. This wireless revolution significantly streamlined the installation and expansion of smart home systems, making them more accessible and appealing to a broader demographic. Also, the evolution of the internet played an important role in connecting various devices to the internet, allowing for remote control and monitoring functionalities.

¹<https://en.wikipedia.org/wiki/ECHOIV>

²[https://en.wikipedia.org/wiki/X10\(industrystandard\)](https://en.wikipedia.org/wiki/X10(industrystandard))

³<https://en.wikipedia.org/wiki/LonWorks>

In the last decade, the smart home market has seen a surge of innovative products. Examples include lighting systems like Philips Hue⁴, which allows users to control lighting colors and schedules through their smartphones; thermostats like Nest⁵, that learn your preferences over time and adjust the heating or cooling to save energy while keeping you comfortable; security systems like Ring⁶, which provide video surveillance and alerts directly to your smartphone; and smart refrigerators that can keep track of expiry dates, suggest recipes based on the items inside, or even allow you to order groceries directly from the touch screen on the door.

More recently, consequent to the proliferation of smart devices and associated protocols, the market witnessed the emergence of numerous home automation hubs. Examples include voice-controlled devices such as Amazon Echo⁷ and Google Home⁸ simplified user interaction with connected devices and enabled the integration of third-party devices into a single system.

2.1.3 The Gap of the Current Smart Home Technology

Despite the rapid growth of the smart home market, notable gaps persist within the current smart home technology landscape. As previously mentioned, the overarching goal of smart home technology is to enrich the home experience including promoting efficiency, energy conservation, and safety. This entails empowering smart objects within the home to perform both digital and physical tasks on behalf of the users.

Costly to replace. Transitioning from conventional to smart homes requires not only the integration of new smart devices, but also replacement of existing legacy objects with smart-enabled counterparts. Unlike merely adding a smart bulb or a smart plug, which are

⁴<https://www.philips-hue.com/en-us>

⁵<https://home.nest.com/>

⁶<https://ring.com/>

⁷<https://www.amazon.com/Echo-4th-Gen/dp/B07XKF5RM3>

⁸<https://home.google.com/welcome/>

relatively inexpensive, replacing major household items like appliances or heating/cooling systems can be prohibitively expensive. Moreover, many legacy objects are still in good condition and replacing them would not be cost-effective or environmentally sustainable. This highlights the need of developing solutions that can smarten legacy objects in a cost-effective and sustainable manner.

Insufficient support for physical tasks. Presently, existing technology predominantly addresses the digital realm (*e.g.*, enabling users to remotely control the brightness of lights or adjust the thermostat). However, the domain of physical tasks remains largely unaddressed. Although there are products like automatic doors and smart blinds that cater to the physical domain, a myriad of other physical tasks are yet untouched by smart technology. This is primarily due to the inherent complexity associated with physical tasks as opposed to digital tasks. While digital tasks often involve wireless signaling, physical tasks necessitate the perception, manipulation, and precise control within the physical world. Addressing these physical tasks is particularly crucial for certain user demographics, such as individuals with motor impairments or those in situational impairments (*e.g.*, scenarios where hands are occupied and unable to open a drawer or door). Hence, there is a compelling need to extend smart home technology to encompass a broader range of physical tasks, promoting a more inclusive, adaptable, and holistic smart home environment.

The goal of this thesis aims to address the aforementioned gap in the current smart home technology by developing novel approaches to smarten legacy objects with the support to perform physical tasks for users. In the following section, the discussion transitions towards the research trends in order to address the aforementioned gaps in the current smart home technology.

2.2 Research Trends in Smartening Legacy Objects

Researchers have been investigating methods to augment legacy objects with the ability to perform both digital and physical tasks. As previously mentioned, smart objects should possess the capabilities to be monitored, actuated, and connected to a central platform, enabling data exchange with other devices. Smartening objects stands in the intersection of multiple research domains including computer vision, embedded system, mechanical engineering, robotics and human-computer interaction. My research primarily focuses on the intersection of human-computer interaction (HCI) and robotics, specifically, smartening objects with robotic augmentations to enhance the sensing and acutation capabilities of legacy objects.

Within this scope, several research areas closely align with the goal including *(i)* developing general-purpose personal robots to assist with a variety of tasks and *(ii)* developing methods to augment physical objects to perform physical tasks automatically.

2.2.1 General-Purpose Personal Robots for Home Automation

Application perspective. From the application perspective, researchers in personal robotics have sought to democratize robots to assist people with a variety of different applications of physical tasks. Generally, employing robots to assist humans in daily tasks can enhance user efficiency. In personal environments, they can aid with office tasks [160], and perform household chores such as cooking [197] and cleaning [58]. In social environments, robots can assist with navigating public spaces [193] or delivering food (e.g., Starship delivery robots⁹).

Furthermore, the presence of such robots proves to be more crucial for other user groups, particularly individuals with motor impairments or elderly people, who may face challenges in performing physical tasks. For example, robots can assist users in different tasks such as fetching an objects from afar [57, 108, 147], opening a medicine bottle for users with limited dexterity [101, 51] or even full-body activities such as assistance with bathing [135].

⁹<https://www.starship.xyz/>

Technical perspective. The development of the personal robotic systems largely rely on the cutting-edge technology within the field of robotics, in a smaller scale, encompassing facets such as robotic manipulation, perception, control and locomotion. These represent ongoing research domains in the broader field of robotics, with a myriad of work continually emerging. To offer a more insightful overview of the discourse in this dissertation, I briefly delve into the related work in robotic manipulation, given its closer relevance to this thesis centered on enhancing legacy objects to execute physical tasks.

Recent advancements in robotic manipulation have primarily centered around enhancing either the ‘brain’ or the ‘limbs’ of robots. In the quest for a better ‘brain’, researchers have ventured into various directions, such as employing machine learning to determine the optimal gripping position of novel objects [98, 187, 188, 86], or augment robots’ perception of the environment and objects through advanced sensing techniques [30, 212, 32]. On the other side, to develop better ‘limbs’ for robots, researchers have been exploring methods of enhancing robotic dexterity through novel design of robotic hands [170, 171, 145, 224] or different types of robotic end-effectors [148, 227, 226, 180]. Advanced control mechanisms are also crucial for robots to perform robust and smooth manipulation of physical objects [168, 18].

The discussed research focuses on advancing the technology of robots in order to perform physical tasks involving physical objects. Instead of adhering to a one-directional approach—waiting for robots to be advanced enough to perform physical tasks—it’s plausible to adopt a bi-directional strategy, in which physical objects can be enhanced to play a contributory role in actualizing smart home automation. A substantial body of research in HCI has been dedicated to enabling the sensing and actuation of physical objects, which I discuss in the following sections.

2.2.2 Augmenting Physical Objects

Augmenting physical objects encompasses two different aspects: *(i)* enhancing the objects to be more sensible allowing for improved monitoring and control, and *(ii)* enhancing the

objects to be more actuable such that the objects can autonomously execute physical tasks and interact with its environment in more dynamic ways.

Sensing of Physical Objects One of the key features of smart objects is their ability to be monitored and remotely controlled. Towards this end, research has ventured into enabling the sensing of physical objects. The focal points of sensing encompass activities and events, spatial information, and the state of the objects. Given that most legacy objects lack embedded sensors to relay the aforementioned information, researchers have probed into diverse methods of employing various sensors to facilitate the sensing of physical objects.

One typical type of sensor is **vision-based sensors**. Camera is one of the most commonly used vision-based sensors as it can provide wide-area and versatile sensing capabilities. For example, researchers can extract spatial information of the objects leveraging visible markers such as fiducial markers [62, 146, 207], optical tracking systems [230, 177] or via computer vision techniques such as color tracking [198], feature matching [15] or point cloud alignment [75]. However, owing to challenges like occlusion and potential privacy concerns, researchers have ventured into exploring other types of sensors to achieve always-on sensing of physical objects.

The sensor technologies in focus include **wireless sensors**, which are apt for monitoring object states and detecting events and activities. These methodologies frequently harness machine learning techniques, which can be trained to recognize various activities while preserving user privacy. For instance, synthetic sensors exemplify a typical general-purpose sensing system that processes multiple raw sensing signals (*e.g.*, accelerometers, microphone, EMI) and possesses the capability to detect various activities of physical objects within a smart home setting [111]. This concept has been further expanded upon by various works within the HCI domain [228, 110, 48, 229], where specific sensor data is employed to afford higher-fidelity sensing of physical objects (*e.g.*, detecting motion status of physical object using RF signals [221]).

Actuating Physical Objects Different from personal robots, which leverages mobile robots or robotic arms to execute physical tasks for users, systems designed to actuate physical objects for purposes of remote control or user assistance embed robotic mechanisms into physical objects. This setup allows these objects to autonomously perform tasks on behalf of users without the necessity of intermediary devices or platforms.

Products like Clicbot¹⁰, Smartians¹¹, and Microbot-push¹² employ retrofitting devices equipped with gear mechanisms to facilitate the remote actuation of physical objects, thereby performing tasks on behalf of users. In the research domain, researchers have developed various systems featuring robotic mechanisms to retrofit legacy objects and enable their actuation. The idea of ‘mechanical hijacking’ was first demonstrated by Davidoff *et al.* as it uses LEGO Mindstorms¹³ as the building blocks to mechanically control physical objects such as volume knobs [39]. This concept was later extended by other researchers in HCI. For example, RetroFab [174], which offers an authoring tool to scan an existing physical interface and automate its controls by adding external mechanical and electronic components. IoTIZER [31] and Mobiot [5] offer various mechanisms to augment legacy objects while considering different factors such as aesthetics and mobile capabilities.

Most of the aforementioned work employs relatively straightforward mechanical mechanisms to actuate the motion of physical objects, such as gear-based or linkage-based mechanisms. The rapid advancements in 3D printing technology have significantly facilitated the easy fabrication and customization of these mechanical mechanisms to accommodate various shapes of physical objects. However, the design of even such simple mechanisms necessitates a certain level of expertise. This includes selecting the appropriate mechanism, designing fasteners to securely attach to the object [103], among other considerations. Moreover, current work still lacks the ability to generate mechanisms for more complex motion, as most of

¹⁰<https://keyirobot.com/>

¹¹<https://www.frolicstudio.com/portfolio/smartians>

¹²<https://keymitt.com/products/microbot-push>

¹³LEGO Mindstorms

the target retrofitting is limited to 1-DoF actions, such as toggling a light switch or turning the knob of an oven.

Part of my my thesis situated within this realm of work, specifically focusing on the generation of mechanical mechanisms affixed to custom legacy objects, while dedicating efforts to address the challenges of user customizability and motion complexity. In the following sections, I discuss from the HCI perspective how researchers have been working on to enhance the user customizability and address the motion complexity in physical object actuation.

2.3 Democratizing the Smartening of Legacy Objects

As previously outlined, enhancing legacy objects to imbue them with smart capabilities presents numerous hurdles in democratizing the process for a broader user base. This has been a major focus in HCI research—democratizing technology to ensure that end-users, regardless of their expertise in the target domain, can still leverage and benefit from the technology itself. In the context of smartening legacy objects, the challenges arise from the complex nature of mechanical design, personal fabrication and robotic motion planning. In the following, I discuss the related research in *(i)* enabling general users to create custom design of personal fabrication and *(ii)* interactive design tool in the domain of generating robotic motion.

2.3.1 Design Tools for Personal Fabrication

The advent of low-cost fabrication technology has facilitated personal fabrication, enabling users to create custom designs for their own use. While the design process retains a level of expertise requirement, researchers within the HCI domain have delved into approaches of empowering general users to create custom designs. This is particularly relevant to smartening legacy objects as it involves the creation of 3D printable mechanism affixed to custom legacy objects. For example, researchers have developed various tools to enable users to

create 3D printed mechanism via interactive interfaces, such as 3D printed springs [72, 73], laser-cut mechanisms [182, 113], hydraulic devices [219] and mechanisms to interact with everyday objects [202]. Other research has explored diverse forms of support such as enabling users to directly manipulate a mobile 3D printer for printing purposes [199], or incorporating augmented reality (AR) in the printing process [166]. The work above lowers the entry barrier for general users to create custom designs for personal fabrication.

2.3.2 Design Tools for Robotic Motion

Another challenge for general users is to design the robotic motion for the smartening mechanism. Robotic motion planning is complex, primarily because users typically do not program the motion directly; rather, this motion is controlled by actuators, often motors, which makes the process non-intuitive. The task demands an understanding of necessary mathematical concepts such as forward and inverse kinematics, which are challenging to grasp and solve, thereby elevating the difficulty of motion design for users without a technical background.

Researchers have looked into computational methods to interactively design robotic motion. For example, Ha *et al.* developed a design tool to enable the robotic devices (*e.g.*, length of links) by directly specifying the desired motion [68]. Megaro *et al.* presented a design system that automates the tedious process of creating 3D-printable robotic creature while allowing for casual users [140] and Geppeto is an interactive system that allows users to design expressive robots by editing complex parameters in a semantic level [43]. Part of my thesis work extends this line of research by enabling users to specify the high level motion of the mechanisms and generating the 3D printable components automatically.

2.4 Summary of the Related Work

In this chapter, I begin by delving into the history of smart homes and identifying the gaps present in current technology. The primary gaps my thesis intends to bridge are twofold: (*i*) enabling physical objects to perform physical tasks for users, and (*ii*) democratizing the

smartening of legacy objects, allowing general users to design and fabricate their own augmentations. Subsequently, I shift focus to discuss the state-of-the-art research in robotics aimed at achieving the sensing and manipulation of physical objects. Conversely, research has also dedicated to augmenting physical objects to enhance their sensing and actuation capabilities, a realm where significant contributions have been made by the Human-Computer Interaction (HCI) community. Lastly, I provide an overview of how researchers are striving to democratize the design process in personal fabrication and robotic design, making these domains more accessible to the general populace. The next few chapters unfold a presentation of four projects that encompass these two themes, starting from embedding actuation into legacy physical objects to augmenting the robotic manipulability and machine vision of legacy physical objects.

Part I

CHAPTER 3

Enabling Users to Design and Generate Add-on Mechanisms to Actuate Legacy Physical Objects

As mentioned in the previous chapters, there remains a lack of support for *physical* tasks in the current development IoT, *e.g.*, adjusting the angles of a desk lamp for optimal brightness as you perform a soldering task. Automating physical tasks has important implications for people with a disability or in a situational impairment. For example, turning on a manual faucet without touching it becomes useful when both of your hands are dirty, opening a pantry is helpful when you are holding bags of groceries, and it is convenient to have a window that closes by itself when sensing the temperature drops.

The recent development of actuatable appliances (*e.g.*, smart blinds [151], switches [109], and TV deck [123]) and reconfigurable furniture [47] suggests a future of ubiquitous robotic things: akin to how ubiquitous computing imagined a world of omnipresent computational power, we can envision a future of everyday objects and appliances equipped with robotic capabilities to interactively carry out a series of complex actions. With such physical tasks and dynamic actions that assist people in a variety of contexts, the future of everyday robotic IoT devices can open a door for a new world of everyday interactive systems.

To bridge us to this vision, instead of waiting to replace and upgrade a whole world of legacy static objects with fully automation-intended gadgets, one reasonable first step is to come up with solutions that allow end-users to augment these objects with actuatable behaviors. Leveraging a democratization of robotic kits and rapid prototyping machines, prior work has proposed the design of external [210, 114] or add-on [173, 195] actuation mechanisms to operate a physical control interface. To generalize the approach to a wider

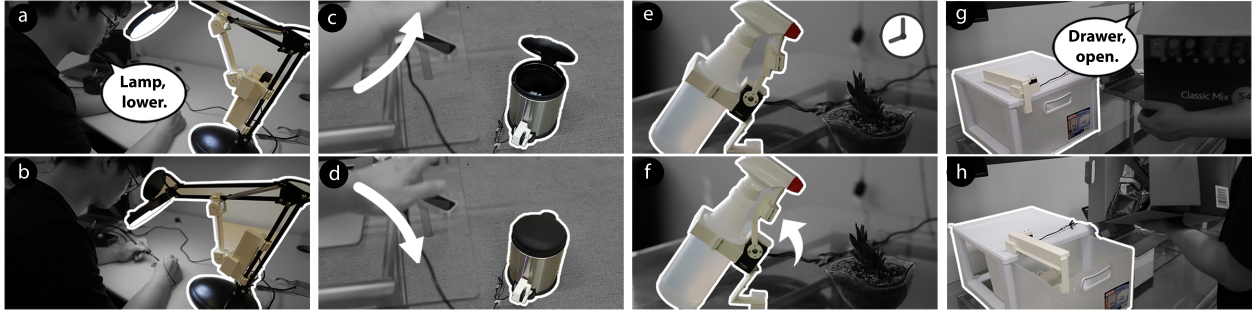


Figure 3.1: Robiot enables end-users to take a video and automatically generates mechanisms that can actuate everyday objects to perform simple physical tasks. All the four examples were designed by participants in our design session: lowering a lamp while busy soldering (a), waving a hand up to open a trashcan afar (c), setting up a scheduled house plant water spraying system (e), and asking the drawer to open itself when having no extra hand (g).

range of everyday things, past research primarily focuses on generating *passive* adaptations to reduce the effort of operating handheld objects [28]. In contrast, *active* mechanisms on everyday objects remains a nascent topic—very little is known how average users can create actuation mechanisms with desired motion to automate everyday objects for even the simplest physical tasks.

In this chapter, I present the design and implementation of Robiot — a design-by-demonstration tool for end-users to fabricate add-on mechanisms to actuate legacy static objects for everyday physical tasks. Using Robiot, users only need to take a short video manipulating an object to demonstrate an intended physical behavior, such as lowering a desk lamp, squeezing a spray bottle, or opening a drawer. To generate the actuation mechanisms, Robiot extracts two requisite parameters: (i) first it performs a scene and motion analysis in the video domain to infer the **type of joint** corresponding to the demonstrated motion; (ii) then it retrieves the object’s 3D model from a pre-constructed repository and identifies **maneuverable/ground parts**—where on the object to ground a mechanism and where to let it exert the actuation. These two parameters lead to a set of candidate mechanism designs, which can be further filtered by adjusting several design parameters, *e.g.*, range of motion, torque and speed. Figure 5.1 shows several of exemplary applications designed using

Robiot, fabricated and installed by the study participants. In an hour-long design session, six participants successfully designed 14 mechanisms. After 3D printing, they were able to assemble and install mechanisms on existing objects with instructions given by the system, finding its usefulness for future use in their environment to actuate own things.

The main contribution of this chapter is an end-to-end pipeline that requires very minimal user input to automatically generate 3D printable actuation mechanisms by a novel combination of 2D video analysis and 3D geometry processing. Although I demonstrated 3D printing as the main fabrication technique, the pipeline’s ability to extract mechanically meaningful information from user demonstration can potentially generalize to other manufacturing techniques.

3.1 Related Work

In this section, I discuss the prior related to the work presented in this chapter more thoroughly. Robiot provides end-users with a design tool that can create mechanisms to attach to and actuate existing everyday objects for simple physical tasks. This goal cross-cuts three areas of prior work: *(i)* personal robots that interact with and assist people with their physical tasks; *(ii)* reality-based design tools that extract information from the real world to create designs that can in turns augment the real world; and *(iii)* computational methods of designing and prototyping functional objects.

3.1.1 Personal Robotics

Research in personal robotics has sought to democratize robots to assist people with a range of physical tasks in home and offices. Designing robots with motion enables users to communicate and dynamically engage with robots, such as having a robot aid a user’s office tasks [78]. Further, to consider a broader audience, personal robots are taking an increasingly important role in enabling people with disabilities [19].

At a smaller scale, robots can perform grasping and manipulation of everyday objects.

Grasping and manipulation remains a fundamental challenge that drives new algorithmic and system designs [184]. At the application level, grasping and manipulation also enables robots to assist people’s daily living tasks, from fetching a mug [209] to taking medications [165]. On the other hand, robots can now support people’s whole-body activity in a large scale, such as a soft assistive robot that provides scaffolding and protection for elderly people when they take a bath [134]. Robot assistance is also frequently used in navigation, from orienting oneself in a work area [201] to maneuvering in a shopping mall [64].

Despite such development, the range of physical tasks a personal robot can handle remains limited due to the large variety of tasks as well as the different objects involved. To make robot’s ability flexible and scalable, prior work has been focusing on using programming by demonstration to teach robot custom behaviors [52, 17, 214]. Although in this way the physical tasks become programmable to a robot, the performance is by and large determined by the one-size-fits-all design of the robot that is available. Worse, at times such a robot might not even be available to perform a bespoke task. To overcome this limit, prior work such as Coros et al.’s provides an expressive means for specifying robotic behaviors by sketching a user-defined motion path, which generates a corresponding linkage design for an automata [36]. Instead of sketching, our design tool allows a user to directly manipulate an existing object to specify a desired physical behavior, and then automatically generates a behavior-specific robotic component that can retrofit to and actuate everyday objects to assist users in automating simple physical tasks.

3.1.2 Reality-based Design Tools

Related to enabling design of actuating physical tasks, past research has explored a class of reality-based design tools that address two important issues: *(i)* how to extract information from the physical world that can lead to *(ii)* creating solutions that can leverage personal fabrication to augment the physical world.

The need to extract information from the physical world coincides with many research goals of Augmented Reality (AR). Early work such as DART allows designers to rapidly

transition storyboards to a work experience based on a camera view and an live AR authoring environment [132] A recent surge of AR technologies gives rise to prototyping tools that incorporate real-world information in the design process, such as directly viewing, positioning and iterating a 2D sketch in a 3D real environment [6, 155, 211, 95, 166].

Obtaining an understanding of the physical world informs new ideas of augmentation, often manifested as the idea of ‘mechanical hijacking’ first demonstrated by Davidoff et al. [39] and later extended by Chen et al. in adapting hand-operated objects for easier manipulation [28]. AutoConnect enables the automatic generation of structures that connect (i.e., holding in place) two user-selected objects based on their scanned digital representations [104]. Printy allows novice users to fabricate fully-functional internet-connected object [7]. Patching provides a hybrid platform that scans, mills, and additively fabricate new components to replace part of an existing object with augmented functionality [149]. Facade uses the crowd to annotate a visually inaccessible physical interface (e.g., buttons without tactile feedback) and generates 3D printable tactile overlay to assist visually-impaired people to use these interfaces [66]. Perhaps the most related to our work is RetroFab, which offers an authoring tool to scan an existing physical interface and automate its controls by adding an enclosure consisting of mechanical and electronic devices [173]. However, Retrofab only addresses actuation specific to operating physical controls and does not consider a more general way to encompass motions from various other everyday objects.

To summarize, as shown in Table 3.1, the most related work in reality-based design tools either focuses on physical interfaces [66, 173], or only addresses everyday objects with passive add-on components [28]. Robiot complements existing research with a generative pipeline to create active actuation mechanisms on everyday objects.

3.1.3 Designing and Prototyping Functional Objects

The eventual goal of our tool is to generate fabrication-ready actuation mechanisms. To achieve such functional design, prior work has demonstrated two approaches: assembly-based and generative design.

Table 3.1: Robiot goes beyond prior work with active mechanisms to actuate a range of everyday objects.

Physical Interfaces → Everyday Objects		
Passive	Facade [66]	Reprise [28]
Active	Retrofab [173]	Robiot

Assembly-based solutions allow users to put together off-the-shelf components for a functional, often complex object. For example, consider a plethora of robotic kits, such as the popular LEGO Mindstorms [114] used in early work of mechanically ‘hijacking’ the control of physical devices [39]. TrussFormer enables users to 3D print large-scale kinetic structures [102]. Zykov et al.’s Molecubes is an open-source modular robotics kit that provides a low-cost, ruggedized and expandable platform with software support for visual and control design [235]. Schweikardt and Gross demonstrated the expressiveness of roBlocks—a reconfigurable modular robotic prototyping tool where small, magnetic, heterogeneous components can be snapped together to create large and complex constructs [192]. Grafter largely automates the process of extracting and recombining mechanical elements from 3D printed machines and affords extracting groups of mechanical elements that already work together, such as axles and their bearings or pairs of gears [183].

Generative design allows users to specify their high-level design goals while leaving the low-level functional considerations to a generative algorithmic process. Autodesk’s Project DreamCatcher takes a data-driven approach to generate hundreds of thousands of design options based on input functional requirements [176]. To explore the many generated design alternatives, DreamLens provides a visualization platform built for exploring large-scale design datasets [138]. DreamSketch allows a user to integrate generatively designed components with the workflow of sketching [92]. To further incorporate users’ intents, Forte loops user input into the optimization process to create structures that not only meet functional re-

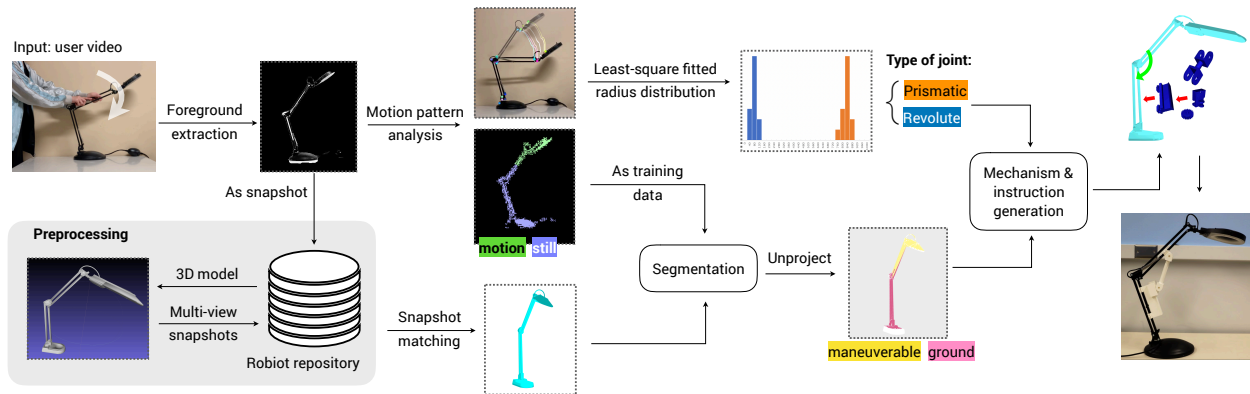


Figure 3.2: Overview of Robiot’s end-to-end pipeline for generating actuation mechanisms from a user’s input video.

quirement but also mimic users’ sketches [29]. Generative design also enables end-user design and fabrication of robots, from creating linkages to exhibit specific motion path [12], to the automation of a comprehensive set of design considerations, including different morphology, proportions, gait and motions [141].

It is also possible to take a hybrid approach that combines existing components as well as a generative process. Desai et al. propose an assembly-aware design pipeline that automatically lays out user-defined electromechanical components and creates 3D printable enclosures to assemble the robot [44]. Robiot employs a hybrid approach: the mechanisms design is generated based on the intrinsic geometry of the object as well as the extrinsic motion demonstrated by the user; then these generated components are assembled together with existing parts (eg/ motor) and installed on the object.

3.2 Robiot: Mechanisms to Actuate Everyday Things

3.2.1 Overview of Robiot’s Workflow

As shown in Figure 3.2, our technical contribution is an end-to-end pipeline that requires minimum user input to generate 3D printable mechanism, which actuates legacy static objects. Using Robiot, a user simply takes a short video demonstrating how they want an

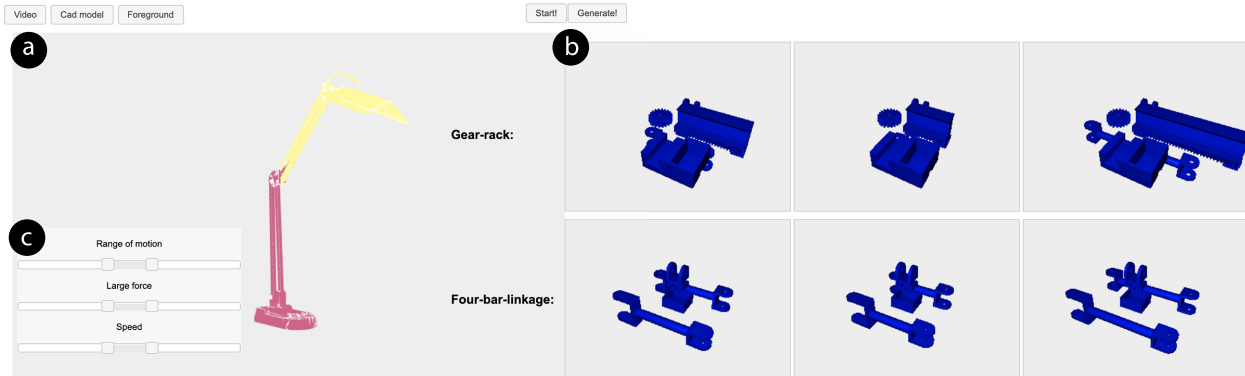


Figure 3.3: A screenshot of the user interface provided to the user once Robiot generated a set of candidate mechanisms from a given video demonstration: the corresponding 3D model (a), the generated mechanisms (b), and sliders to filter the list by range of motion, torque and speed (c).

object in motion. A wide range of these rigid-body objects' motion can be expressed as linear or rotational [28]. Thus to generate the enabling mechanisms, Robiot models the actuated components in two types of *joints*: prismatic (linear) and revolute (rotational). As shown in Figure 3.2, Robiot performs a scene and motion analysis of the input video to extract the following information.

Type of joint - The object is extracted from the video and an optical flow technique analyzes the motion, which is used to classify whether the motion is linear (prismatic) or rotational (revolute). This information leads to specific mechanisms that can actuate the object to behave as the user demonstrates in the video.

Maneuverable vs. Ground parts - First a 3D model is retrieved from an existing repository that contains preprocessed information for matching which 3D model best corresponds to the object as viewed in motion. Further, results from the above optical analysis are used to segment the 3D model into maneuverable and ground parts. Robiot then automatically generates a list of possible mechanism designs that also raise implicit constraints inferred from the input video, e.g., the size of the object's components, the required minimal torque. As shown in Figure 3.3, Robiot also provides more advanced features that allow a user to

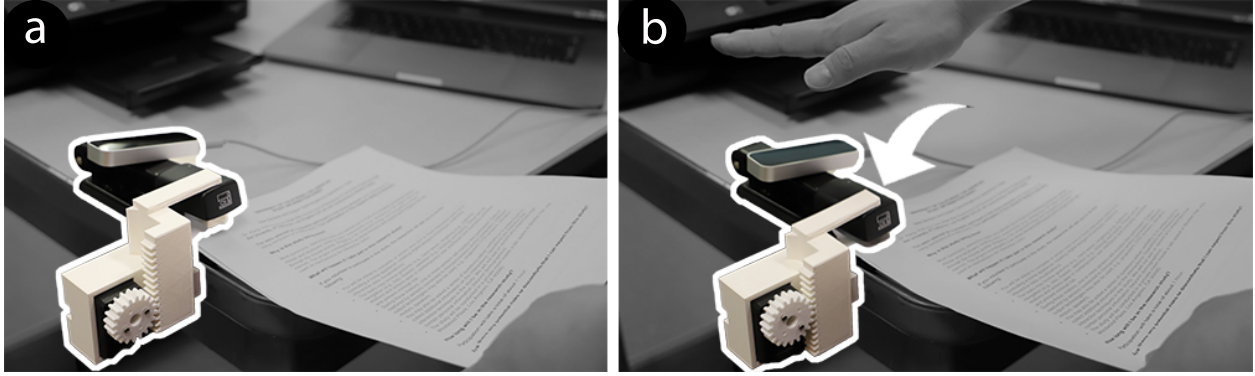


Figure 3.4: Automatic stapler is one common office appliance, that Robiot can robotize from a cheap manual stapler

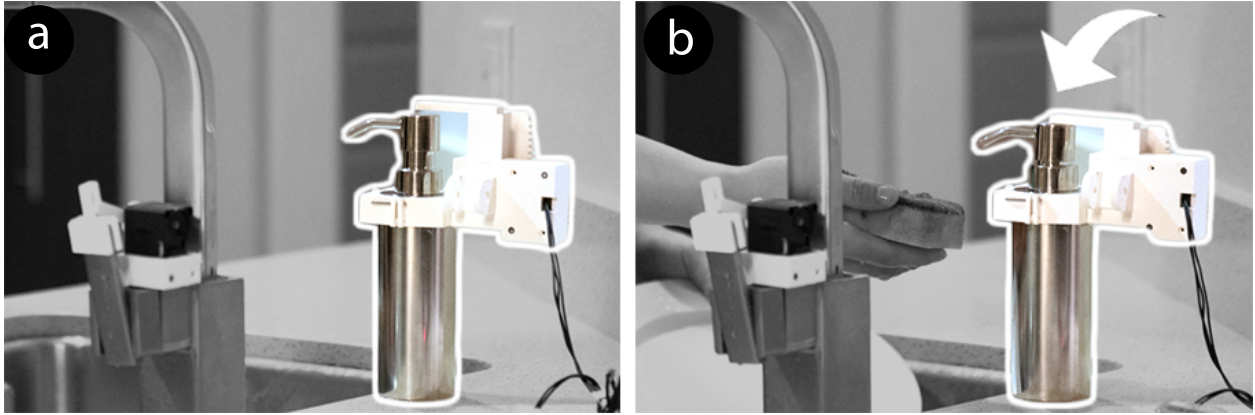


Figure 3.5: Robiot mechanism attached on top of a soap bottle performs pressing to squeeze liquids without touching

tweak and filter design options by adjusting preferred range of motion, torque, and speed.

Below we first showcase a series of examples designed using Robiot’s workflow while leaving the technical details later in following sections.

3.2.2 Examples Generated by Robiot

We present a series of examples created by Robiot, which automatically generates the 3D models of the mechanisms from a user input video. As we focus on the design tool, all the subsequent interaction was developed ad hoc (*e.g.*, using commercially available voice or

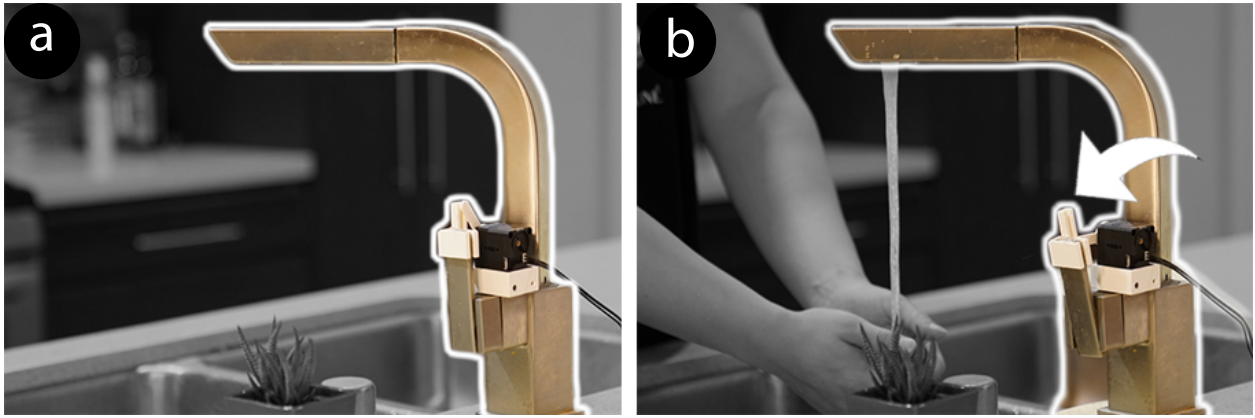


Figure 3.6: A manual faucet can automatically turn to release water by attaching a mechanism to pull the handle.

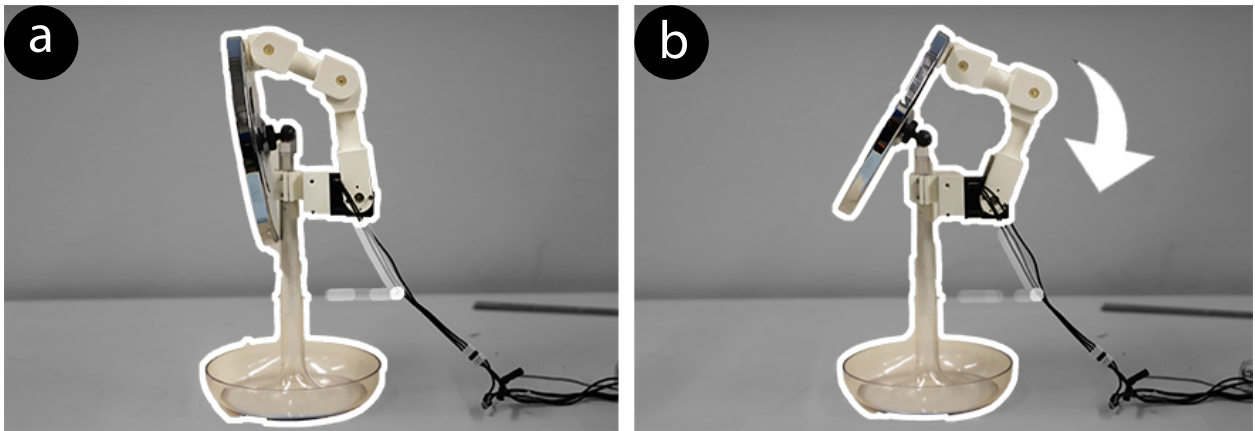


Figure 3.7: Automatically adjusting mirror can aid a user who wears make-ups and contact lens using both hands

gesture sensing input devices) as a way to demonstrate Robiot’s potential to perform simple physical tasks for users.

Figure 5.1 showcases exemplary applications with mechanisms installed to everyday objects at home and offices. When both of user’s hands are occupied and tied to a task, such as for soldering, an automatically adjustable lamp with a reading lens can come closer to him, assisting his delicate task such as soldering (Figure 5.1a,b). In the similar vein, as shown in Figure 5.1, when user’s hands are full with garbage grabbed from a counter top (c) and a heavy box with clutter (g), automatically opening trash can (d) and drawer (h) become convenient for her to ease the cleaning task. For a busy office worker who is likely to forget to water the plant regularly, a squeezer mechanism attached to a spray bottle will do the watering tasks according to a predefined schedule (e-f).

Figure 4-7 further demonstrate a wider range of example applications with actuation mechanism attachments. There exist many commercial automated staplers, which we can replicate using a Robiot-created mechanism that performs the same task using a cheap stapler (Figure 3.4). When the user’s both hands are dirty, a soap bottle that automatically presses the top to squeeze liquid soap (Figure 3.5) on her hand and a water tap turning on water help those who do not want to spread the mess (Figure 3.6). As another example, wearing make-ups is one of the most complex tasks, from which people can benefit from a mirror with automatable angle adjustment so that its usage becomes hands-free.

3.3 Implementation

In this section, we detail step by step process of creating a mechanism, from user input to ready-to-print 3D model for end users to assemble and install with the instruction.

3.3.1 Preprocessing

Robiot’s analysis of real-world objects starts from a repository of 3D models corresponding to it. As opposed to 3D scanning of the object, the models in the repository offer clearer and

better-defined mesh information than the currently-limited scanned data that often requires additional post-scan processing. Such a repository can also be populated with manufacturers cataloging the CAD models at design time and 3rd party dataset. However, the advancement in scanning technology might soon provide a viable alternative in lieu of a repository.

As shown in Figure 3.2, for each object, Robiot performs a one-time preprocessing step by taking snapshots of the 3D model at a set of predefined locations spherically around the object. Snapshots are stored together with the object’s 3D model and will be used for retrieving the 3D model as detailed below.

3.3.2 #1 Extracting an object’s 2D and 3D representations

The input of Robiot’s generative pipeline is a video of a user manually manipulating an object to demonstrate the action, expected to be produced by some mechanisms. Our first step is to extract the object’s 2D and 3D representations from the video (based on the preprocessed 3D repository).

As shown in Figure 3.2, after identifying the first stable frame we perform a scene analysis—a foreground extraction to obtain the object’s 2D representation as a binary mask M_{video} (with the 1’s representing the object and 0’s the background). We implement this step using GrabCut [181], although other methods (e.g., deep learning based direct object segmentation [107]) can also be used to replace this component in Robiot’s pipeline.

Next, we use M_{video} to retrieve a 3D model from the Robiot repository by snapshot matching, *i.e.*, finding a 3D model that has a snapshot that best matches M_{video} . For each 3D model, we perform a stepwise searching process. Specifically, for each snapshot we first binarize it into $M_{snapshot}$ and scale it to match the aspect ratio of M_{video} . We then measure how well $M_{snapshot}$ matches M_{video} by computing a matching score:

$$s_{matching} = \frac{sum(M_{video} \wedge M_{snapshot})}{\sqrt{sum(M_{video}) \cdot sum(M_{snapshot})}} \quad (3.1)$$

We identify the 3D model that has the highest $s_{matching}$ as the object’s 3D representation;

we also save the corresponding $M_{snapshot}$ for latter processing.

3.3.3 #2 Determining the type of joint

Once the area containing the object is extracted, the next step is to perform a video-based motion pattern analysis. First we extract feature points using the Shi-Tomasi corner detector [87], and then use the Lucas-Kanade method [129] to calculate the optical flow of these feature points during the course of the video. As shown in Figure 3.8, each feature point is ‘tracked’ frame by frame, resulting in a 2D trajectory comprised of an array of X/Y coordinates. Next we filter out noises and jitters by setting an empirically defined threshold to cut off feature points whose trajectory coordinates with a low covariance.

Now that we have collected trajectories representing how the object should be actuated, the next step is to determine whether such motion can be enabled using a revolute or prismatic joint. For each trajectory, we use a least-square method to fit it to an arc. The rationale is that if the actuation is revolute (rotational), the pivot should physically be part of the object thus the fitted radius must be significantly smaller than the fitted radius of a prismatic actuation. Now we compare the radius and the distance between two states—initial vs. final. Note that without occlusion, the initial/final state can be robustly extracted from the first/last frame of the demonstration video, respectively. As the length of an arc is approximately equal to $len = R \sin \alpha$ (where α is the rotation angle) if $len < \frac{R}{4}$, resulting in $\alpha < 15^\circ$, the joint is regarded as prismatic joint, otherwise it is regarded as revolute joint. As shown in Figure 3.8, we compute a distribution of the fitted radii from all the trajectories, which exhibit a clear separation between the two types of joint.

One challenge here is that some objects might have very few ‘sharp corners’ that can be used as feature points for the optical flow analysis. To address this, we incorporate the user’s hand, which provides ample feature points. As shown in figure 3.10, as the hand is used to manipulate the object, its motion must match that of the object’s. Thus we use the hand as a supplement when there is a lack of feature points detectable from the target object. We detect the hand’s position using a skin color based method [175].

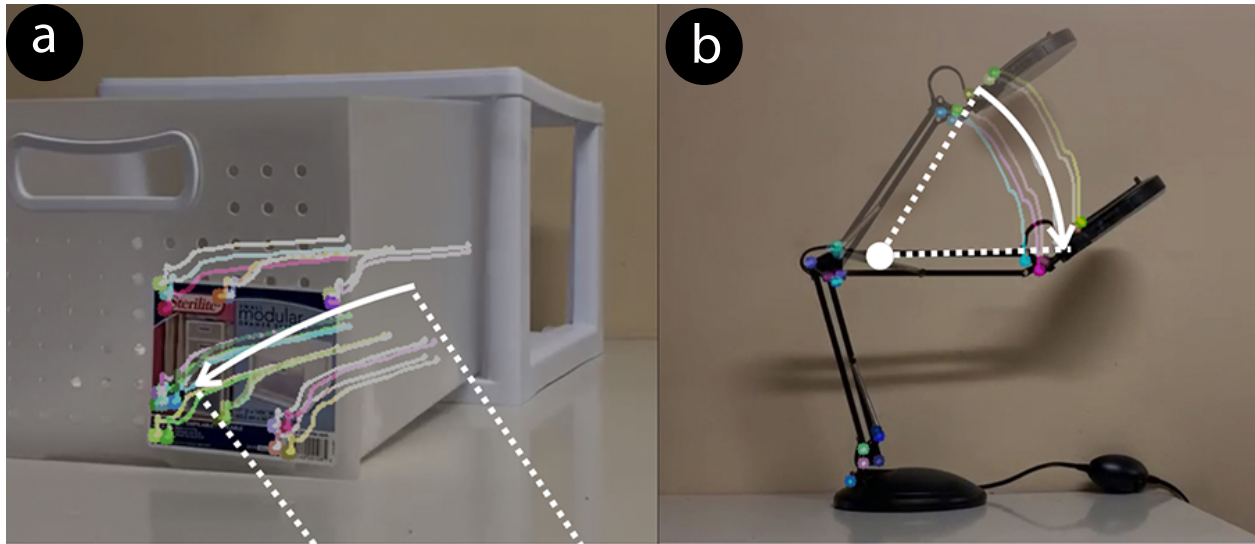


Figure 3.8: Optical flow analysis shows distinct difference between prismatic (a) and revolute (b) joints by comparing the least-square fitted radius (prismatic joint's fitted radius is much larger, extending beyond the camera view).

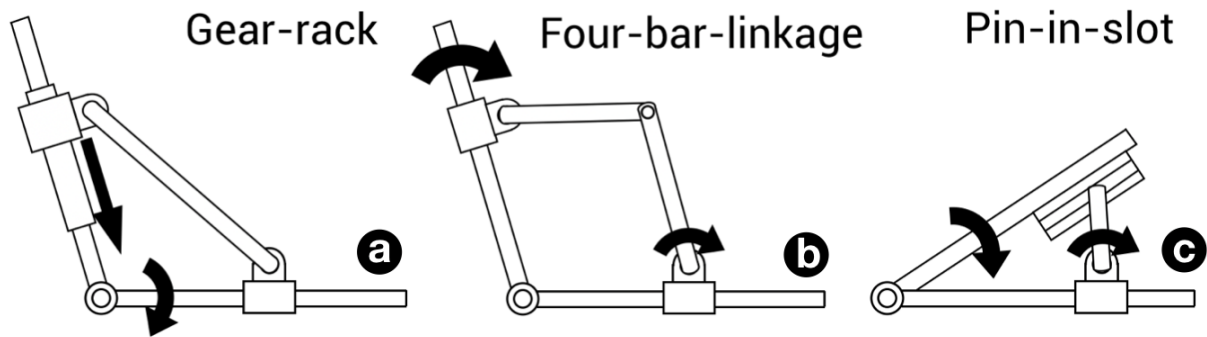


Figure 3.9: Technical sketches of the three mechanisms showing how they are actuated.

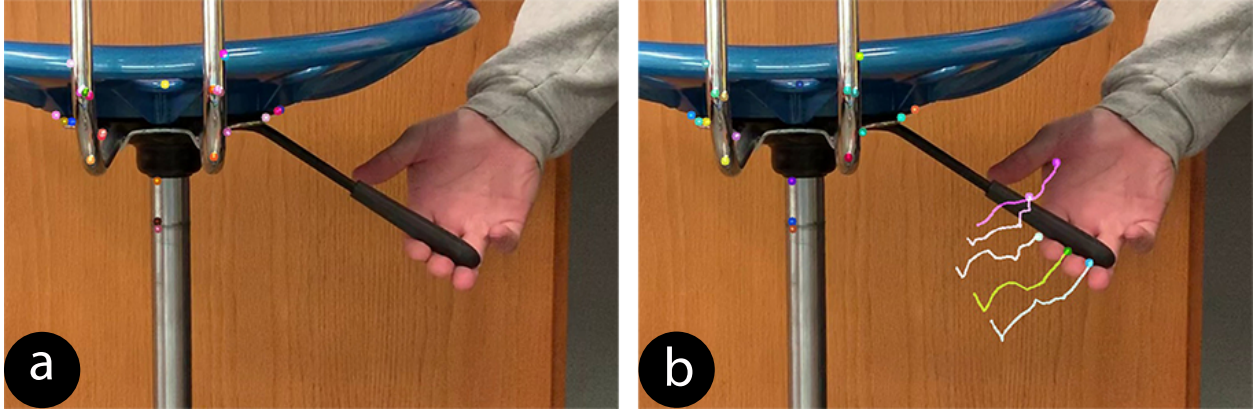


Figure 3.10: Some objects due to their geometry and texture provide very few feature points. To solve this problem, we incorporate a user’s operating hand whose motion correlates to the manipulated object’s.

3.3.4 #3 Finding maneuverable parts and grounds

Having identified the type of joint, the next step is to find out where and how to attach a mechanism to an object. The joint should be fixed to part of the object that does not move, *ground*, and should ‘grab on’ to part of the object that can move, *maneuverable parts*. The key of the following steps is to identify maneuverable parts and grounds, not just in the video domain but also in the 3D space as eventually we will generate 3D models of mechanisms attached to the actual object.

Recall that the above motion analysis (Figure 3.8) has already identified a set of feature points with significant motion trajectory. We use these feature points to train a segmentation model (we use k-nearest neighbor [3]) and apply it to the snapshot ($M_{snapshot}$) of the object’s 3D model, marking each pixel either as maneuverable or ground. As shown in Figure 3.2, we then unproject pixels of the snapshot back to the 3D model using ray casting. As a result, each face of the 3D model is associated with one or more snapshot pixels. We take a majority vote to determine whether the face should be considered maneuverable or ground.

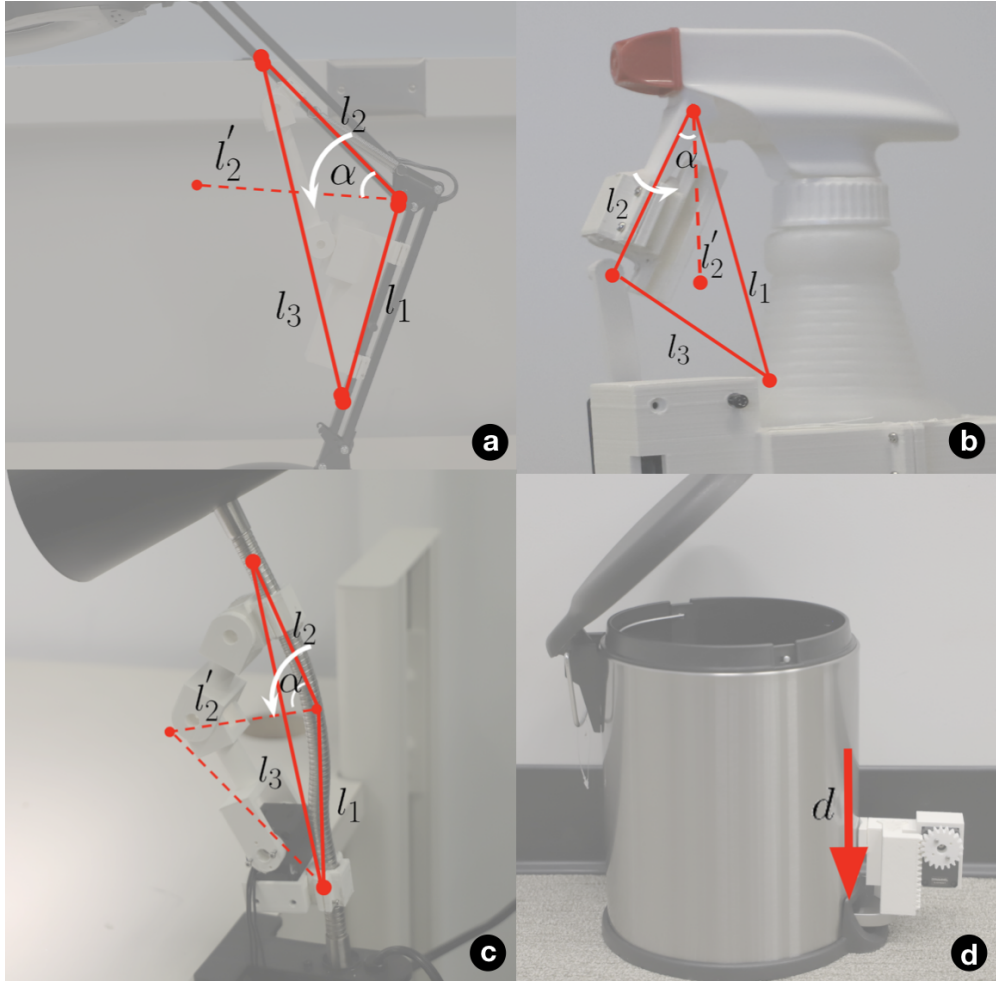


Figure 3.11: Three mechanisms and their parameters (gear rack appears twice for both types of joints). For revolute joints, different mechanisms can be considered and modeled after the same set of parameters – l_1, l_2, l_3 and α .

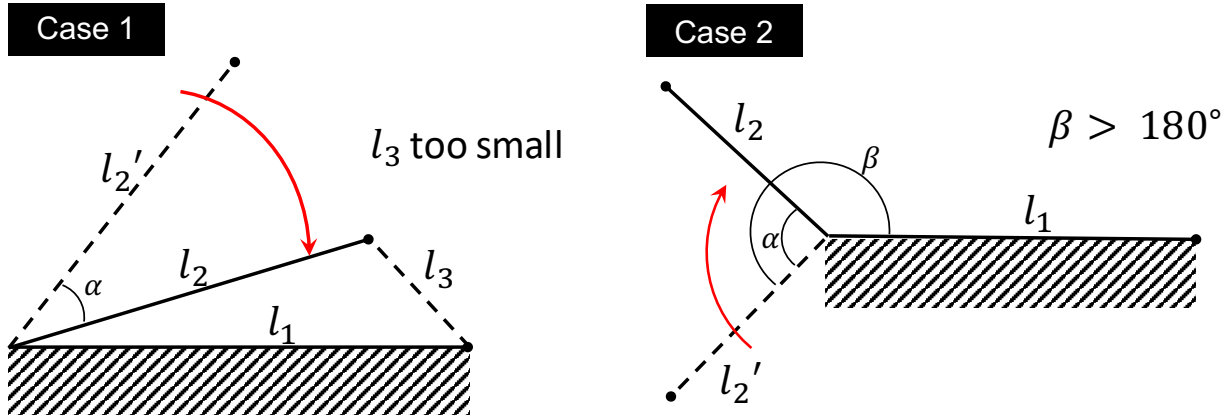


Figure 3.12: Two cases where gear rack does not apply

3.3.5 #4 Generating Actuation Mechanisms & Instructions

As described in an overview, Robiot provides a list of possible mechanisms as options from the library—gear-rack, four-bar linkage, and pin-in-slot—that can afford actuating various everyday objects. Figure 3.9 shows how the three mechanisms are actuated basically. Below we describe how mechanisms are chosen based on constraints and generated from a core set of parameters.

Prismatic joint (linear motion) . Gear rack is the only mechanism that we use for both types of joints. As shown in Figure 3.11(a), the gear-rack system translates the rotary motion of the motor into linear motion, which enables a lamp height (linear) to be adjusted by a motor (rotation). Most of the gear-rack components are standardized, except for the length or rack to be determined by the range of motion, computed from the union of trajectories from the optical flow analysis.

Revolute joint (rotational motion) . When the type of joint is revolute, all three (See Figure 3.11) mechanisms can be considered and modeled after the same set of parameters. Each of the three mechanisms can be generated based on three parameters and their relations: the lengths of the two links l_1 and l_2 (see Figure 3.11) and the range of motion (rotation) α .

Despite their similarity, the choice of one mechanism over another is based on intrinsic constraints. Gear-rack allows larger motion and larger torque at a cost of larger installation

space, whereas pin-in-slot could be installed in a smaller space but cannot provide large torques; four-bar-linkage stands in between and has much fewer applicability constraints.

Specifically, for gear-rack as a revolute joint, two constraints contribute: (i) the size of the motor (currently set to default in the system) and the two connecting components limit the minimum length of l_3 ; (ii) the range of rotation α cannot exceed 180° (figure 3.12). For pin-in-slot, the main constraint is whether it can meet the required torque on the pivot τ_p . For the same motor torque τ_m , pin-in-slot generates a much smaller τ_p ($\tau_p = \tau_m \sin(\beta - 90^\circ)/l_2$) compared to the other two mechanisms.

Programmatically generating 3D models of mechanisms All mechanisms are generated procedurally using basic primitives and boolean operations. For example, for gear-rack, we use trapezoid shape to generate the teeth of the rack and use rectangle or self-defined closed curve to extrude to get the 3D model; the gear is based on an existing example provided by OpenJSCAD. For pin-in-slot, the link attached to the motor to transmit the power from the motor contains a wheel and a connecting rod, which are generated using cylinders. For four-bar-linkage, the link is cylinder-shaped and we use cylinders to generate the connecting joints between two links.

Installation: fasteners and instructions For fastening, we employ Chen *et al.*'s method [27] to compute the circumference of a cross section corresponding to a maneuverable/ground part. We then generate a pipe clamp as part of the mechanism that can be bolted to fasten the mechanism onto existing objects. Other attachment techniques (*e.g.*, [27, 104]) can be also applied based on the target shape.

User interface After the system automatically generates a set of recommended mechanisms, Robiot's UI allows a user to further specify desired strengths and properties (*e.g.* range of motion, required torque and speed of the motion).

Finally, we generate instructions to assist end-users to install the generated mechanisms onto existing objects. We provide standard instructions for fastening a pipe clamp with bolts and configuring a motor; for each specific case, as shown in figure 3.14, we also visualize where on the object to install which part.

3.3.6 Software and Hardware

Robiot’s front end is written in JavaScript using jQuery¹ for UI development, three.js² for 3D graphics, and OpenJSCAD³ for procedurally generating the geometry of actuation mechanisms. The back end Python. Everything runs on a MacBook Pro (15-inch, 2016 year) with a 2.7 GHz Intel i7 and 16 GB 2133 MHz LPDDR3 memory. In our design session and demonstrations, the front end runs on a Google Chrome web browser. We use Dynamixel XL-430 W-250 motors to power the actuation mechanisms, which were all 3D printed using an Ultimaker S5 using primarily white PLA.

3.4 Design Sessions

To validate Robiot, we conducted informal, qualitative design sessions with six participants (aged 20-25, female=3, male=3). The objective of the study is to let participants create mechanism designs to actuate a set of everyday objects using Robiot’s generative pipeline. In so doing, we try to elicit users’ initial reaction and feedback to the system in order to validate Robiot’s easiness to use, its usefulness for automating physical tasks, as well as what to further improve to enhance its efficiency in robotizing things.

3.4.1 Participants

We recruited participants from the university. One participant had a Mechanical Engineering background and one an Electrical and Computer Engineering background, both of which self-reported that they were knowledgeable in mechanical engineering. The other participants did not have any engineering background. Amongst all participants, three had experienced CAD systems, while the others did not. One participant did not even know what CAD

¹<https://jqueryui.com/>

²<https://threejs.org/>

³<https://openjscad.org/>

means.

3.4.2 Apparatus, Tasks and Procedure

There were two different sessions in two days to budget time for 3D printing mechanisms that participants designed on the first day, which they continued to assemble and interact with them on the second day.

Design Session (Day 1) - started with a five-minute quick tutorial. We introduced to a participant how Robiot works step by step using a simple educational example—making an old-school stapler automatic (Figure 3.4). Once the participant understood the concept and the process of Robiot, they proceeded to try out our design tool. Participants were free to choose at least two from a set of seven objects we provided, including lamps ($\times 3$), spray bottle, squeeze bottle, trash can, make-up mirror, and drawer. These objects strike a balance between prismatic and revolute joints, also variations of the same object (lamps), and between different actuation types for similar functionalities (spray vs. squeeze bottles).

The main tasks consisted of participants using Robiot to create an actuation mechanism by taking a video (using an iPhone XS max running iOS 12.1.4) provided by us) as they manipulated each object. To avoid leading the participant, for each object we showed them images of the initial state (*e.g.*, drawer closed) and final state (*e.g.*, drawer open). The participant was asked to manipulate each object to achieve the final state.

After taking the video, participants explored and selected from a number of mechanism designs generated by Robiot from a viewer using a laptop (Figure 3.13b). As we would fabricate user-created mechanisms, we had to budget the printing time, thus allowing each participant to choose two objects, each of which with one mechanism design. The first session took about 45 minutes.

Assembly & Interaction Session (Day 2) took place after we 3D printed participants' designs. On Day 2, the participants were given instructions for assembly generated by Robiot Figure 3.14, based on which they assembled the printed mechanisms and attach to

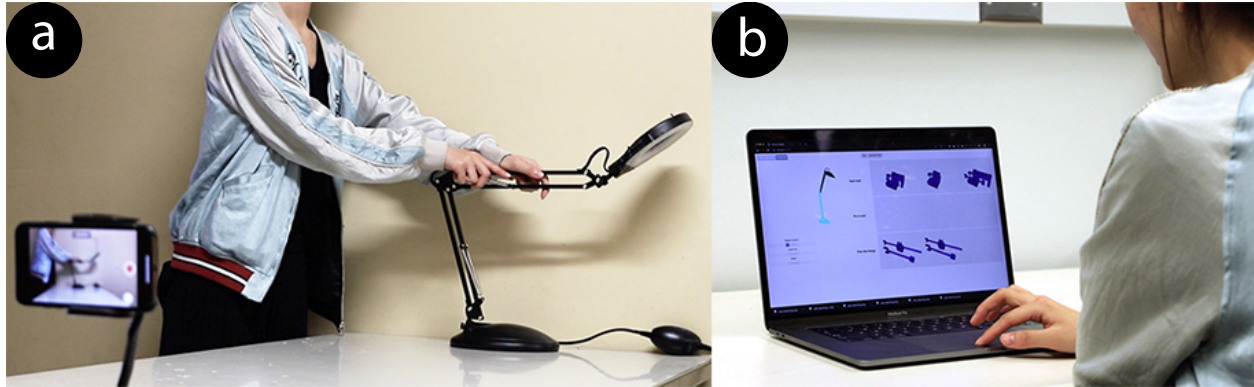


Figure 3.13: Participants in our study took videos of them manipulating everyday objects (a), whereby Robiot generated corresponding actuation mechanisms for participants to explore on our user interface (b)

the corresponding objects. Participants could try out interaction with the actuated objects using pre-defined gestures implemented via a Leap Motion⁴. Our main goal is to let participants experience their created mechanisms in action as they act; studying different types of techniques to interact with such mechanisms and letting them assign desired interactions to actuate them is beyond the scope of this paper, which we leave for future work.

At the end of the session on Day 1, participants filled out a questionnaire regarding overall user experience, including the difficulty to learn how to use Robiot and whether the process required extra effort than what they had expected (in the Likert scale 1-7). At the end of the Day 2, they filled out another questionnaire to answer (i) how difficult it was to assemble the mechanisms; (ii) whether the installed mechanisms behave as they expected; (iii) perceived usefulness of having such mechanisms to actuate legacy objects. Then we solicited feedback about the entire design and fabrication process and any suggestions for improvement.

⁴<https://www.leapmotion.com/>

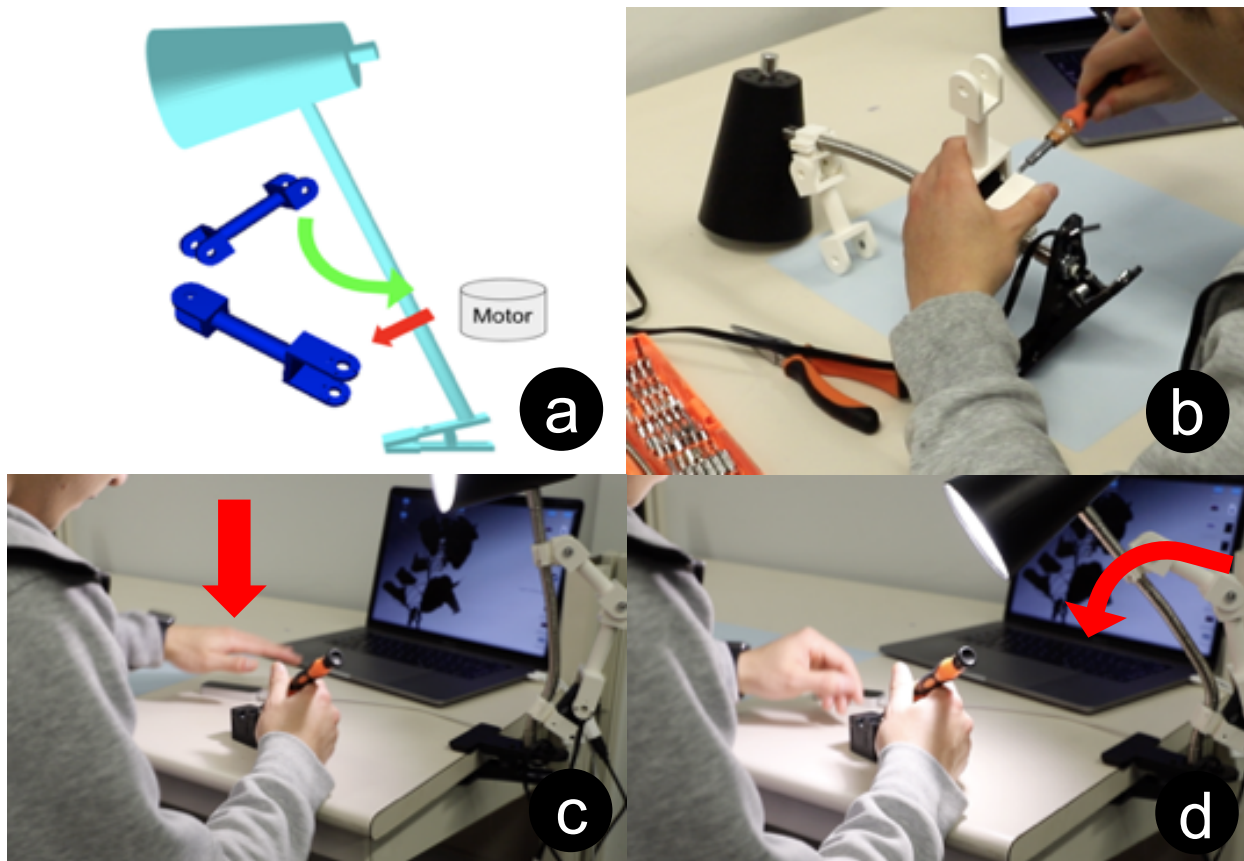


Figure 3.14: Robot generates an instruction for assembly (a), then a participant can assemble and install following that instruction (b), to interact with the robotic things using gesture (c) lowering the lamp height to work with optimal brightness when soldering (d)

3.4.3 Results and Findings

Participants created a total of 14 mechanism designs using Robiot. All but one participant successfully assembled their designs (the only failure case was due to a critical component broken before the second session).

We first analyzed participants' questionnaire responses for the quantitative analysis. Table 3.2 summarizes the results on each question, gauging easiness and usefulness of the tool and design pipeline. Then we transcribed video recordings to observe key insights from participants' behaviors for qualitative analysis. We transcribed video data based on the context (e.g., taking video, using Robiot desktop system, assembly, etc.), logging participants' spoken responses (e.g., "why does this look [the] same?") and description on their behaviors (e.g., P1 tried to handle 2 DOF at a time). Then we classified this data to identify findings as follows.

Easiness of using the System - The most participants reported that the pipeline is simple and the user interface is easy to use, being able to operate without much background knowledge (Q1-2). Participants commented "*I've never thought that machines can be made through that simple steps*" (P4) and "*I thought designing robotic tools involves heavy measurement, designing and trial and errors. But using Robiot, I simply record the video and it automatically did all the work*". However, one participant reported an important aspect about instruction "*I won't know I can rotate and zoom if I am using it independently*" (P6), which suggest user interface improvement to better inform possible functionalities, what design options are available when users perform a design task.

Accessible Pipeline and Design Assistance - Participants had no problem assembling and installing the mechanisms, interacting with objects actuated by the mechanisms (Q3-4), reporting "*The part is straightforward, I just need to put the joints together and tighten screws. Then everything works well*" (P1) as well as "*It's only few components with clear instructions*" (P3). Also, all participants were satisfied with generated motions, because robotized objects behaved as they expected from the beginning of the design. However, one participant mentioned "*It works well though not very sensitive*" (P4), raising concerns on

* 1: Strongly disagree – 7: Strongly agree

1	2	3	4	5	6	7	Mean
Q1: It is easy to learn how to use Robiot							
1	-	1	-	-	3	1	4.8
Q2: The process requires less effort than I had expected							
1	-	-	-	1	2	2	5.3
Q3: It is easy to assemble the mechanisms							
-	-	-	1	2	1	2	5.7
Q4: The installed mechanism behaves as expected							
-	-	-	-	1	2	3	6.3
Q5: It is useful to have such mechanisms to actuate legacy objects							
-	-	-	-	1	5	-	5.8

Table 3.2: Selected statements with survey scores, counts in each cell indicate how many participants rated their scores

the granularity of motion Robiot can generate. We will discuss in more detail later in the paper.

Usefulness of the Tool and Aesthetics - All participant were in favor of the tool as they foresee potentials (Q5) to help them "customize [my] own things that meet my own needs" (P3), "on many objects around me" (P4). Though, participants also have higher expectations in aesthetics and design to fully utilize the system in their everyday lives, commenting that "Without proper design, these modern mechanisms will look strange on old-time objects" (P1) and "The design part could be previously done by specialists, instead of automatic algorithm"

(P2). We expect addressing concerns around design aspects would expand future use case of Robiot, by involving more users who care aesthetics in creating custom robotic things.

Remaining Challenges - There were a few common challenges among participants. Some participants struggled to understand what ‘initial state’ and ‘final state’ meant. In hindsight, we realized such wording was too technical, and perhaps an alternative expression such as ‘before/after’ would have been more understandable. Also, participants did not react positively to the sliders that can further adjust design parameters and filter a subset of generated mechanisms. Although participants were satisfied when finishing the process, they did comment on a lack of understanding of how things work, such as *”I cannot understand why I choose one mechanism, or how to choose one option”* (P6). In addition to the instructions we provided for assembly, in the design phase, animated previews of each suggested mechanism in action and step by step instructions for users to manipulate a design interface would help them feel more engaged in the design process with less confusion.

3.5 Discussion

In this section, we discuss existing issues, limitations, and opportunities for future work.

3.5.1 Future Technical Work

There are several technical details Robiot needs to focus on in the future. In cases where a user accidentally blocks the camera at the beginning or the end of the video, one future direction will be employing computer vision techniques to detect such occlusion and providing a simple UI for the user to select a better, unblocked start/end frame.

As there are usability issues with sliders in user interface, one future direction of user interface will be providing interactive tutorials to help users understand the mechanical effect of adjusting each slider.

3.5.2 Scale of Mechanisms

Robiot suggests mechanisms that best suit user-specified action and generate a 3D printable model by a desktop 3D printer. The scale of mechanisms and mechanical elements (e.g., size of gear teeth and the length of the rack) are dependent on the capability of the printer, with common hardware settings. Investigating possibilities for the system (1) to design larger mechanisms to support furniture scale objects' actuation and (2) to handle granularity of the motion, which is defined by the size of mechanical elements, could be an interesting extension of our work.

3.5.3 Camera Angle to Capture Desired Motion

To best extract series of motion path captured from a video, users need to capture the objects in motion *orthogonal* to the movement paths. Because Robiot currently lets users design mechanisms in one degree of freedom at a time, motion extraction and generating mechanisms are based on 2D, where an orthogonal scene best derives the motion. Prior work has investigated retaining 3D information when converting 2D videos [82, 124], by estimating depth from 2D scenes. Another future direction of Robiot is extending its capability to extract features for motions with depth in 3D from 2D videos, and generate mechanisms in multiple degrees of freedom at a time that addresses 3D motions.

3.5.4 Designing Motion Beyond Given Affordances

Currently, Robiot helps users to create 3D printable actuation mechanisms to perform *expected* action of legacy objects. For example, users are likely to design height adjusting mechanisms for a chair and a rotating mechanism for an old water faucet. Users' choices on actuating mechanisms are decided by the existing affordances of a physical interface, it is hard to imagine a user would change these affordances, such as making a linear switch to attach on a rotating faucet and turning pulling drawer opening in rotational angle. Reprise is an approach that allows users to change the type of required movement to perform actions

on physical handheld objects, particularly for people with fine motor impairments [28]. One interesting future direction could be applying this technique to generate mechanisms that enable users to alter the type of motion to perform a fixed physical task, such as rotational to linear motion and vice versa.

3.5.5 Sensing and Designing Custom Interactions

The main contribution of ours is an end-to-end pipeline, enabling designing fabricable mechanisms; sensing and defining custom interactions, and mapping them to desired actions are beyond the scope of this paper. Nonetheless, there exist commercial kits that welcome average users to install add-on motion sensors to trigger actuating home IoT (e.g., [1, 85]), and it becomes common to perform such tasks using connected devices or voice assistance using home intelligence devices (e.g., [79]). We demonstrated the feasibility of mapping custom gesture interactions using LeapMotion, which opens future opportunity to implement novel interfaces for end users to define custom user interactions to fine-control the motion, given different user requirements. With these interfaces, existing work on novel sensing techniques on everyday objects to detect unique object touch [186] or direct sensing of a human body to detect motion [35] can be applied to enrich user experiences in everyday use of robotic things.

3.5.6 Performing Ungrounded Action

Recent advancements in ubiquitous computing have presented the future vision of moving objects that do not require explicit user intention to perform such actions. For example, Nissan showcased their visionary self driving cars through smart slippers [158] and chairs [157] that self-organize. It is also viable to imagine salt and pepper bottles shaking by themselves, as having them on a soup bowl is a common routine. Envisioning the future with omnipresent robotic things that accommodate people’s everyday routines and expected activities as triggers (e.g., sandals coming to you when you come into the door, desktop self-adjusting height as you stand up to refresh your posture) by predefined activity sensing,

Robiot sheds lights on a new possibility for end-users to robotize objects at home and office to make their everyday life easier towards functional home/office.

CHAPTER 4

Enabling Users to Design and Generate Embedded Mechanisms to Robotically Augment Default Functionality for Legacy Objects

Reconfiguring shapes of objects enables transforming existing passive objects with robotic functionalities, *e.g.*, a transformable coffee cup holder can be attached to a chair’s armrest, a piggybank can reach out an arm to ‘steal’ coins. Despite the advance in end-user 3D design and fabrication, it remains challenging for non-experts to create such ‘transformables’ using existing tools due to the requirement of specific engineering knowledge such as mechanisms and robotic design. In this chapter, I present *Romeo*—a design tool for creating transformables embedded into a 3D model to robotically augment the object’s default functionalities. Romeo allows users to express at a high level, (1) which part of the object to be transformed, (2) how it moves following motion points in space, and (3) the corresponding action to be taken. Romeo then automatically generates a robotic arm embedded in the transformable part ready for fabrication. Romeo is validated with a design session where 8 participants design and create custom transformables using 3D objects of their own choice.

4.1 Introduction

Objects that can transform its geometry and/or functionality (which we referred to as transformables) hold the promises of dynamically adapting to multiple usages by reconfiguring shapes, either automatically or via manual reconfiguration as depicted in Figure 4.2 with several examples.

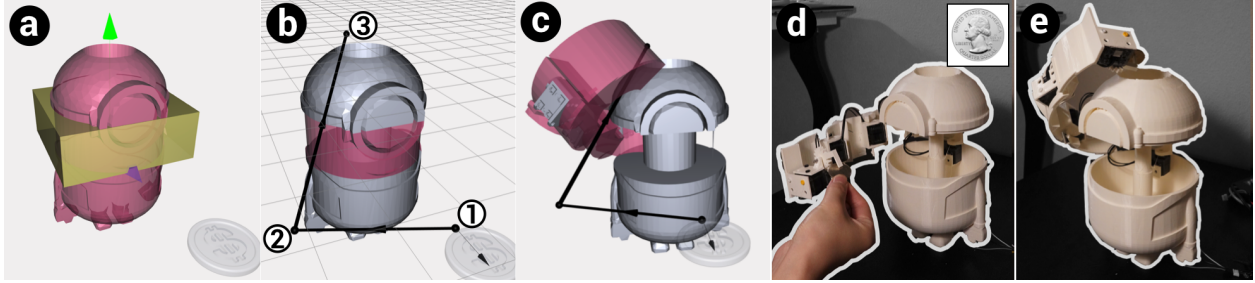


Figure 4.1: Romeo enables a user to transform a Minion model into a coin-stealing piggybank by selecting the mid-section of the Minion as the transformable part (a), specifying motion path for the transformed part to pick up a coin and place it in the bank (b), generating the corresponding transformable robotic arm (c) and 3D printing and installing the result (d-e).

The advent of computational design and 3D modelling tools offers the possibilities for casual makers [84] to create custom objects. However, for non-technical users, designing transformables remains challenging due to the requirements of expert-level engineering knowledge such as mechanisms and robotic design that are not provided in existing end-user oriented design tools.

Prior work focused on *actuating* passive objects [28, 174, 117] using attachable mechanisms, yet has not taken account how to *transform* given objects. Prior work that addressed transformables tends to focus on computational analysis of geometry and optimization [225, 231, 83], rather than providing design assistance to create user-defined custom transformables. Further, these approaches almost exclusively exploit geometry as the only constraint when generating a transformable design (*e.g.*, [225]), with little consideration of obtained functionality, *i.e.*, how a transformable can perform a user-defined task.

I present Romeo—a design tool for generating and embedding transformable parts in 3D models to robotically augment the default functionality. Figure 5.1 shows an exemplar workflow of Romeo: A user selects a cross section of the minion to be a transformable part (a), specifies a motion path to pick up a coin and place it in the bank (b), based on which Romeo generates the corresponding transformable robotic arm (c), which is then 3D printed



Figure 4.2: Existing examples of objects with multiple functionalities by transforming its geometry: a stealing coin piggybank (a), a transformable cup holder (b), and an iPad cover folded into a stand (c).

and installed with motors to perform the coin-stealing task (d-e).

I conducted a design session with eight participants conducting both controlled task to replicate a task using given 3D objects (stirring spatula and sanitizing tissue box) and open-ended tasks (design a piggy bank using participant-chosen 3D model) of creating transformables. Overall, participants were able to understand and complete design using Romeo as a design tool, to define functional tasks by parts selection and motion path specification, to generate ready-to-print transformable parts for fabrication, and to assemble the parts into a fully functional transformable object.

The main contribution is an end-user design tool that addresses *both* geometric and functional constraints in generating a transformable part embedded into an object’s 3D model (geometric constraint) while being able to transform and perform a user-defined task (functional constraint).

4.2 Related Work

Romeo provides end-users with a design tool to augment an object’s functionality by transforming a subset of the object. This goal cross-cuts three areas of prior work: (i) designing

objects that can transform original shapes (*ii*) reality-based design tools that extract real-world information (e.g., object geometry and context) to create designs; and (*iii*) interactive design of robotic characters.

4.2.1 Computational Design of Shape-Changing Objects

Past research has explored the design of articulated objects that can actuate existing shape by transformation. Calì *et al.* proposed to generate assembly-free models by inserted joint configurations of an animation rig to articulate mechanical object [22]. Bächer *et al.* takes a skinned mesh as input and estimate the corresponding virtual articulated part segment to compute placement of the joints [13]. Ureta *et al.* proposes an interactive system for creating 3D printable joints with user-controlled appearance while taking into account the range of motion achievable [208]. All this work focuses largely on inserting assembly-free joints in the articulated parts given the input shapes, rather than to transform such objects away from their given shapes.

One body of research on transforming shapes is concerned with the design of reconfigurable objects. Li *et al.* and Garg *et al.* proposed computational approaches to design foldable furniture to save space while not in use [115, 61]. In this process, Boxelization helps transform a 3D object into a series of small cubes where adjacent cubes are either linearly-linked and/or fold [231]. Huang *et al.* explores the design and animation of the motion of transformation based on the input 3D model and target skeleton representing the desired figure [83]. Yu *et al.* investigates transforming the object with telescopic structures using user sketches or an arbitrary mesh as input [223]. Perhaps the most related to our work is the design approach by Yuan *et al.*, which uses the target model and skeleton as user input to automatically generate fabricable transformable objects [225]. However, Yuan *et al.* only addresses the automatic generation of transformables based on input geometry constraints, but does not consider higher-level goals of the desired function as input. Across prior work, there is a lack tool support for non-expert users to create custom transformable objects.

Also, design of origami robots is also related as such robots can be initially fabricated

as flat sheet then be folded into a complex 3D shapes. Schulz *et al.* proposed an end-to-end system for design of robots with ground locomotion [191]. Mehta *et al.* proposed a tool that users can create printable 3D origami-inspired robot from high-level structural and functional specification [142, 143, 144]. Romeo focuses on using existing 3D shapes with a default function and transform into another 3D shape with augmented functionality.

4.2.2 Reality-based Design Tools

Romeo enables end-users to define a task that the transformable part will conduct, to interact with real-world objects (*e.g.*, picking up and depositing a coin). Existing work has explored reality-based design tools that address (*i*) how to design objects with kinematic and robotic features from (*ii*) high-level, real world context-based design goals as an input, while leaving low-level functional considerations to a generative algorithm.

Design of objects with kinematic feature that could physically interact with the real world has been a focus of many research. Obtaining an understanding of the physical world informs new ideas of augmentation, such as ‘mechanical hijacking’ first demonstrated by Davidoff *et al.* facilitated physical interaction with real-world object using LEGO MindStorm toolkit [39]. TrussFormer enables users to design and 3D print large-scale structures with kinematic features [102]. Grafter helps end users extract and reconstruct mechanical elements from 3D printable machines. It affords users to be aware of real-world constraints when fabricated, allowing to clean out sweeping space along with rotation axles[183]. RetroFab offers an authoring tool to scan an existing physical interface and automate its controls with an enclosure consisting of mechanical widgets and electronic devices [174]. Robiot turns legacy static objects into robots by generating 3D printable attachment mechanisms to automate physical tasks [118]. Similarly, Romeo aims at empowering end-users to create transformable parts of objects that are able to physically interact with real-world objects, given context-aware, user-defined task.

Generative design is also related to our work in that it allows users to only provide high-level information as input. Reprise invites users to express what type of action is

applied to an object at a high level, and so generates adaptation for hand-held objects for easy manipulation [28]. AutoConnect promotes the automatic generation of the attachment mechanisms between two user-selected objects based on input 3D model or scanned digital model of real-world object [103]. Further incorporate users' intent, Forte loops user input into the optimization process to create structures that meet functional requirement as well as mimic users' sketches [29]. Patching provides a hybrid platform that scans, mills, and fabricates new components ad-hoc, to replace part of an existing object with updated user-context [203]. DreamSketch allows a user to integrate generatively designed components with the workflow of sketching [93]. In comparison, Romeo employs a hybrid approach that combines the automation of robotic feature generation as well as a generative process: users input a 3D model and specify a reality-based target task, based on which the tool generates the corresponding transformable parts.

4.2.3 Interactive Design of Robotic Characters

The eventual goal of Romeo is to generate functional ready-to-transform objects based on user-defined high-level specification. To achieve this, Romeo builds upon prior work that examine interactive design of robotic characters.

Megaro *et al.* present a design system that automates the tedious process of creating 3D-printable robotic creatures while allowing for customization for casual users [140]. Recent research focused on computational approaches for non-expert users to design animated or robotic characters by high-level motion specification such as motion curves [37, 205, 68]. LinkedIt enables end-user design and fabrication of robots, from creating linkages to exhibit specific motion path [14]. Geppeto is an interactive system that allows users to design expressive robots by editing complex parameters in a semantic level [43]. Using modular electromechanical components known as 'computational abstractions', novices become capable to easily create custom robotic devices using a visual design environment [46, 45]. In contrast, Romeo enables the generation of robotic mechanisms by only requiring users to specify sample points along a motion path and to select an action to be taken along the way,

thus simplifying the specification of tasks into just a few steps.

4.3 Overview of Design and Fabrication Process

We break down Romeo’s process of generating a user-defined transformable part into four steps, which we briefly describe here and discuss in further details in the next few sections:

1. **Selecting which part of the object to be transformed.** To start, Romeo allows users to select a part of an 3D object to be transformed by sweeping its cross-sectional area along one of the X/Y/Z axes.
2. **Specifying motion points to follow and corresponding action to be taken.** Romeo lets users specify a task that consists of a series of motion points for the end-effector of the robotic arm to follow, as well as what action should be taken at each motion point: *(i)* pick or place, *(ii)* follow a trajectory and *(iii)* attach to a surface.
3. **Generating a robotic arm embedded in the transformable part.** Romeo then generates a robotic arm that follows the user-specified motion points to perform a task, segmenting user-specified parts into a series of joints. The object-embedded arm is visualized and the resulting motion is animated so users can iteratively modify their design.
4. **Generating components for fabrication and deployment.** Finally, Romeo generates fabrication-ready 3D models, guides to assemble all the components along with motors, and software needed to control the motors that actuate the robotic arm.

4.3.1 Preprocessing

Before the workflow starts, Romeo assumes the input 3D model has been preprocessed using existing CAD tools: *(i)* the model has been oriented to be as axis-aligned as possible, *e.g.*, a spatula’s handle aligned with one of the X/Y/Z axes, a minion model at an upright orientation; *(ii)* non-transformable functionality has been implemented in the model, *e.g.*, a



Figure 4.3: A screenshot of Romeo’s user interface: a) target object; b) reference object; c) functional buttons, from left to right: selecting transformable part, specifying motion points and action, generate embedded robotic arm, animation, export and d) button to restart the current step

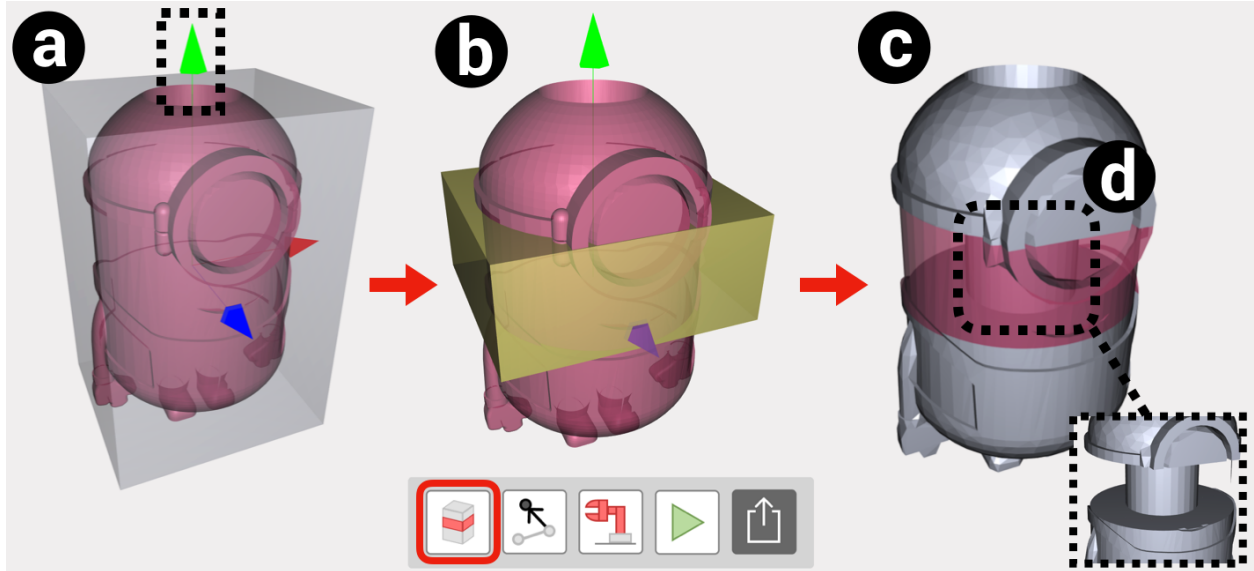


Figure 4.4: Romeo enables selecting part of an object to transform by sweeping cross-sectional area along X/Y/Z axis.

model for piggybank has been hollowed for storing coins; *(iii)* optionally, reference objects have been placed around the model with its relative position, *e.g.*, a pot placed next to a spatula to serve as a reference when a user specifies motion points of the spatula stirring in the pot.

#1 Selecting Part of An Object to Transform

Currently Romeo supports selecting a transformable part as a cross-sectional area swept along one of the X/Y/Z axes of the object with its bounding box (Figure 4.4). Such an axis-aligned selection approach is designed to simplify the 3D manipulation task for non-expert users. Admittedly, it trades off expressiveness, *e.g.*, selecting a semantically-meaningful part unaligned with X/Y/Z such as minion’s goggle. which we further discuss later as future work.

Once an axis is selected by clicking an arrow parallel to each axis, users could adjust the starting and ending position of the sweeping area, by dragging the two corresponding

surfaces of the bounding box highlighted in yellow (Figure 4.4b). A boolean operation is conducted to obtain the intersection of the target 3D object and the selected cuboid. This intersection is then set to the transformable part and the rest is automatically set to the static part (Figure 4.4c).

One issue in this step is that the transformable part is likely to divide the object into two disjoint components in the static part, sometimes causing overhang due to a vertical segmentation (*e.g.*, Figure 4.4 c). Romeo can detect such cases by comparing the transformable part's and the entire object's bounding boxes. If disjoint components are detected, Romeo generates a cylinder in between as a connection (d). The radius of the cylinder is proportional to the dimension of the transformable part, while constrained by the need to leave sufficient space for mounting motors around this pillar.

#2 Specifying Motion Points and Action

The next step is to specify the motion points of the selected transformable part, in order for the parts to follow and operate the corresponding action. The transformation is led by the end-effector—a component that follows the path defined by user-specified motion points. For example, the tip of the transformed arm of the piggybank is the end-effector that will follow the motion points to steal the coin (Figure 4.8).

Motion points involve two types of information: 3D positions and orientations (the orientation of the end-effector when it approaches a point). At each motion point, the end-effector will take the action in one of the three types: pick-and-place, following a trajectory or attaching to a surface.

Position One well-known challenge here is letting users directly specify a 3D point on a 2D screen. Romeo addresses this by providing a two-step process. First, the user specifies the 2D position of a motion point on a reference plane coplanar to the cross section of the transformable part (Figure 4.6a): for the very first motion point, the reference plane cuts across the centroid of the transformable part and for the subsequent points, the reference

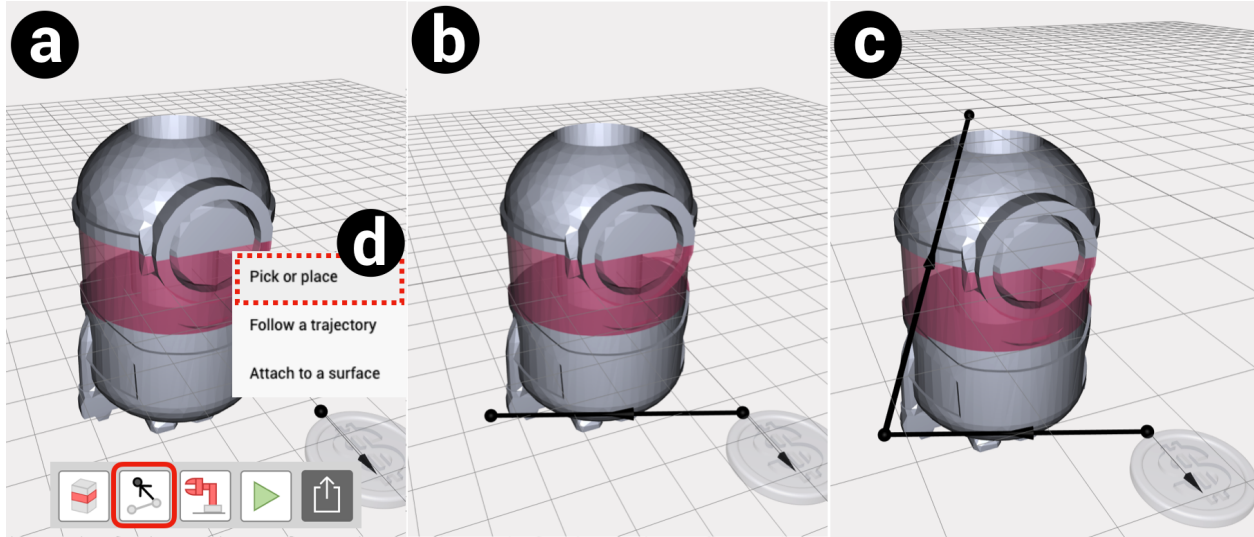


Figure 4.5: Users can specify the motion points for the end-effector to follow in order (abc) and the corresponding action to be taken (d).

plane cuts across the previous motion point. Next, the user specifies the third dimension by sliding the motion point along a reference line perpendicular to the reference plane (Figure 4.6c). To help user have a better understanding about the relative position of motion points in 3D space, a top view and a side view are displayed on the right side of the tool (Figure 4.6b-d). While the end-effector tracks the motion points in sequence, users could specify a loop by setting the last point close to the first point (the threshold is 50 mm).

Orientation As shown in Figure 4.6e-g, we use a spherical widget similar to [28], to help a user specify the orientation of the end-effector as it approaches one motion point from the previous motion point. Orientation is only required in the special context, for example, the contact face of the stamp needs to be parallel to the paper placed on the ground, while stamping. Users can click on the spherical surface to specify the orientation, or by clicking anywhere outside to indicate that no specific orientation is needed.

At each motion point, the end-effector can perform one of the following three actions:

- **Picking/placing** Picking an object and placing it at another location is the most common task for a robotic manipulator useful in many real-world contexts, *e.g.*, in the

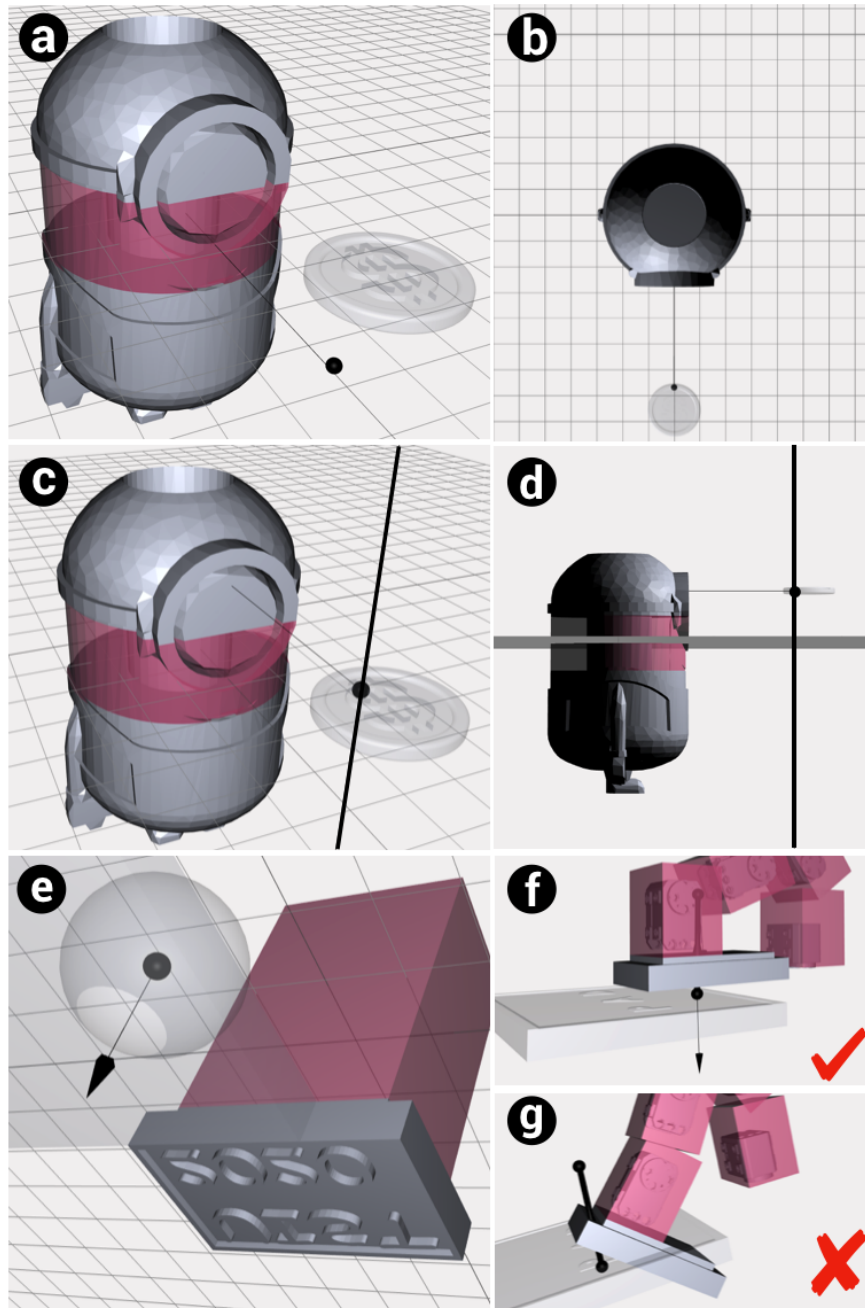


Figure 4.6: Romeo allow users to first specify the 2D position on a reference plane (ab), and then the third dimension along a perpendicular reference line (cd). A spherical widget is used for specifying the end-effector orientation (d). Lack of orientation specification may result in bad result (fg).

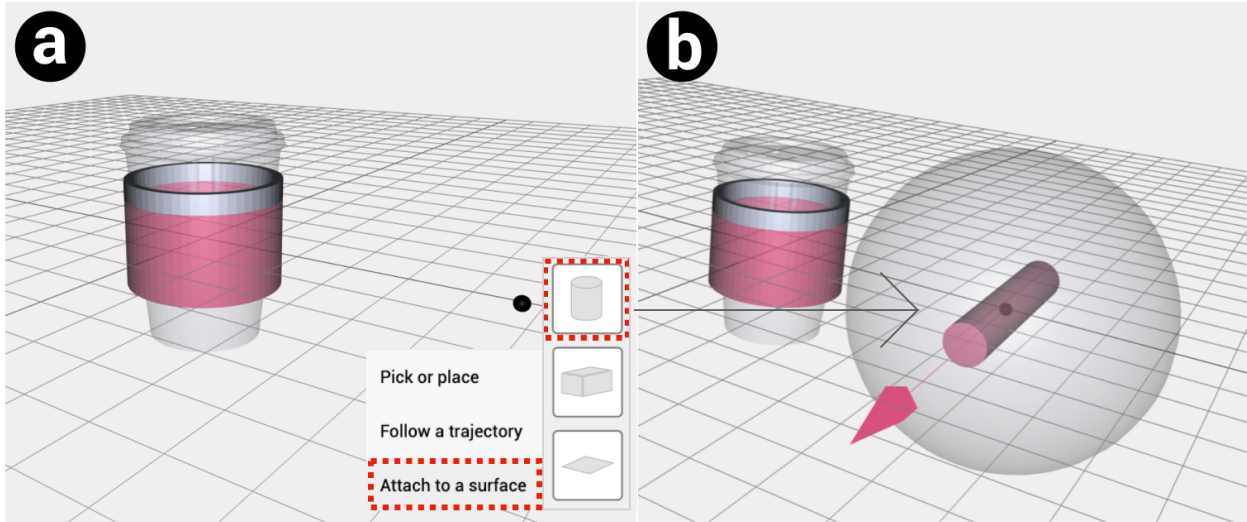


Figure 4.7: Romeo lets a users to specify the target surface for attaching action (a) and further define the orientation of the surface (b)

assembly line. The first choice of this action automatically becomes a *picking* action and the following choice will be regarded as a *placing* which is subsequent to the prior action.

- **Following a trajectory** indicates that the end-effector simply moves to motion point without taking any other action. For examples, additional motion points between pick and place are simply trajectory following.
- **Attaching to a surface** can be used to transform an object to attach some parts to an existing physical object, *e.g.*, coffee cup holder (Figure 4.7a) that can be mounted on a chair’s armrest. Romeo provides three standard types of surface to represent the existing geometry: cylinder, rectangular-prism and flat-plane [103] to specify an attachment surface. A user places a surface object similar to specifying the position of a motion point, then the user can further specify the orientation of the surface using a spherical widget (Figure 4.7b).

#3 Generating An Embedded Robotic Arm

As the user finished defining the task, Romeo takes the transformable part and motion points as parameters to generate an embedded robotic arm. The union of all the points reachable by a robotic arm is found and called as workspace, determined by linkage length and the moving range of each joint.

One challenge here is the trade-off between embedding more links to enlarge the workspace and securing space to host these links the limited volume of selected part. We found an optimal balance for Romeo to segment the part into a four-joint robotic arm, as with four joints, Romeo could change the moving direction of each joint to cover a wide variety of user-defined tasks. We determined the number of joints to be four after investigating several pilot examples. We define different combinations of each joint as configurations of the robotic arm.

Another consideration for a user to decide is at generating a robotic arm, to determine the *base* and the *end-effector*. In the case of the coin-stealing piggybank (Figure 4.8b), it is natural to use the static part as the base and the end of the transformable part as the end-effector for pick/placing coins. However, in some cases the relationship needs to be flipped. For example, for a spatula (Figure 4.8a), the static part (*i.e.*, the blade) becomes the end-effector as it carries the stirring function requisite for the user-specified task. In Romeo, a user can click a button (third from left in Figure 4.3c) to switch between two types of transformation, using the static part as the base vs. as the end-effector.

Below we detail two key steps for Romeo to generate a robotic arm from the selected part: (*i*) segmenting the transformable part and (*ii*) generating joints between segmented links.

4.3.2 Segmentation of the Transformable Part

Romeo employs two ways of segmentation based on the shape of the transformable part: slender and non-slender.

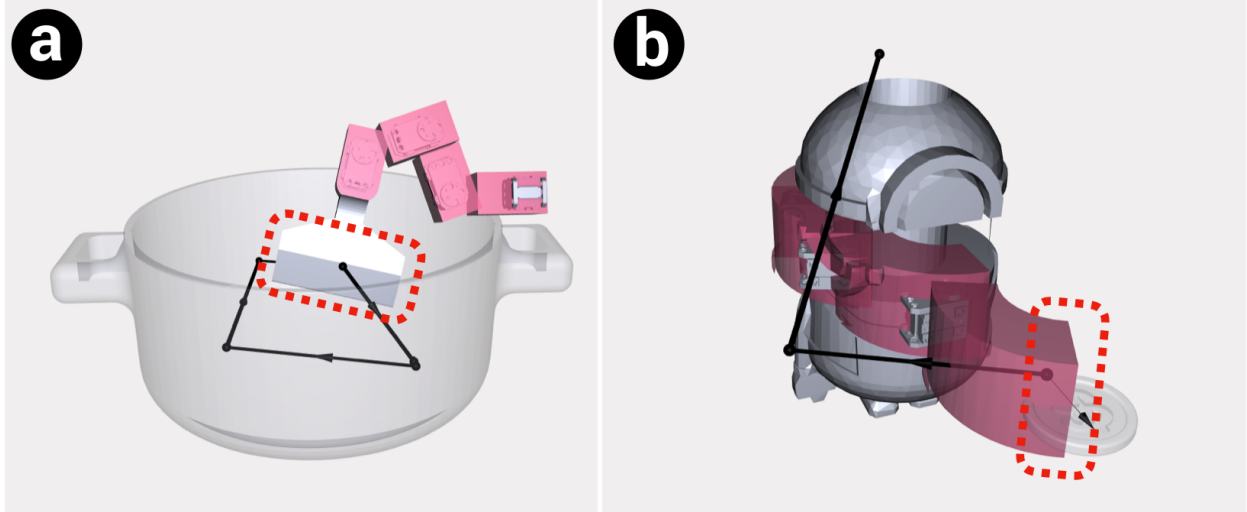


Figure 4.8: End-effector can be on different part of the object: static part of spatula (a) or end of the transformable part of Minion piggybank (b)

Slender shape Romeo considers a transformable part as slender if one of the dimensions (X/Y/Z) is at least four times as longer as the others ¹. For this type of shape, Romeo segments it by quartering along its longest axis into links of equal lengths (Figure 4.9a).

Non-slender shape For a non-slender transformable part, we need to first determine a *principal axis*, the normal axis of the plane whereon the transformable part unfolds. Romeo picks the cross-section of transformable part with the largest area as the unfolding plane and set its normal as the principal axis. Then Romeo quarters the transformable part equally around the principal axis to operate segmentation (Figure 4.9b).

4.3.3 Searching for Robotic Arm Configurations

Romeo generates a four-joint robotic arm, divided into two groups—one *steering* joint and three *driving* joints (Figure 4.12). Generally speaking, a steering joint controls the overall orientation of the robotic arm, the choice of which is based on the motion points; the driving joints control how far the joint can reach, parallel to the principal axis. By only changing the

¹Recall that we have preprocessed the object to make it as axis-aligned as possible.

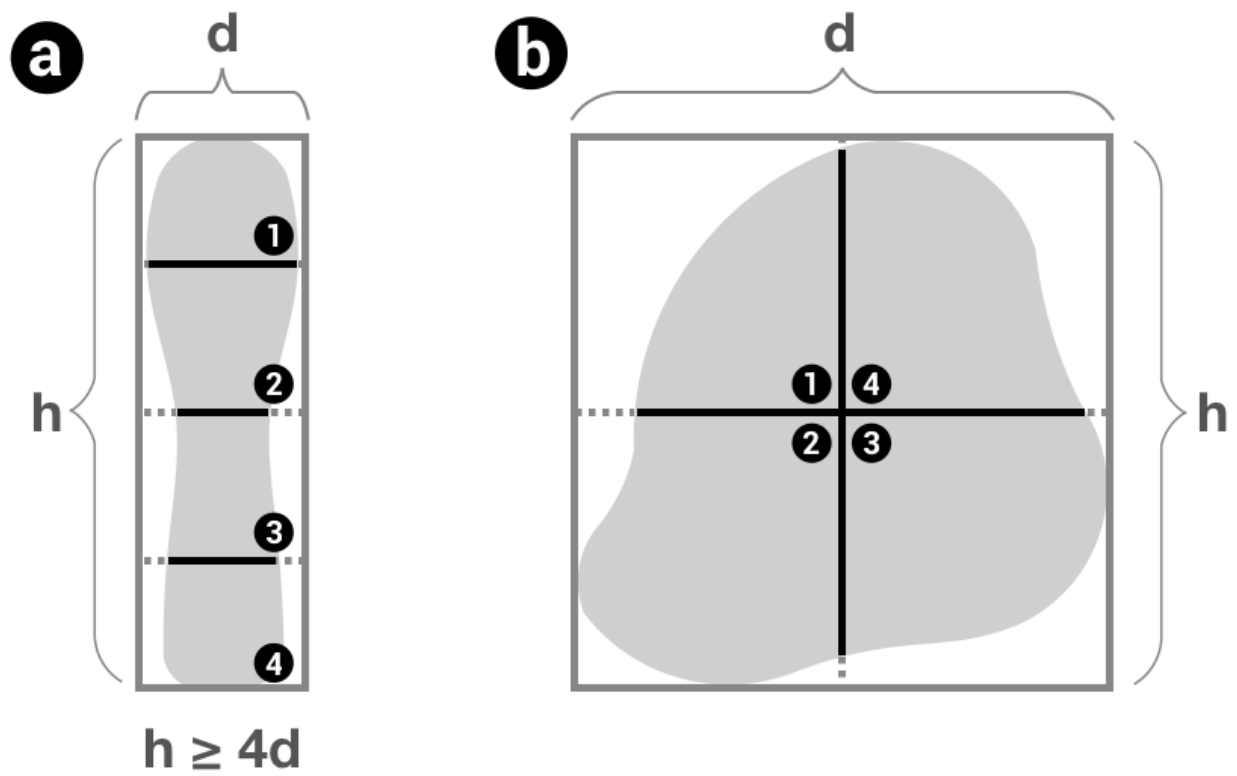


Figure 4.9: Two types of segmentation, (a) slender (b) non-slender shapes.

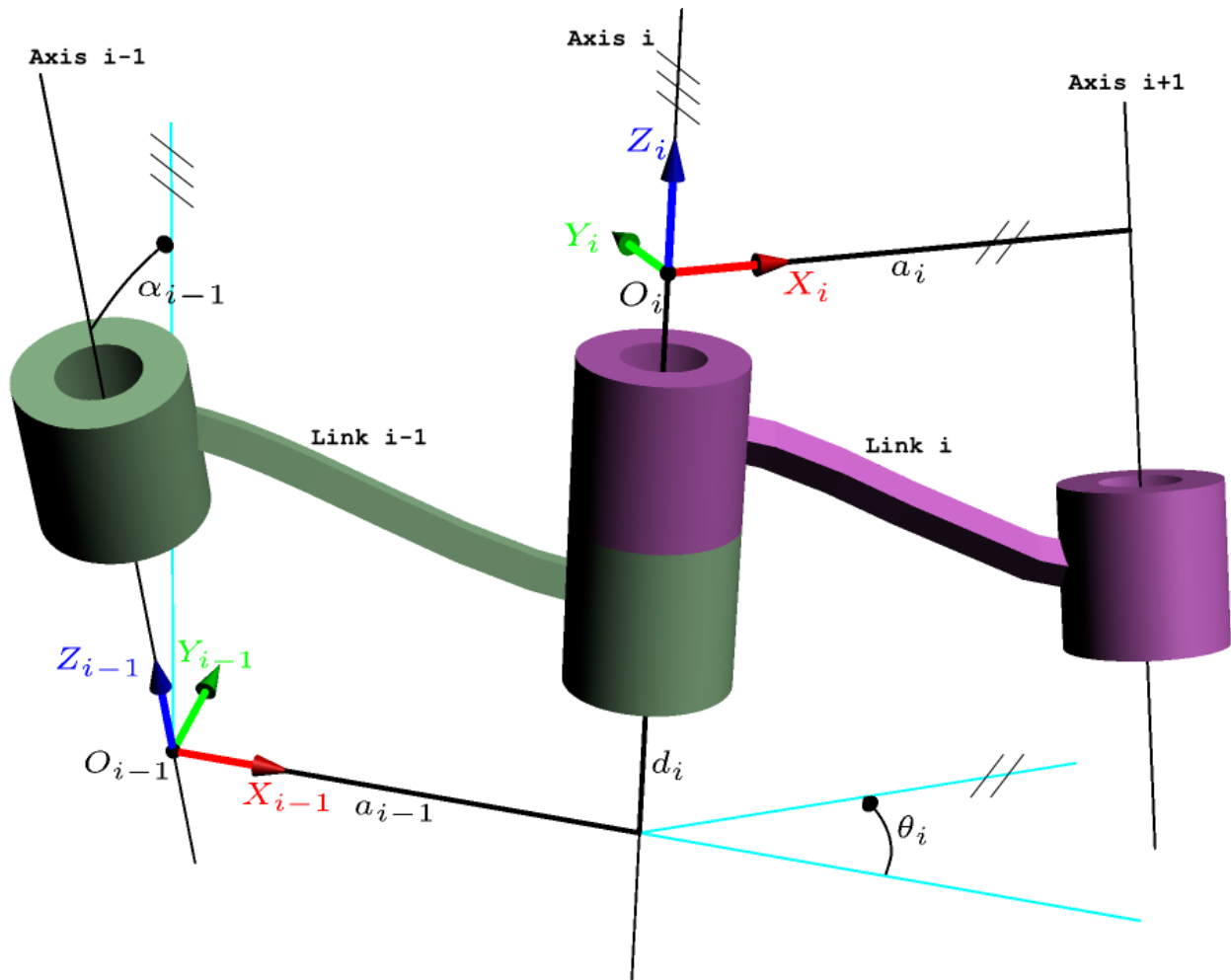


Figure 4.10: Four parameters to translate the coordinates. The figure is borrowed from Wikipedia: DH Parameters (Accessed 5/5/2020)

axis of the steering joint while keeping the same design of the driving joints, Romeo simplifies the number of possible configurations to three. Then for each configuration, Romeo samples value at each joint within its range to calculate the workspace and compare with each other to pick the workspace closest to the motion points.

To compute the workspace, we first introduce modified Denavit–Hartenberg (DH) parameters [70]— four parameters (a , α , d , θ) in mechanical engineering to represent spatial linkage systems, such as a robotic arm (Figure 4.10 indexes each parameter). With DH parameters, the position and orientation of the i^{th} joint relative to the $i-1^{\text{th}}$ joint can be

represented as a transformation matrix (c for \cos , s for \sin):

$${}^{i-1}_i T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, the cartesian position and orientation of the end-effector can be computed by ${}^0_N T$ using forward kinematics:

$${}^0_N T = {}^0_1 T {}^1_2 T {}^2_3 T \dots {}^{N-1}_N T$$

In our case, upon the transformable part selection, Romeo first generates a robotic arm with seven joints representing the three configurations in a single form of a robotic arm. Three joints correspond to the steering joint for each configuration, while another three joints represent the driving joints. The 7th joint represents the end-effector. With the list of DH parameters, Romeo computes the position and orientation of the end-effector by sampling values at each joint. For each configuration, Romeo only changes the value of the corresponding steering joint while keeping the other two constant.

To determine the best configuration for the user-defined task, Romeo takes the sampled end-effector positions and orientations as a workspace (represented as its convex hull as depicted in Figure 4.11) and calculates the minimal distance to the motion points. If there exists the orientation requirement, Romeo selects end-effector position only within the specified orientation range. Finally, Romeo picks the configuration whose distance to the motion points has the minimal RMSE (Root-Mean-Square-Deviation). In the following, we discuss details about arranging the steering and driving joints in distinct cases based on the resultant robotic arm:

(i) *Unfolded robotic arm* For object in a slender shape, the generated embedded robotic arm is already unfolded at its initial state, *e.g.*, spatula (Figure 4.12a). Therefore, Romeo

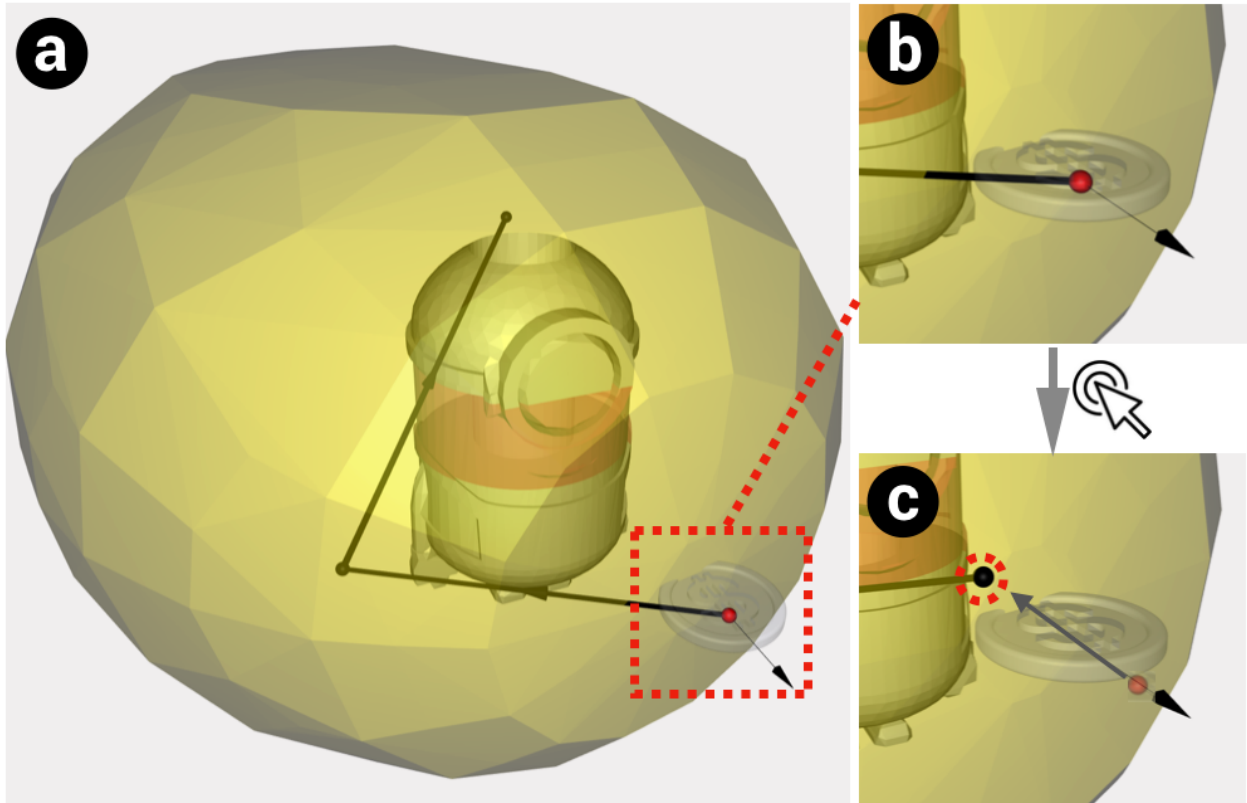


Figure 4.11: Visualized workspace (a). Double clicking makes the motion points outside, highlighted in red (b), snaps to the closet workspace (c).

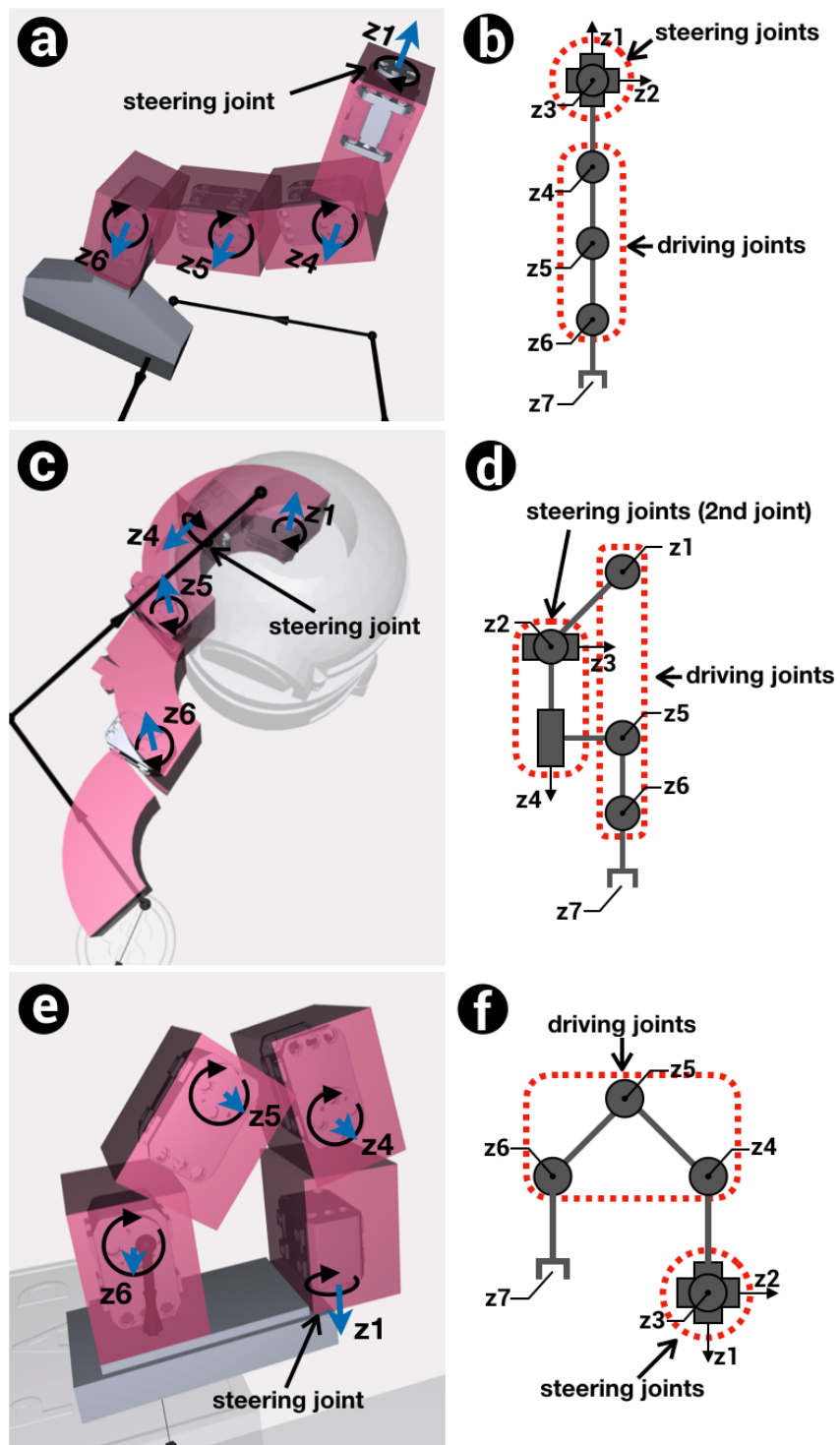


Figure 4.12: Corresponding robotic arm representatives and the joint placement for three distinct cases.

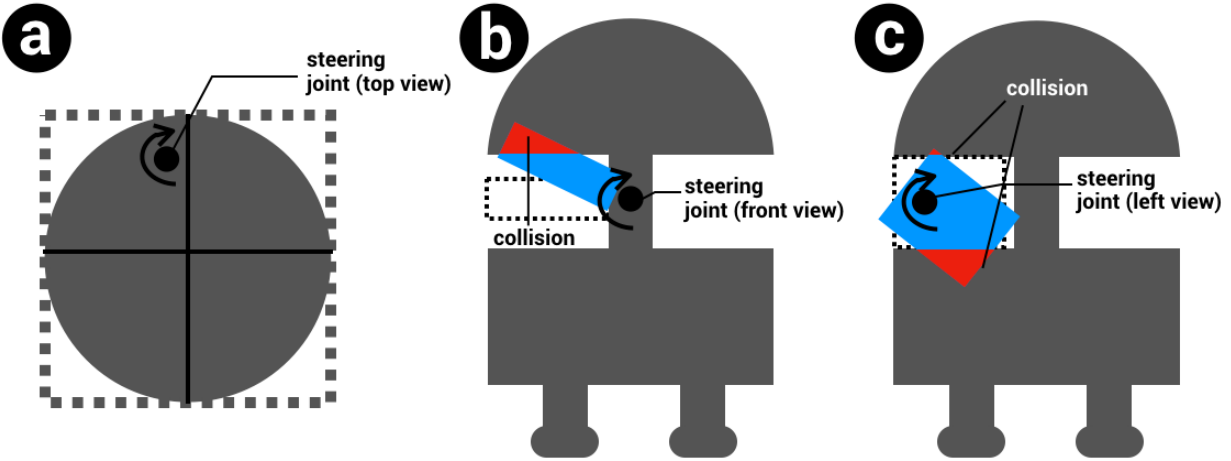


Figure 4.13: If the steering joint locates at the first joint while the transformable part lies in the middle, both cases of b and c will cause collision between geometries.

places the steering joint at the location which is farthest to the end-effector (Figure 4.12b).

(ii) *Folded robotic arm/End-effector on transformable part* For objects that have a non-slender shape and the end-effector is on the transformable part, the initial state of the embedded robotic arm is folded, *e.g.*, a piggybank (Figure 4.12c). Romeo places the steering joint at the second joint from the static part (Figure 4.12d), because if transformable part locates in the middle of two static parts, it is likely cause collision with the object when unfolding, due to the steering joint placed at the first joint (Figure 4.13).

(iii) *Folded robotic arm/End-effector on static part* For objects with a non-slender shape but the end-effector is not in the selected part (*e.g.*, See stamp in Figure 4.12e), the initial state of the embedded robotic arm is set to the folded state. Romeo places the steering joint to where it is farthest from the static part when the transformable part is unfolded. The steering joint will be fixed to the ground, so to become a base of the generated arm (Figure 4.12f).

Please refer to Appendix for the complete DH tables. **Other search criteria** include the following.

Matching the orientation When searching for the best configuration, we also need to

match the orientation if it is specified at a certain motion point. Romeo uses the X-axis of the end-effector joint as it matches the pointing direction. The pointing direction of the stamp is represented by the X_7 axis (as illustrated as a red arrow in Figure 4.14a), which is used to match the user-specified orientation at the motion point (black arrow in Figure 4.14b). Recall the transformation matrix of end-effector relative to the base ${}^0_N T$:

$${}^0_N T = \left[\begin{array}{ccc|c} n & o & a & t \\ \hline 0 & 0 & 0 & 1 \end{array} \right]$$

in which n is a normalized vector ($n \in R^3$) representing the direction of X-axis of the end-effector. Take the users specify direction as dir and the positions of the end-effector whose orientation matches $|n - dir| \leq 0.5$ are included to compute the minimal distance to the motion points.

Obstacle avoidance The collision between the transformed robotic arm and the remaining object itself (mainly the static part) is a significant factor that determines the performance of the robotic arm. Currently, Romeo defines the bounding box of the static part(s) as an obstacle. While searching for the best configuration, Romeo eliminates pose options of the robotic arm in which a joint position intersects the obstacle’s bounding box.

Users’ modification After the configuration of the robotic arm is determined, there could be cases where some of the motion points still situate outside of the workspace. To address this, Romeo detects if the point is inside the workspace by forming a convex hull of all the sample points and highlighting exterior points in red (Figure 4.11b-c). Then Romeo enables users to double click on such points, which snaps it to the nearest points on the workspace surface.

#4 Generating Components for Fabrication

As the final step, Romeo automatically generates 3D printable components. To generate assembly-ready components, Romeo performs a series of Boolean operations to create space

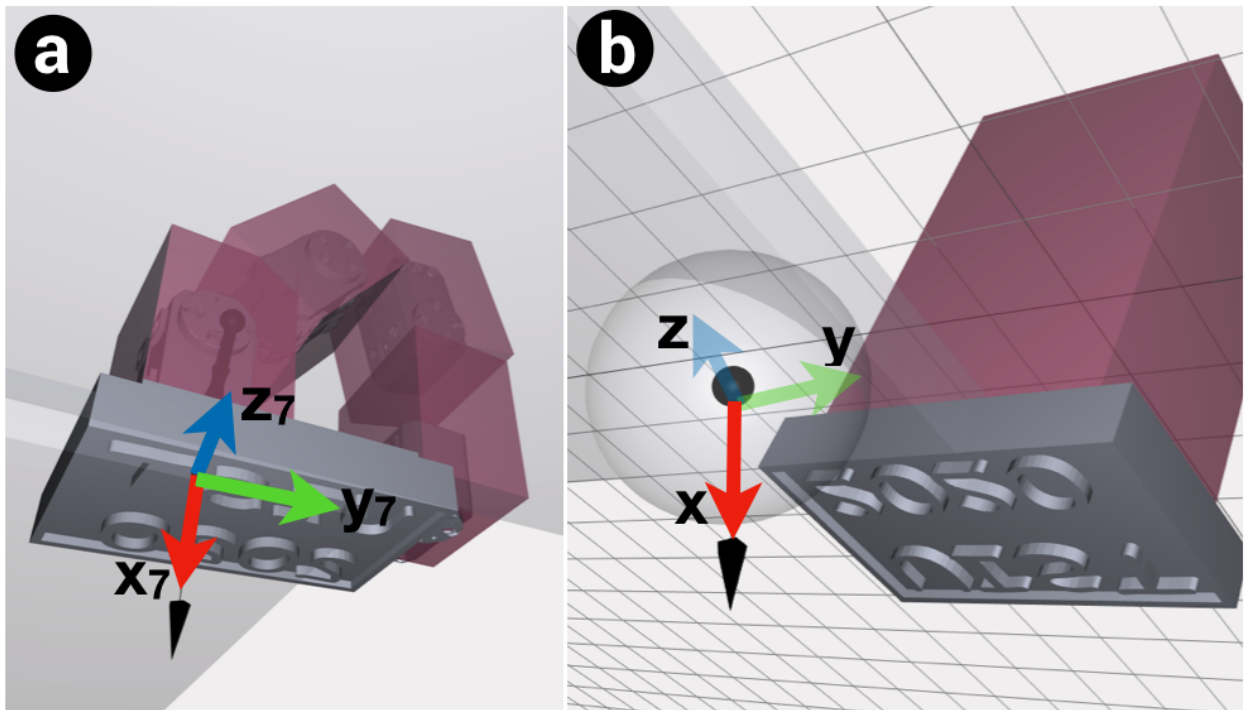


Figure 4.14: Matching the orientation by aligning the x-axis of the end-effector with the arrow direction of the spherical widget control. The coordinate system represent the target orientation of the end-effector (a). Black arrow indicates the specified orientation (b).

for motor fasteners, joint connectors, screw holes and end-effectors.

Motor fasteners Romeo allows users to choose whether to embed motors when generating the embedded arm. For manual transformation, Romeo generates hinges between each link. For motors integration, the system computes each motor’s pivot position. The links connect directly to the motors without any power transmission system (*e.g.*, gear system, four-bar-linkage system). Therefore, pivot positions are set in a way that it prevents links from colliding with each other, as shown in Figure 4.15a. Based on the pivot position, Romeo adds a ‘shell’ to mount the motor.

Joint connectors Romeo creates two types of joint connectors. Type A (Figure 4.15a) is used when the two connecting links move in the same plane, while type B (Figure 4.15b) is used when there is an offset between the two moving plane. For type A , the system fillets on the link corners to prevent self-collision and to secure spaces for the joint connectors to move. For type B, Romeo creates a fastening mechanism using screws.

Screw holes Screws are needed to fasten motors and joints with the links, thus, Romeos adds screw holes as shown in Figure 5.16a. The system creates rivet holes specifically for Dynamixel XL-320 motors², although other types of motors can be supported in future work.

End-effectors For motion points that include ‘picking and placing’ action, Romeo generates a robotic gripper as the end-effector which will need additional motor to actuate it. We currently provide one universal gripper design that can be directly imported into Romeo (Figure 5.16b). For ‘attaching to a surface’ action, Romeo employs AutoConnect’s approach [103] to generate clamp fasteners to be associated with the different types of attaching surface.

4.4 FABRICATION AND SOFTWARE IMPLEMENTATION

Romeo’s front end is written in JavaScript using jQuery for UI development, three.js for 3D graphics, and ThreeCSG for progressively generating the geometry of components. The

²<http://www.robotis.us/dynamixel-xl-320/>

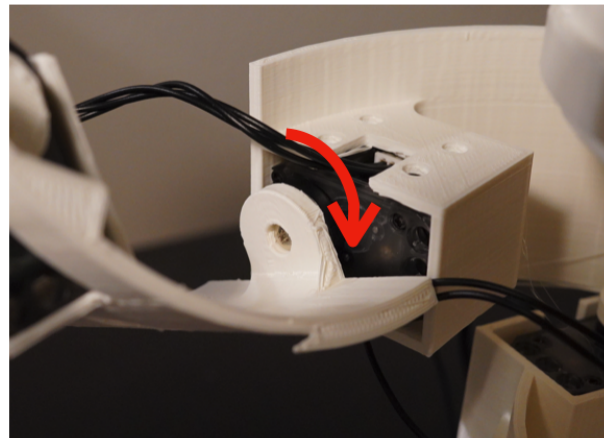
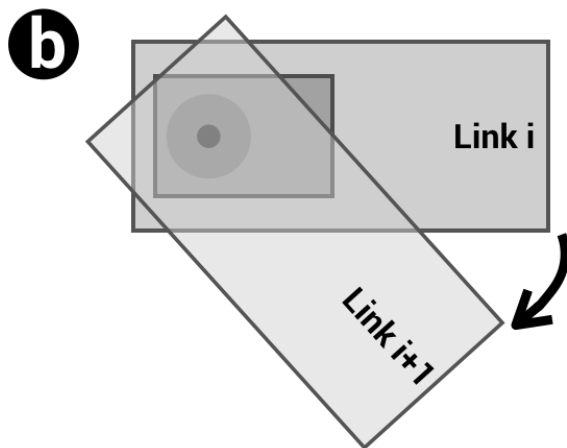
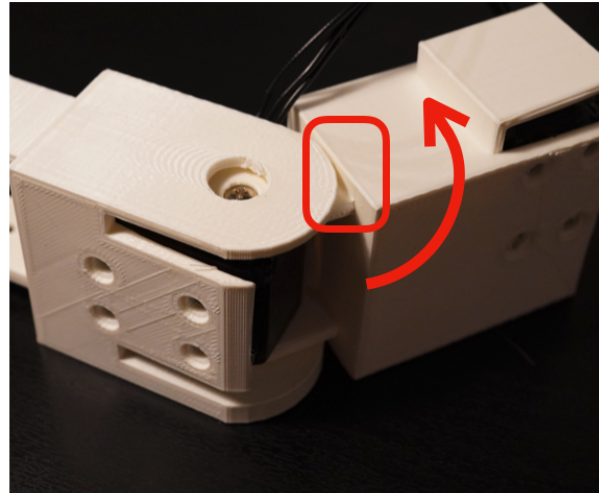
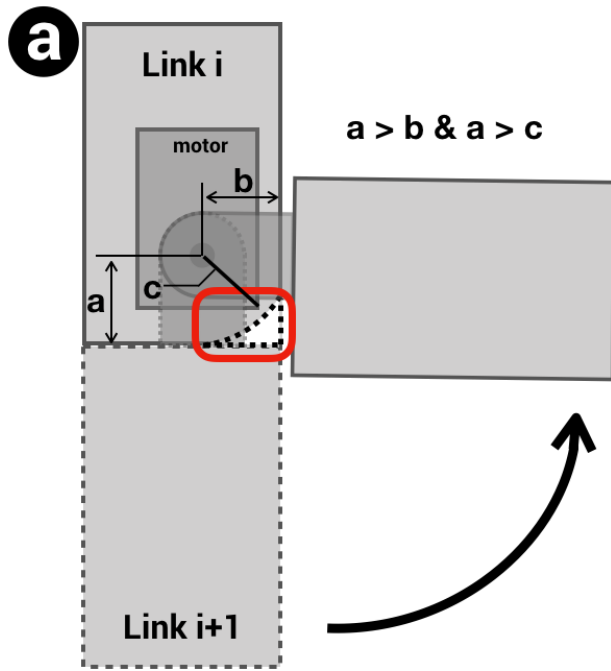


Figure 4.15: Two types of joint connectors.

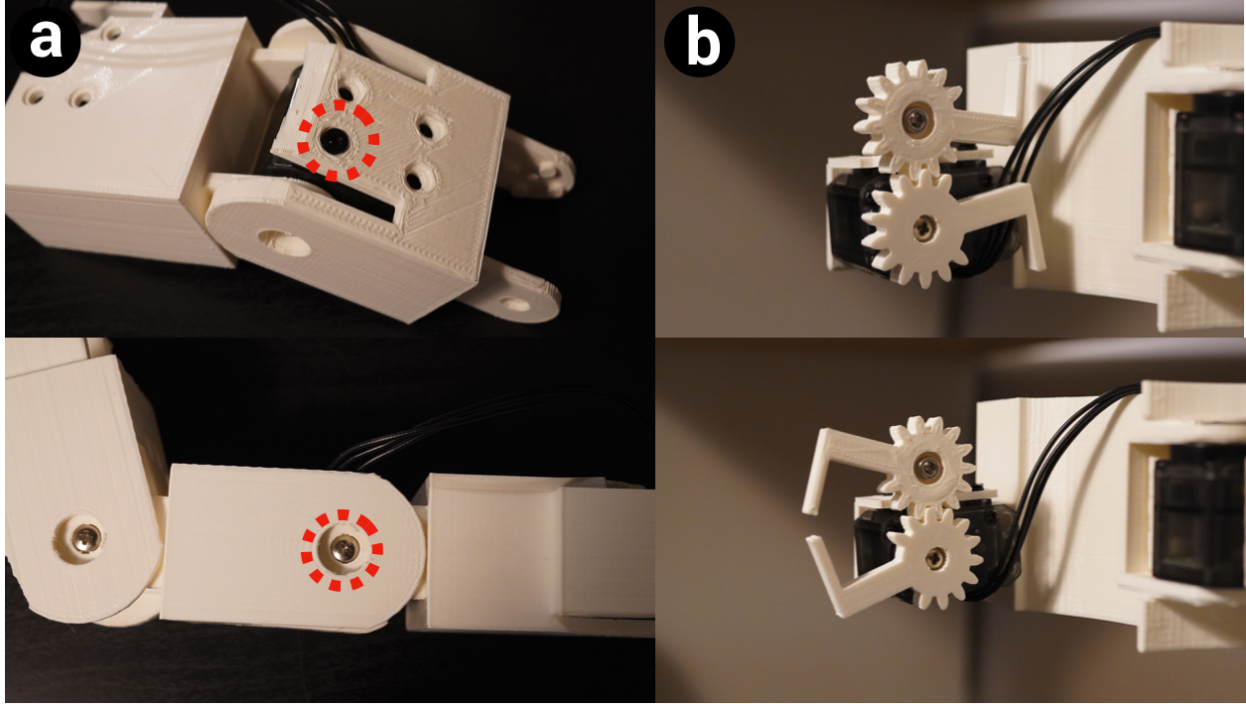


Figure 4.16: Romeo generates screw holes (a) and gripper (b).

back end is written in MATLAB 2018a. Everything runs on a MacBook Pro (15-inch, 2016 year) with a 2.7 GHz Intel i7 and 16GB 2133 MHz LPDDR3 memory. In our design session and demonstrations, the interface runs on a Google Chrome web browser. For remote participants, we use TeamViewer³ for screen sharing and remote control. We use Dynamixel XL-320 motors to actuate the transformed objects, all parts are 3D printed by the Ultimaker S5 using white PLA.

4.5 Examples Generated using Romeo

We present a series of examples created by Romeo. As we focused on demonstrating Romeo's potential to augment objects' functionality, we used other CAD tools for non transformable-related postprocessing, *e.g.*, making a fastener to affix a spatula to the side of a pot, adding a small water tank to the flowerpot for keeping water.

³<https://www.teamviewer.com/en-us/>

Figure 17—23 showcase exemplary augmentation of daily objects by transforming part of the objects into a robotic arm: A cup holder’s lower half can be (manually) extended to attach to a chair to free the user’s hands (Figure 4.21). A paper towel stand unfolds its base to hold the paper towel helping user pull apart a sheet (thus avoid touching the roll using wet hands) (Figure 4.17). A spatula can be robotized so that it is affixed to the pot’s handle and to automatically stir the pot to free a user’s hands (Figure 4.19). A conventional stamp with an UIST 2020 logo can be robotized to become a self-inking stamp (Figure 4.18). For a busy office worker who is likely to forget to water the plant, the flowerpot can be transformed to water itself regularly (Figure 4.22). Under the contagious COVID-19 pandemic, a box of sanitizing wipes can disinfect the doorknob every time someone touches the doorknob (Figure 4.20).

Figure 5.1d and Figure 4.23 depict examples with different 3D models transformed into a ‘stealing piggybank’ using Romeo.

4.6 Design Sessions

We conducted informal, qualitative design sessions to validate Romeo’s ability to support non-expert users’ design of transformables using existing 3D models.

4.6.1 Participants

We recruited eight participants (aged 23-26, female=5, male=3). No participant had background in Mechanical Engineering. Five participants had Electrical and Computer Engineering background, two of which had experience in the computer vision area of Robotics. The rest Participants did not have background in engineering. Amongst all participants, three had experience using CAD tools before.

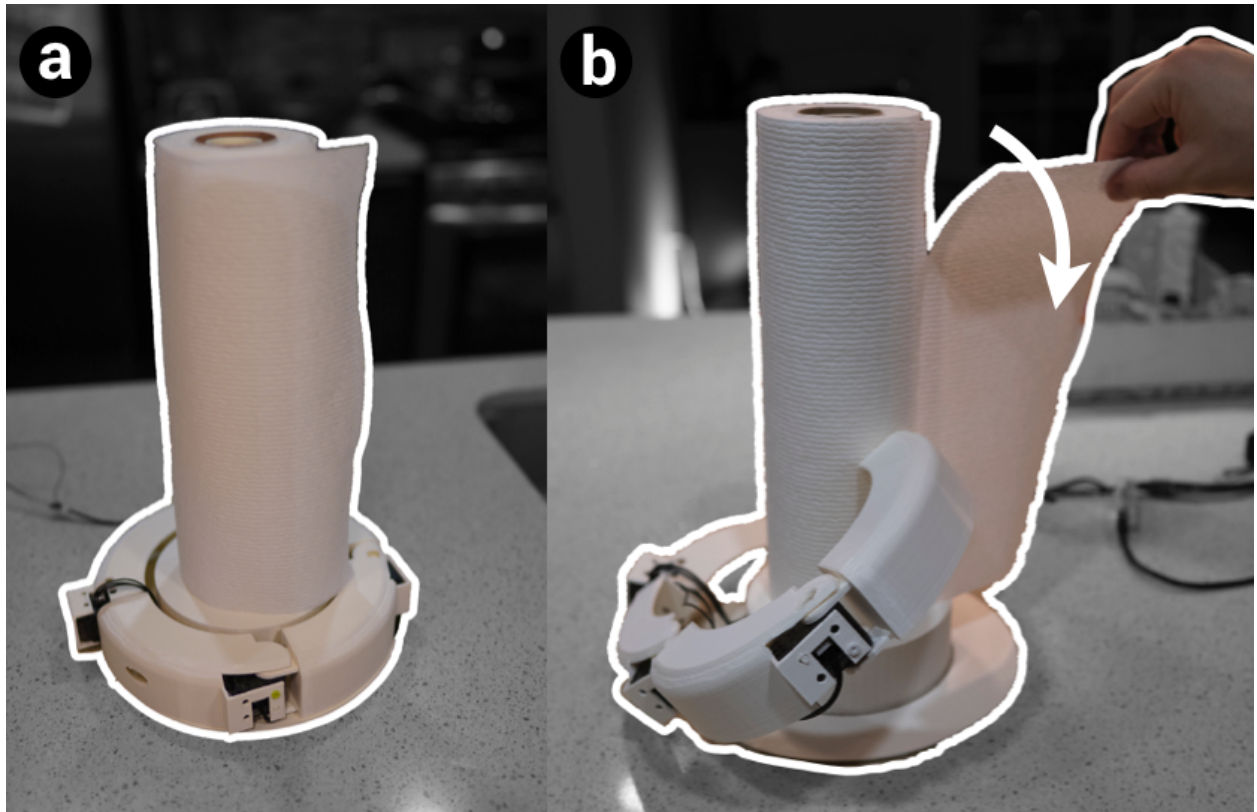


Figure 4.17: By robotizing a paper towel holder, a user can pull one sheet without contaminating the others.

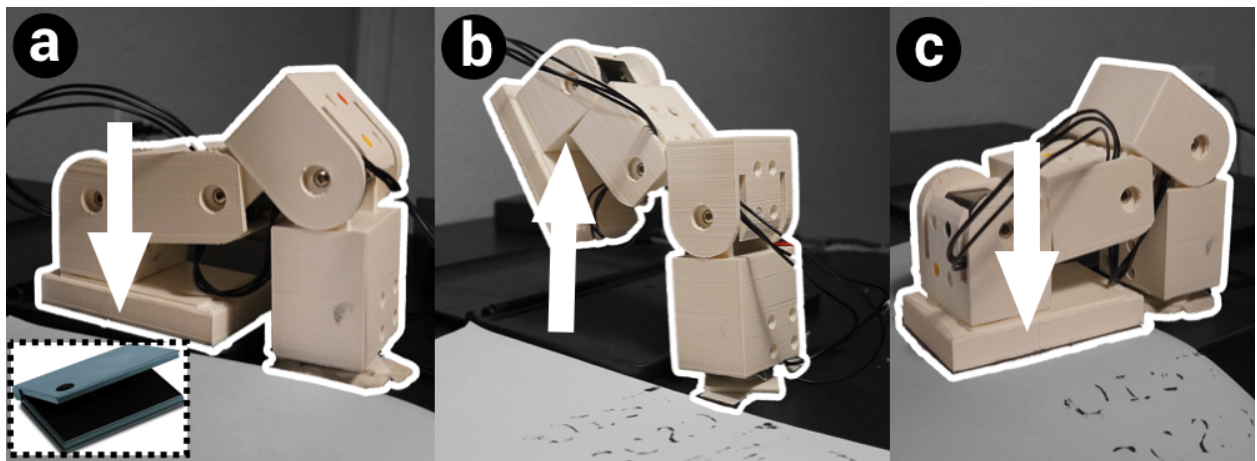


Figure 4.18: A conventional stamp can be robotized by Romeo to become a self-inking stamp.

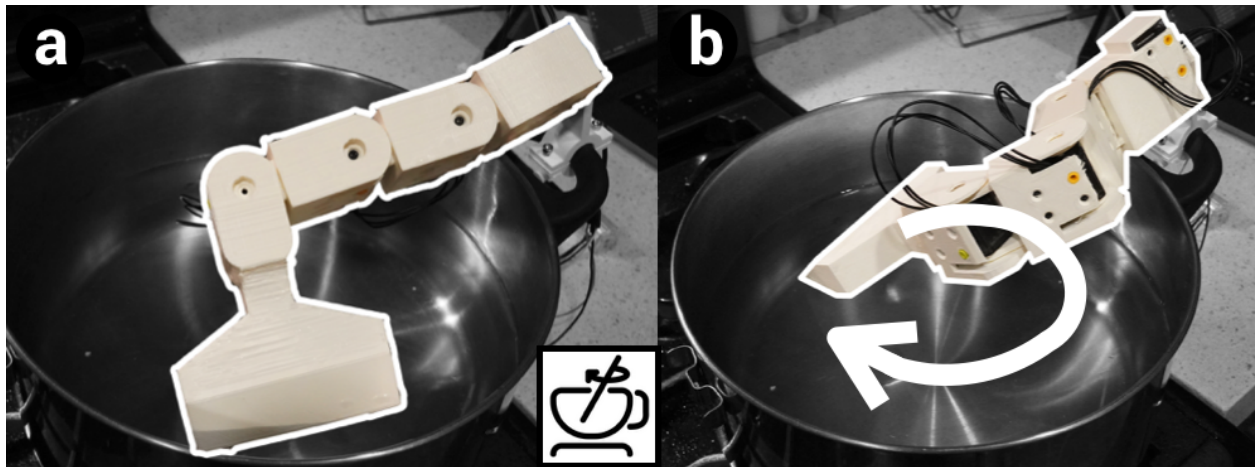


Figure 4.19: Romeo can robotize a spatula to automatically stir the pot when unattended.

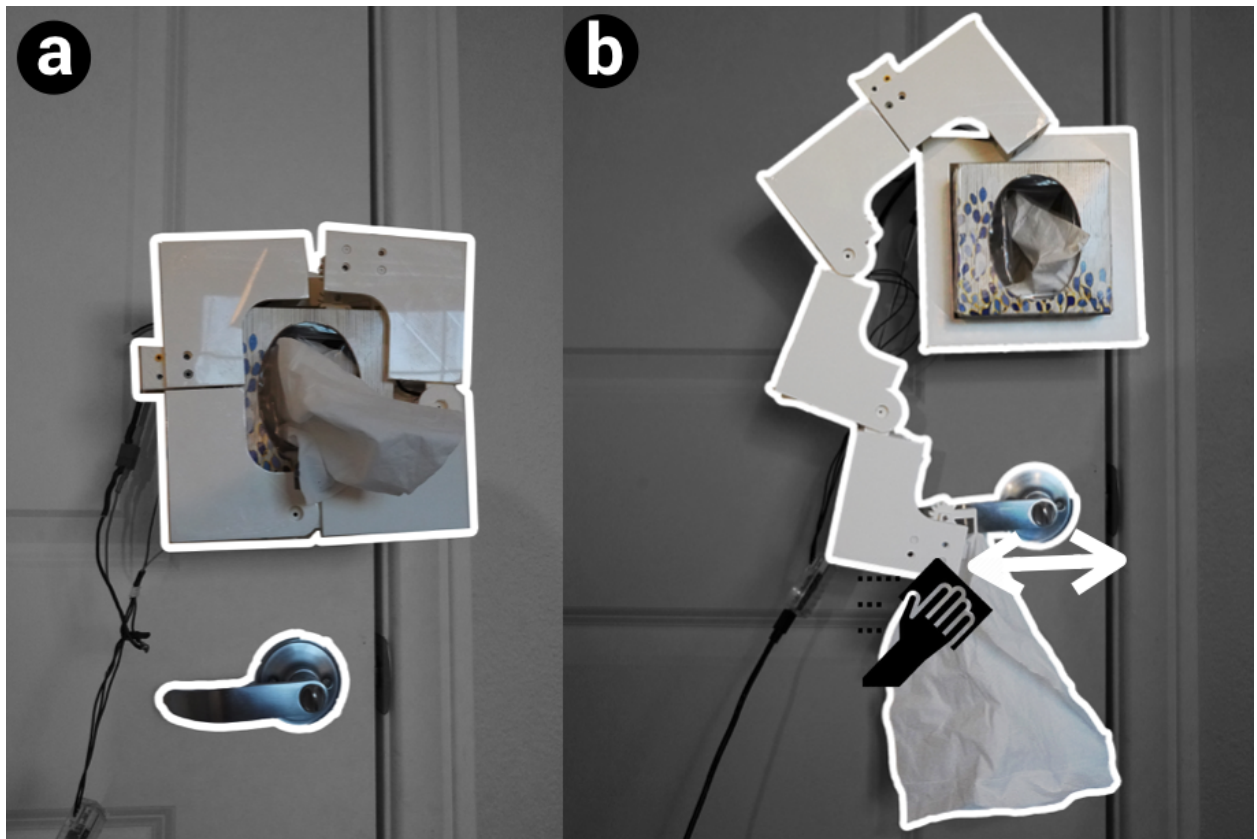


Figure 4.20: A tissue box can be stick to the door and wipe the doorknob every time someone opens the door.

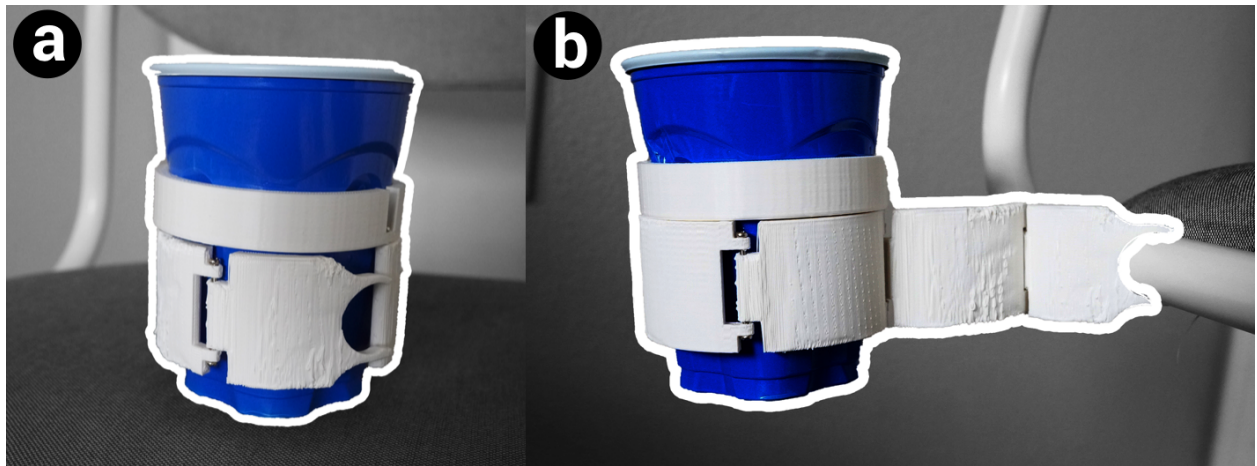


Figure 4.21: A cup holder can extend the lower part of it to attach to the chair.

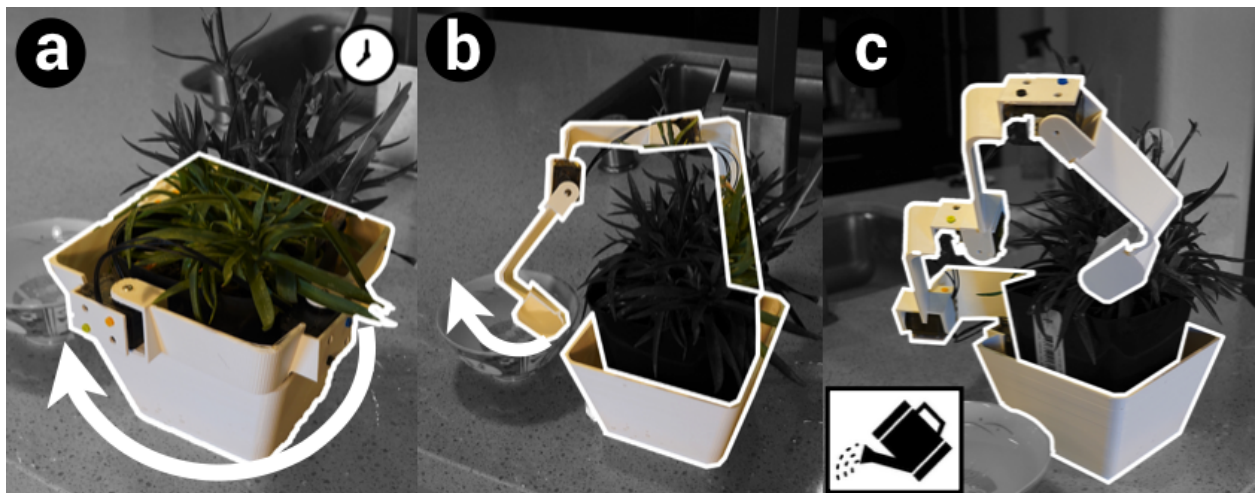


Figure 4.22: A flowerpot can be robotized to automatically water the plant according to a predefined schedule.

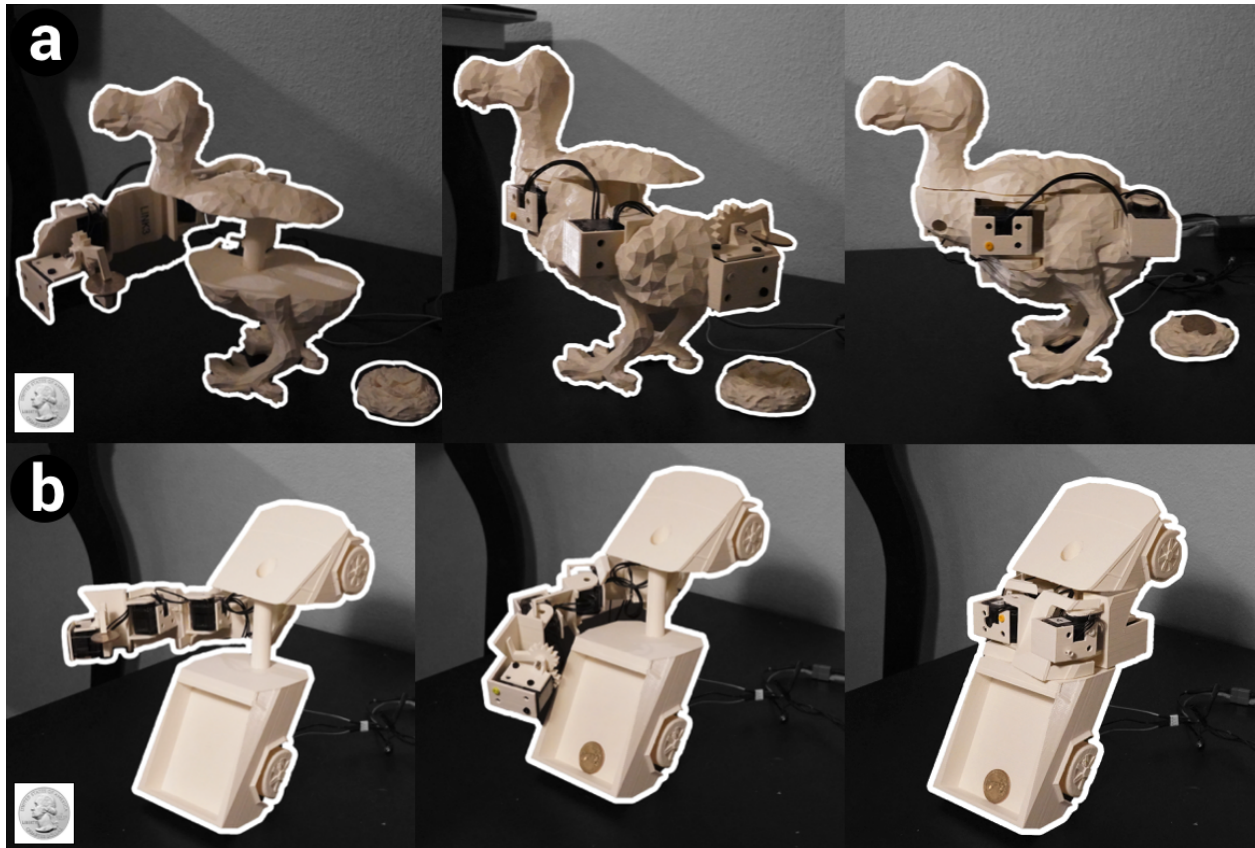


Figure 4.23: Different 3D objects can be augmented into a stealing coin piggy bank using Romeo. The examples are made by the participants in the design sessions: a dodo bird (a) and a Tesla cybertruck (b)

4.6.2 Apparatus, Tasks and Procedure

We split design tasks in two sessions spanning two days to budget time for 3D printing the resulting transformables. In the first session, participants designed transformables via TeamViewer’s remote desktop to interact with Romeo running on the experimenter’s computer. The components were then 3D printed for assembly, tested with its designed task in the second session on the next day. However, due to the COVID-19 outbreaks, only four participants were able to meet in person to join the second part for assembly and test the fabricated results. Therefore, alternatively, the results of remote-only participants’ designs are assembled and tested by the experimenter and the process is recorded to be shown to these participants.

Design Session (Day 1) To start, each participant was introduced some basic knowledge of how an robotic arm works (*e.g.*, how a 6-DOF arm moves along with rotary joints) and how to use Romeo by walking through an example of making a snowman-shaped coin-stealing piggybank. Then participants proceeded to use Romeo for one controlled and one open-design tasks. Each task’s objective is using Romeo to augment an object by creating transformable parts to add a new functionality. In the first (controlled) task, the participant was to replicate either the spatula (Figure 4.19) or the tissue box examples (Figure 4.20). Specifically, the spatula would be transformed to stir in a pot and the tissue box to pick a tissue and wipe a door knob. In the second open-design task, the participant used a 3D object of their own choice to design a coin-stealing piggybank.

After the first session, we manually post-processed the resulting components, creating holes for cables and hollowing internal volume to reduce printing time.

Assembly & Interaction Session (Day 2) For the four participants who were able to join the in-person assembly session, they were given instructions (Figure 4.24b) based on which they assembled the printed parts. The logged data of their design were fed into a universal code for actuating the motors. Then the participants test the automated tasks, if they match with the simulated animation.

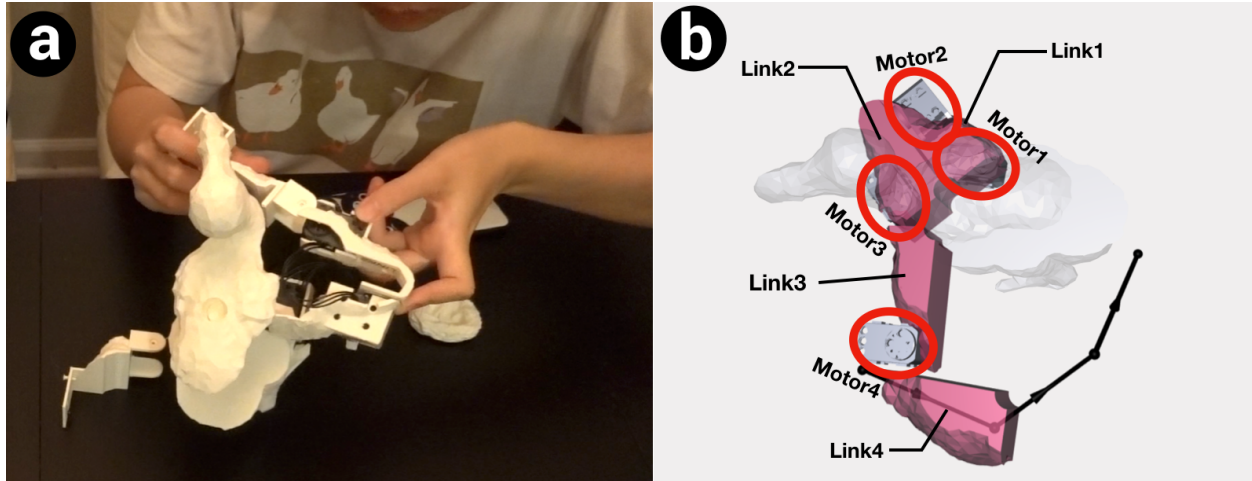


Figure 4.24: A participant installs the printed components (a) following a provided instruction (b).

Metrics & Measurements At the end of the Day 1 session, participants filled out a questionnaire regarding overall user experience, including the difficulty to learn how to use Romeo and whether the process required extra knowledge than what they had expected (in Likert scale of 1-7). After the Day 2 session, the four participants filled out another questionnaire to answer how difficult it was to assemble the fabricated results, whether the installed mechanisms behave as they expected, and perceived usefulness of having such augmented transformable objects. Finally, we conducted a brief interview and solicited feedback about the entire process and suggestions.

4.6.3 Results

Overall, during the first session of using Romeo to create transformable designs, participants were able to complete the controlled and open-ended tasks. During the second session with four participants, they were able to complete assembly given the instruction, linking parts in order and connect motors in the right orientation. They were mostly satisfied with the design flow and underlying automation that aided the design and assembly (Satisfaction score in Mean=5.25–6.75/7 across questions). See Figure 4.25 in Appendix for quantitative summary of the result and Figure 4.26 for all participants design.

4.6.4 Observations & Findings

We present the findings by questionnaire responses, analyzing the logged observation of participants’ behaviors and spoken responses during the design session as follows.

Users enjoyed the simplicity of Romeo’s design process with automation, noting that it is very simple with only four steps (P8). Participants seemed to highly value the automated mechanism generation, as it removed the needs in engineering expertise (P4) and reduced the required intellectual efforts (P3)—“After I specify points, Romeo helps me do the other things even though I don’t have any related background” (P1). “[There was] No need to consider about the transformation and movement design. All I need is a clear goal” (P4). Although Romeo requires some manual adjustment if the design is not viable (*e.g.*, double-clicking an outside point to snap it back to the workspace) participants felt the overall process is “completely automatic” (P6-7). Participants also appreciated the animation function that helps visualize and validate resulting movement, showing their satisfaction about the simulation of a designed task. “It’s very clear to help me understand how the movement is achieved” (P1).

However, since Romeo is targeted for novices and does not require any pre-knowledge about mechanical engineering and robotic design, the participants have some confusions in handling interfaces during the study. The major challenge seems to be understanding where to place the end-effector. As introduced in the earlier section, users can place the end-effector either on transformable part (as in the piggy-bank model, Figure 4.23) or on the static part (as in the spatula example, Figure 4.19), but some users have difficulties in determining which of these two approaches to use. “ I can try either one to decide which is right” (P1).

Participants also had difficulties in understanding how a transformable will be anchored, *i.e.*, where the base is. For example, P1 selected only the middle part of the spatula handle in the first trial, thinking she needs to leave the top part to serve as the base that anchors the spatula while stirring. Another participant also questioned “If I specify the middle part, will the top part move with it?” (P8). However, after seeing the generated results, participants naturally understood the consequence, becoming able to select the placement of end-effector

in the subsequent tasks: “I did not understand why we did them. I understood it after couple of examples though!” (P2).

Such confusion was partial result of our design choice, abstracting away low-level details for users, while allowing them to rapidly explore different designs and see results, so to iterate on their earlier decision (*e.g.*, where to place the base/end-effector). We can also support users to explore different possibilities simultaneously to choose between different results. Further approaches to solve this problem are discussed in the next section.

* 1: Strongly disagree – 7: Strongly agree							
1	2	3	4	5	6	7	Mean
DQ1. It is easy to understand how Romeo works using the snowman example							5.75
			1	1	5	1	
DQ2. It is straightforward to follow the romeo design flow from the interface							6
				2	4	2	
DQ3. I found it easy to specify which part of the object to be transformed							5
	1		2	2	1	2	
DQ4. I found generating a robotic movement easy once parts and points specified							6.25
		1		1		6	
DQ5. The visualization and animation helped you understand and validate the designed movement							6.75
				2	6		
DQ6. I think I can independently create a custom robotic object using Romeo in the future							6.125
				2	3	3	

* 1: Strongly disagree – 7: Strongly agree							
1	2	3	4	5	6	7	Mean
AQ1. It is easy to assemble the components given the instruction.							5.25
			1	1	2		
AQ2. The installed transformable object behaves as what is shown in the animation.							5.5
		1			2	1	
AQ3. The installed result is able to finish the task you designed.							5.5
			1	1	1	1	

Figure 4.25: Participants’ rating on the Romeo assessment. DQ refers to questions asked in the design session with 8 participants and AQ refers to questions asked in the assembly session with 4 participants

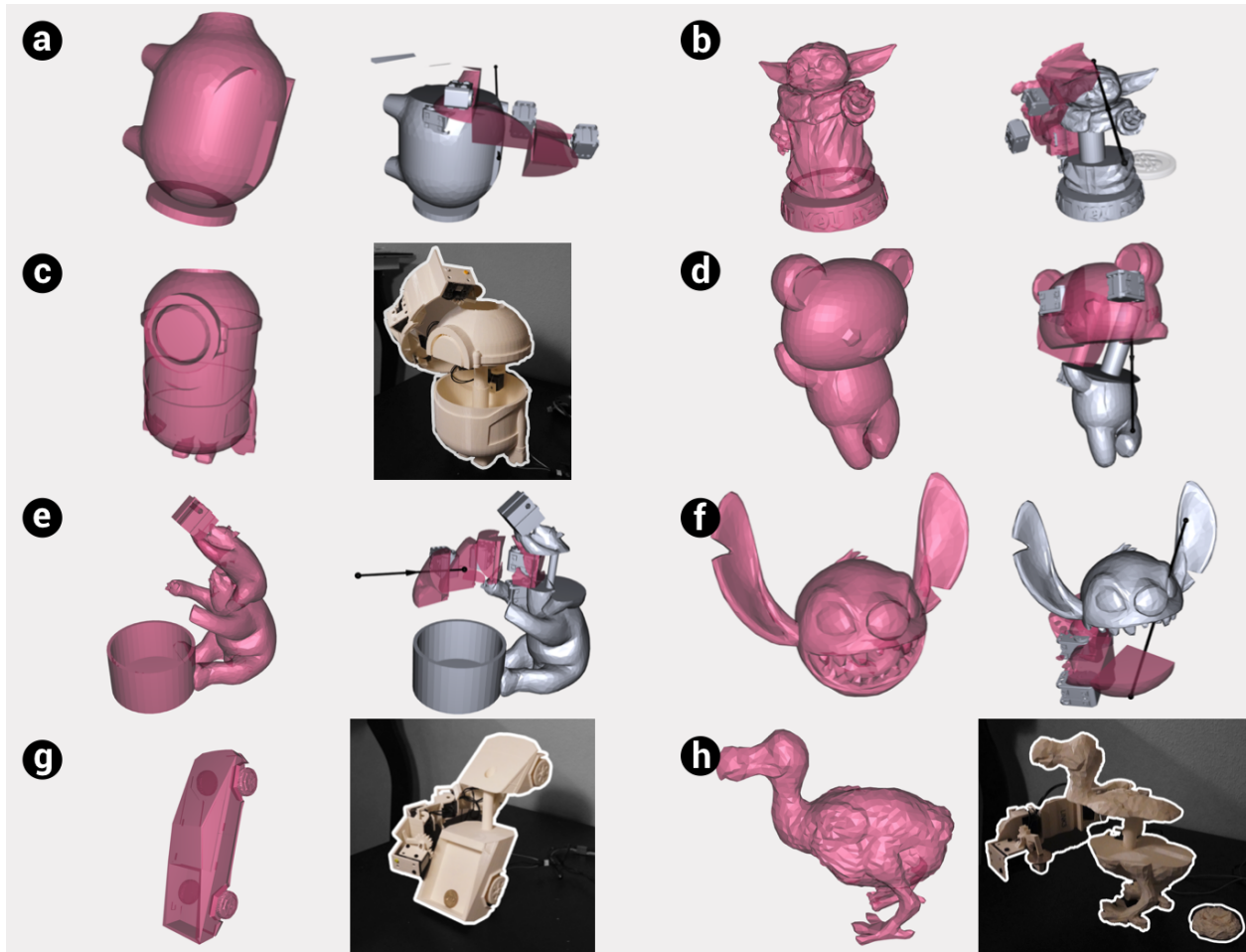


Figure 4.26: Participants' design of coin-stealing piggy bank of their chosen 3D objects: (a) piggy bank, (b) baby yoda, (c) minion, (d) rilakkuma, (e) polar bears, (f) stitch, (g) Tesla cybertruck and (h) dodo bird. (Original 3D models are designed by Thingiverse users: layerone, MarVinMiniatures, sota919, Anthonylu, MakerBot, Erinfezell, wov, stargatedalek

4.7 Discussion, Limitations & Future Work

Scale of Mechanism One of the limitation is that Romeo requires the input 3D models have sufficient volume to embed the motors in each link. Further, the scale of an input 3D model also determines the possible range of movement of a robotic arm transformed from part of the object. As a next step, Romeo can support automatically scaling a 3D

model to meet both the requirements of housing motors and accommodating sufficiently large workspace, while estimating the printing time/cost to help users make an informed decision whether to use the scaled model. Yet, even with the sufficiently large initial volume of an input object, it is possible that a user selects too small parts to embed transformable mechanisms, as also questioned by one of our participant: “If I only want to transform ears, will that be too small?” (P7). Guiding users with alert message to notify that the parts need to be larger, and automatically adjusting to the minimal-scale would aid users to avoid any potential design errors.

Challenges in Selecting a Transformable Part To simplify the design process, we currently allow users to specify a transformable part by sweeping an object’s cross section along one of X/Y/Z axis, in other words, Romeo may not have good performance on irregular shapes such as bent or twisted objects. Future work should explore techniques for more expressive selection of transformable parts in order to support cases such as selecting the face of a bear, the wing of a bird, the arm of a minion character. One possible solution is to include functions provided in existing CAD tool such as a brushing interface to select parts by painting areas of interest and automatically offsetting enough volumes [222] or automating complex mesh-segmentation with semantic segmentation [89]. Also, users’ selection of transformable part may affect the object’s original functionalities. Future work may enable highlighting the original functional part of the object using a machine learning model to guide users not to select that part as transformable. Further, users’ selection of transformable part may cause separation if the object has concave part or hollows (*e.g.*, selecting top half of the head of stitch in Figure 4.26f), jointness detection can be part of the future work to eliminate the unexpected disjointed parts.

Error Detection and Automatic Geometry Processing Current segmentation algorithm has good performance if the transformable part has convex and solid shape but it might cause disjoint links if the transformable part has high concavity such as branching shapes. Future work can detect the concavity of selected transformable part and explore an adaptive segmentation method to avoid creating disjoint components. Romeo automatically

generates the motion path for conducting the tasks, however, without detecting the collision between the motion points, which may result in collision as also observed in one case in our assembly session. Furthermore, while detecting the collision of the motion points, Romeo only considers the bone without the skin of each link, which may result in self collision between links. An interesting future direction could be applying trajectory planning algorithm and obstacle avoidance of robotic arm to generate collision-free motion path and applying the skin collision analysis of each link to achieve optimized bone embedding.

Desired Task Specification Assigning task by each point appears unintuitive to some users. P4 asked “Do I need to specify the action for every point?”, and P8 thought there would be some high-level task description to select, questioning “Is the following a trajectory action means the action of wiping?”. Future work can provide a user with an option to specify the task type upfront and provide more details on demand (*e.g.*, at which point to pick/place) and real-time animation can also provide users with more intuitive feedback. Another interesting possibility is allowing users to describe a high-level task and search in a library of existing transformable designs that meet such requirements.

Pre/post-processing Currently, an input 3D model needs to be pre-processed to simplify meshes, to align the bounding box’s axis parallel to the coordinates, to rotate and locate it in a way that motion points can match the world-coordinate to aid the user’s understanding of its relative position to the target 3D object. To reduce weight, material cost and printing time, one common post-processing step is hollowing the components using existing 3D modelling tool. In the future, both pre- and post- processing steps can be engineered into Romeo for better integration.

Part II

CHAPTER 5

Making Physical Objects Robotically Manipulable with 3D-Printable Add-on Mechanisms

One important vision of robotics is to provide physical assistance by manipulating different everyday objects, e.g., hand tools, kitchen utensils. However, many objects designed for dexterous hand-control are not easily manipulable by a single robotic arm with a generic parallel gripper. Complementary to existing research on developing grippers and control algorithms, we present Roman, a suite of hardware design and software tool support for robotic engineers to create 3D printable mechanisms attached to everyday handheld objects, making them easier to be manipulated by conventional robotic arms. The Roman hardware comes with a versatile magnetic gripper that can snap on/off handheld objects and drive add-on mechanisms to perform tasks. Roman also provides software support to register and author control programs. To validate our approach, we designed and fabricated Roman mechanisms for 14 everyday objects/tasks presented within a design space and conducted expert interviews with robotic engineers indicating that Roman serves as a practical alternative for enabling robotic manipulation of everyday objects.

5.1 Introduction

Various types of robots that inhabit our living spaces, such as vacuum cleaners, robotic toys, home/office-assistive robots, promise to aid humans with numerous everyday tasks. Future robots are expected to provide physical assistance for the elderly at home or help with household chores that require the use of diverse tools and everyday objects [94].

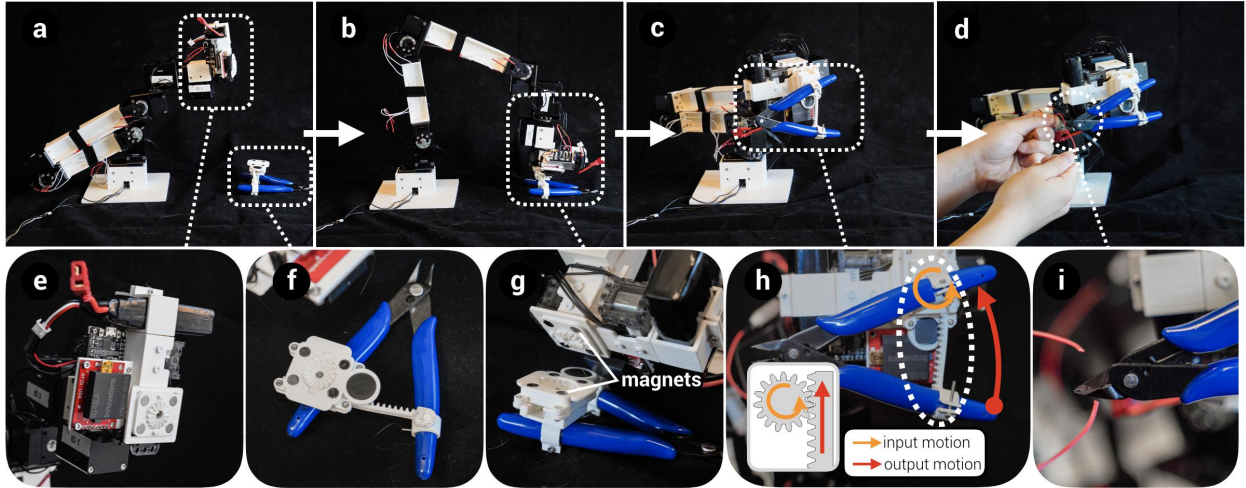


Figure 5.1: Roman is a novel robotic design making everyday handheld objects more robotically manipulable, *i.e.*, easier to be manipulated by a conventional robotic arm. This figure shows a sequence of a Roman-enabled robotic arm picking up a wire cutter on the desk and performing a wire cutting task collaborating with a human (a-d). Roman provides a magnetic gripper (e) for the robotic arm to easily attach and augment the wire cutter with Roman mechanism (f). Snapping in the mechanism using magnets (g), the gripper can actuate the gear-rack movement on the wire cutter (h) to perform the cutting task by squeezing the cutter’s handles (i).

Many recent advances in robotic research (primarily on manipulation [213, 136, 57, 215]) aim to solve robotic arms’ manipulation problem for niche, fixed tasks, *e.g.*, opening medicine bottles, handling cookware, and sorting waste. However, creating a *universally dexterous* robotic arm remains challenging as there are a large number of everyday objects that are not robotically manipulable, *i.e.*, difficult to manipulate by a consumer-level robotic arm with a generic parallel gripper, especially so for those *dynamic* objects that have multiple mechanically movable parts, *e.g.*, a pair of pliers and a spray bottle with a pump pushable toward the body.

To tackle this robotic manipulability challenge, prior work often considers the action of grippers and manipulators as completely decoupled, reducing the complexity of task planning and control involved [10]. To date, researchers have focused on developing manipulation strategies by analyzing the best grasping points of *static* objects [213, 136]. However, when it comes to *dynamic* objects such as a spray bottle with a pump pushable toward the body, prior work tends to mimic the pose and force of a human with *multiple* robotic arm [51, 126] or dexterous robotic hand [4] rather than tackling the task self-containedly with a *single* robotic arm. Meanwhile, on the objects’ side, augmenting objects with actuatable mechanisms is a new approach to enhance their interactivity or functionality (*e.g.*, [28, 118]); however, little has been explored on how to enable robotic arms to better manipulate such augmented objects.

We present Roman—a suite of hardware design and software tool for robotic engineers to make everyday handheld objects more Robotically manipulable by a consumer-grade 6-DoF robotic arm.

Roman’s hardware components consist of *(i)* a library of 3D printable powerless mechanisms that are attached to and can drive different handheld objects to perform specific tasks, *e.g.*, squeezing a cutter to cut wires (Figure 5.1a→d); *(ii)* a gripper (Figure 5.1e) that uses magnets to securely and automatically attach/detach a robotic arm to/from an object¹. The gripper also contains an RFID-based module to recognize which object the robotic arm is

¹Caveat: currently, an object’s placement needs to be known for auto-attachment

expected to manipulate thus to run the corresponding control program for specific tasks.

Roman’s software component is a user interface for robotic engineers to register and specify the motor input of custom tasks with pre-defined templates and real-time feedback. A user can rapidly and iteratively author a control program for a Roman-enabled robotic arm to manipulate a handheld object. Note that Roman does *not* reinvent tools for generating 3D models of add-on mechanism, which is already supported; rather, Roman’s software focuses on creating a control program to operate such mechanisms as different handheld objects require unique sequences of action to perform a specific task, which would otherwise be tedious to specify even for robotic experts.

To validate Roman, we first designed and fabricated mechanisms for 14 everyday objects to demonstrate how a generic 6-DoF robotic arm (built with 3D printed links and Dynamixel XH-540 motors²) can manipulate these objects to perform specific tasks. Further, we conducted a preliminary interview with four robotic engineers. Participants found Roman’s approach complementary to the current research on robotic manipulation within the human environment, which could be beneficial in specific task scenarios such as cooking, electronics assembly, and caring for house plants. Provided with the pre-fabricated gripper and attaching mechanisms, all participants were able to use Roman’s software module to replicate the wire cutter scenario (Figure 5.1).

Overall, Roman empowers robotics engineers to use 3D printable powerless mechanisms attached to different objects for enhancing these objects’ robotic manipulability with task-specific control program embedded in the mechanisms. Our specific contributions are as follows.

- **Categorization of the robotic manipulability problem in everyday handheld objects (§5.2)**, including the challenge of speed and range of motion, and the challenge of dexterity when manipulating an dynamic object with its constituent parts (squeezing, twisting, and pumping);

²<https://www.robotis.us/dynamixel-xh540-w150-t/>

- **A library of 3D printable add-on mechanisms able to manipulate both static and dynamic objects (§5.6.2)** based on variations and combinations of spur gears, bevel gears, gear racks, and pin-in-slot mechanisms, which can be attached to everyday objects to enhance their manipulability by a consumer-grade 6-DoF robotic arm;
- **The design of a versatile magnetic gripper (§5.6.1)** that can securely attach to the add-on mechanisms on an object, run the control program to drive the mechanism, and detach from the mechanism without outside intervention;
- **Software that enables robotic engineers to interactively author a program (§5.7)** to control the Roman mechanism-installed hardware to perform object-specific tasks.

At present, Roman has not achieved total autonomy, as there is no sensing modules to detect where an object is positioned or how it is oriented for pick-up (which is a separate topic well studied in robotics and computer vision [53, 63]). Currently, in most of our examples, the object is handed off to the robotic arm by a human (*e.g.*, kitchen utensils) or the position of the object is manually input to the robotic control program by the human user (*e.g.*, the spray bottle example).

5.2 Categorizing Challenges of Handheld Objects for Robotic Manipulation

Many everyday objects are not robotically manipulable due to the following challenges:

- **Speed and range of motion challenges** when manipulating an object as a whole. For example, objects that require a high speed/frequency of manipulation may exceed the motor’s capability at the end-effector, *e.g.*, a rapidly rotating whisker at a high speed to whip cream (Figure 6.11a). Meanwhile, manipulating a screwdriver at certain angles might exceed the reach of a situated robotic arm (Figure 6.11b)











generic 6-DoF robotic arm. To achieve this, we first need to understand *what* objects are currently inaccessible and *how* they are inaccessible for robotic manipulation. Everyday objects are mostly designed to be manipulated by humans. With flexible fingers and the coordination of four limbs, humans are able to manipulate a wide range of objects with different manipulation complexities, from picking up a cup to playing a piano. In contrast, there exist a large number of everyday objects that are not manipulable by generic robotic arms due to the fact that a general-purpose gripper has limitations in performing different types of grasping and manipulation tasks [10]. As shown in Figure 5.4, we consider two categories of such objects: those manipulated as a whole (*e.g.*, raising/lowering a knife, rotating a screwdriver) *vs.* those manipulated by constituent parts (*e.g.*, squeezing a pair of pliers’ handles, twisting a pepper grinder cap against the bottle, pushing the pump towards the body of a bottle of hand sanitizer).

5.2.1 Objects Manipulated as a Whole (Static Objects)

As shown in Figure 5.4A-B, objects in this category require a ‘grasp & move’ type of manipulation as the body of the object moves as a whole while performing tasks, *e.g.*, the whole knife moves vertically while chopping vegetables (linear motion), the whole egg beater rotates when mixing eggs (rotational motion). Although in theory such objects can be manipulated by a robotic arm—by first grasping it using a gripper and then performing manipulation by a series of joint rotations to create the movement, there remain two limitations that prevent the robotic arm from manipulating such objects to perform tasks as well as humans: Speed and Range of motion.

5.2.1.1 Speed limitation

A robotic arm mostly manipulates objects at its end-effector, which generates large loads on the arm due to the moment of inertia. Therefore, tasks such as knife chopping (Figure 5.4A1), which require a fast periodic motion at the end-effector may exceed the robotic arm’s capability.

Objects manipulated as a whole		Objects manipulated by constituent parts		
Linear	Rotational	Squeezing & Releasing	Twisting	Pumping
A1  Knife	B1  Screwdriver	C1  spray bottle	D1  Door knob	E1  Hand sanitizer
A2  Spice can	B2  Whisk	C2  Wire cutter	D2  Pepper grinder	E2  Pump


 Movement of the object when manipulated by human

Figure 5.4: Everyday handheld objects are often not manipulable by a generic robotic arm with a common parallel gripper: when manipulating an object as a whole, speed and range of motion are two main limitations and; when manipulating an object by its constituent parts, the dexterity required to both grasp and perform a range of manipulation (squeezing/releasing, twisting and pumping) is the main challenge.

5.2.1.2 Range of motion limitation

Using a conventional robotic gripper, some object manipulation requires a certain type of grasp, *e.g.*, rotating a screwdriver (Figure 5.3a). However, when the object is further away, even though it is still within reach of the robotic arm, the grasp is different and as such, it is no longer able to afford the same manipulation (Figure 5.3b). In other words, grasping the tool at a certain angle might render the manipulation impossible because reaching that angle would already constrain the robotic arm's joint rotation, thus limiting how it can perform subsequent manipulation (Figure 5.3b). In some edge cases, the manipulation might be interfered with by the physical surroundings, *e.g.*, the ground (Figure 5.3c). In robotic terms, under such circumstances, the robotic arm is said to be outside of its *dexterous space* [122] when having to perform the manipulation at certain angles.

5.2.2 Objects Manipulated by Constituent Parts (Dynamic Objects)

In contrast to objects manipulated as a whole, there exist objects made up of and manipulated by their movable parts, as shown in Figure 5.4C-E. Manipulating such objects tends to be more difficult for a robotic arm, which needs to first grasp the object stably and then perform a dexterous manipulation like human hand(s), *e.g.*, grasping a pepper grinder and then twisting the grinding cap (Figure 5.4D2). We summarize the following three types of manipulation that makes objects not manipulable by generic robotic arms, each of which can be either one- or bi-directional.

5.2.2.1 Squeezing

In order to manipulate objects that require ‘squeezing’ (*e.g.*, a wire cutter in Figure 5.4C2), a robotic arm (*e.g.*, with a common parallel gripper) would have to pick two points on the squeezing handles for a firm grasp, and then apply a steady force to squeeze the object. However, as the best grasping points for performing the squeezing manipulation (*e.g.*, near the tip of the handle) are usually the furthest away from the center of gravity, this makes the grasp unstable and slippery.

5.2.2.2 Twisting

The twisting force applied to objects *e.g.*, a door knob (Figure 5.4D1) would produce a rotational torque on the robotic arm itself after securing the grasp, which may cause the whole system to become unstable. This is also the most common failure in the *DARPA Robotic Challenge* [105].

5.2.2.3 Pumping

Different from the above, after grasping an object, the robotic arm needs an additional contact surface to perform the pumping task (*e.g.*, to press the hand sanitizer in Figure 5.4E1), which makes it nearly impossible to perform by a common gripper with a parallel design.

Finally, note that some objects (*e.g.*, pepper grinder, pump) would require bi-manual manipulation. In other words, a *single* robotic arm, even as dexterous as a human hand, would find it challenging to manipulate these objects.

5.3 Related work

5.3.1 Augmenting Generic Robotic Interfaces and Everyday Objects

Roman can be thought of as enabling two types of augmentation: (*i*) augmenting a generic robotic arm so that it can attach to and manipulate various handheld objects and (*ii*) augmenting everyday objects so that they become manipulable by the robotic arm.

5.3.1.1 Augmenting generic physical interfaces

Existing work has explored various designs for extending the functionality of generic interfaces. For example, HERMITS extended the capability of self-propelled tangible user interfaces (TUIs) by designing mechanical shell add-ons for TUIs to dock and drive dynamic interactions [152]. Katakura *et al.* introduced a novel way of augmenting a 3D printer head into a robotic manipulator with the mechanical attachments printed by the printer itself [90, 91]. Other researchers augmented a pin-based display by the conversion of the mechanical motion of pins with passive mechanisms to enrich the pins' dynamic interaction [153, 189]. HapLinkage simulated the motion and haptic feedback of virtual hand tools using linkage mechanisms and generated a design space of handheld tools based on motion types and force profiles [120]. Davidoff *et al.* designed mechanical actuators based on the Lego MindStorm toolkit to operate on switches that were intended for humans only [39]. In comparison, Roman's mechanisms augment the physical interface of a robotic arm, *i.e.*, the end-effector, with a versatile magnetic gripper with a focus on assisting the robotic arm to manipulate different everyday objects that would otherwise be difficult to manipulate.

5.3.1.2 Augmenting everyday objects

Another focus of previous research is extending the capability of everyday objects. Reprise makes handheld objects easy-to-manipulate by people with disabilities by generating adaptations attached to the object [28]. Medley enhanced the material properties of 3D printed objects by embedding different reusable materials into the model [26]. Romeo extended the default functionality of everyday objects by embedding a transformable robotic arm into the 3D model of the object [116]. RetroFab augmented the interactivity of physical interface by adding an enclosure consisting of mechanical and electrical components that could automate physical controls [174], which was later extended by Robiot in automating dynamic everyday objects, *e.g.*, adjusting a lamp’s joint angle [118].

RetroFab and Robiot employed *active* mechanism (*i.e.*, with motors onboard), which inevitably makes the augmented object bulky and expensive to scale to the numerous everyday objects in the environment. In contrast, Roman utilized a passively actuatable mechanism attached to an object with only mechanical components to be manipulated by a generic robotic arm.

Since Roman proposes a different complementary approach for robotic arms to grasp and manipulate everyday objects, below we will review related work in robotics research, focusing on gripper design and manipulation.

5.3.2 Relationship to Robotic Research

5.3.2.1 Gripper design

In Robotics, grippers are the most common type of end-effector and an important medium for robots to interact with the real world. The design of robotic grippers has been extensively studied in academia and industry by researchers and practitioners to explore designs of different types of robotic end effectors, *e.g.*, grippers for pick and place, tight grasping, and more. There are two major strategies in designing a gripper: for general purposes or for a specific task. Researchers have explored different types of general purpose robotic

grippers including linkage-based parallel mechanism robotic grippers [71, 99] and compliant underactuated robotic grippers [112, 34, 125, 150]. Meanwhile, exploring how to configure task-specific grippers is an emergent topic that has gained recent attention. Feix *et al.* provided a taxonomy to categorize the potential tasks and the corresponding design features of the robotic gripper [56]. Researchers have designed grippers for different shapes of target objects [154], *e.g.*, for picking objects lying on flat surfaces [9, 11], for assembly tasks [218], or for operating a heavy load [200].

Furthermore, the design of robotic grippers should not only consider the geometry of the objects, but also the interaction with the environment and the kinetostatic properties of the grippers [10]. Different factors are considered in previous research such as dynamic loads [156] and active surfaces after grasping [159, 65]. These are closely related to the tasks conducted by robotic grippers in daily life interacting with everyday tools, which are summarized by the robotic grasping and manipulation challenge [54]. Roman uniquely combines both strategies by implementing a general purpose versatile gripper that could grasp different objects and object-specific mechanisms that could be driven by the gripper to achieve a dynamic manipulation of everyday handheld objects.

Another important gripper-related topic is to create ‘tool changers’ that adapt to the different shapes and sizes of everyday objects. There are different types of tool changers for a robotic arm: automatic tool changers that are electro-mechanically actuated [67] or passive mechanisms actuated by a host robot [16, 169]. Similar to our method, researchers also proposed the design of mechanical tool for robots to grasp different daily objects with modular two-finger gripper but only focused on the grasping of objects with different sizes [81]. Adding to that, our method could solve a more complex manipulation problem of manipulating tools with movable parts (Figure 5.4), which is almost unattainable using a two-finger type end effector. Furthermore, since Roman provides customized mechanisms for different objects that can be driven by the same custom gripper, our method provides a less complex method to program the motion of the arm than solutions that incorporate tool changers.

5.3.2.2 Manipulation

Besides innovating the gripper design, researchers have also investigated methods of control and task planning that program a given robotic arm to perform tasks in human environment. Some research focused on developing a control strategy based on perception such as analyzing the geometry of the objects to obtain the best grasping point [213, 136] or imitating human operation to open medicine bottles [51, 126]. Some developed new algorithms and system designs for specific contexts such as grasping flat objects [185]. At the application level, robots can assist people with daily living tasks which range from fetching a mug [57] to taking medications [101]. In contrast to the above approaches of enhancing perception and control, Roman aims for a different and complementary goal of achieving manipulation by augmenting everyday objects to be more manipulable by a generic robotic arm.

Different from traditional robotic research, a concept of *dexterous manipulation* was first defined by Okamura *et al.* in which a robotic gripper moves objects from one configuration to another [162], *e.g.*, adjusting the angle of a phone in the hand. Such a concept is still being actively studied by researchers in scenarios such as robotic in-hand manipulation [206, 4, 233]. However, such manipulation requires precise control of the forces and motions of fingered or specialized robotic hands and therefore cannot be accomplished by conventional robotic grippers. Roman, as a complementary solution to dexterous manipulation, enables a conventional robotic arm to connect with a versatile magnetic gripper to grasp and manipulate everyday objects with dexterity enabled by object-specific mechanisms.

5.4 Examples: Roman Makes Handheld Objects Robotically Manipulable

We showcase a series of examples where Roman mechanisms attached to handheld objects make them more manipulable by a consumer-grade 6-DoF robotic arm (equipped with a Roman gripper). There are two major types of application scenarios where the objects need to be robotically manipulable: (i) A human collaborates with a Roman-equipped robotic

arm, *e.g.*, the human would hand an object over to the robotic arm where it would then be picked up and manipulated, in tasks that the robot is expected to manipulate a lot of different handheld objects, *e.g.*, making a scrambled egg (Figure 5.6de, Figure 5.14, etc) and (ii) a Roman-equipped robotic arm takes the place of human in performing repetitive tasks by manipulating a hard-to-manipulate object, *e.g.*, a spray bottle. We focus on demonstrating the wide capabilities of Roman in enabling object manipulation, while discussing the technical details of the gripper design and the mechanism generation method later in subsequent sections.

5.4.1 Manipulating Objects as a Whole

5.4.1.1 Increasing speed

As mentioned above, some objects may require a high speed of manipulation that exceeds the motor's capability at the end-effector. Figure 5.5 and Figure 5.6 showcase several different examples where Roman enables a high speed of manipulation: by attaching a pin-in-slot mechanism, the Roman gripper can drive a knife to perform repetitive high-speed vertical motion, *e.g.*, for chopping scallions (Figure 5.5a-b), or to mimic a human's tapping motion on the spice bottle to spread enough white pepper (Figure 5.5c-d). With a spur-gear mechanism, the Roman gripper enables a high speed of rotation of the whisk to mix an egg (Figure 5.6d-e).

5.4.1.2 Expanding range of motion

Roman helps expand the range of motion while manipulating specific objects. For example, grasping a screwdriver at certain distances/angles may prevent a conventional robotic arm from performing the rotating manipulation (Figure 5.3). Therefore, Roman adopts a bevel gear mechanism that could change the rotational axis of the input, expanding the range of motion of the screwdriver's manipulation (Figure 5.6a-c). Similarly, by attaching the whisk at a small angle (as opposed to being perpendicular) when fastened to the output spur gear,

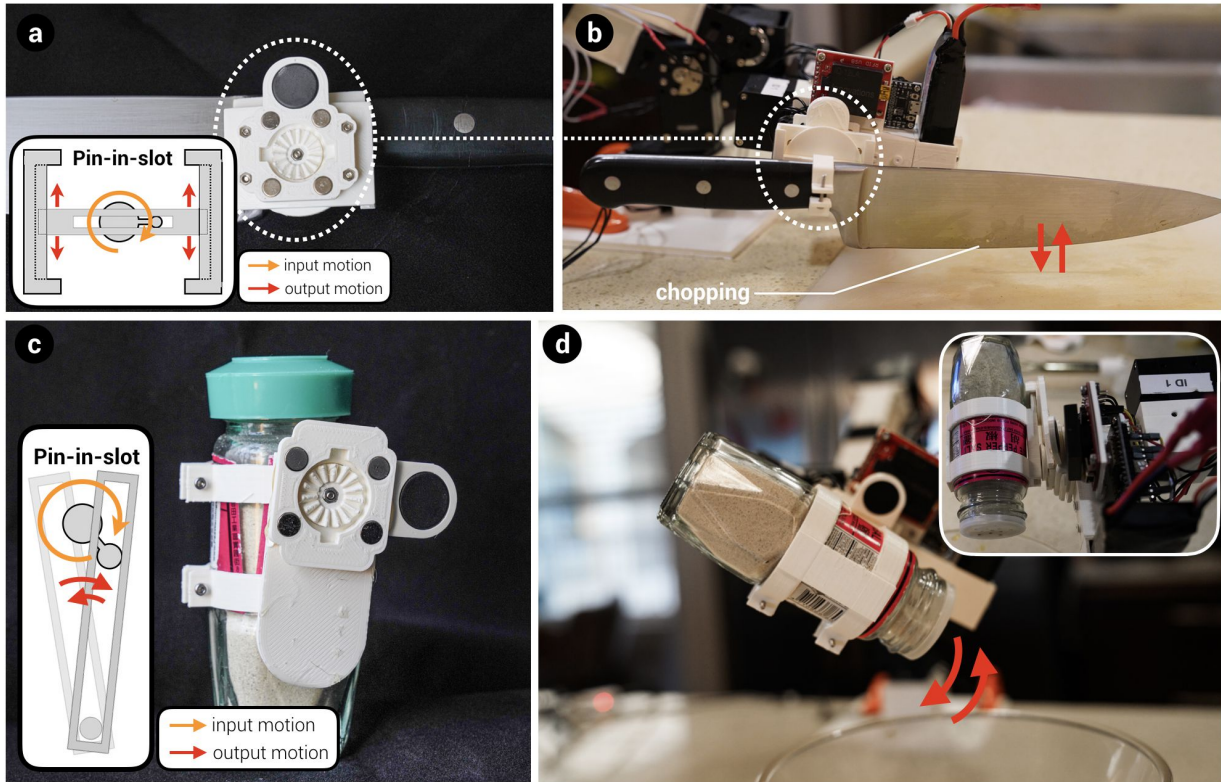


Figure 5.5: Configuration of the pin-in-slot mechanism (a) to produce periodic up and down motion with a knife (b) and an alternative configuration of the pin-in-slot mechanism (c) to produce periodic side to side motion for spreading peppers from a spice bottle (d).

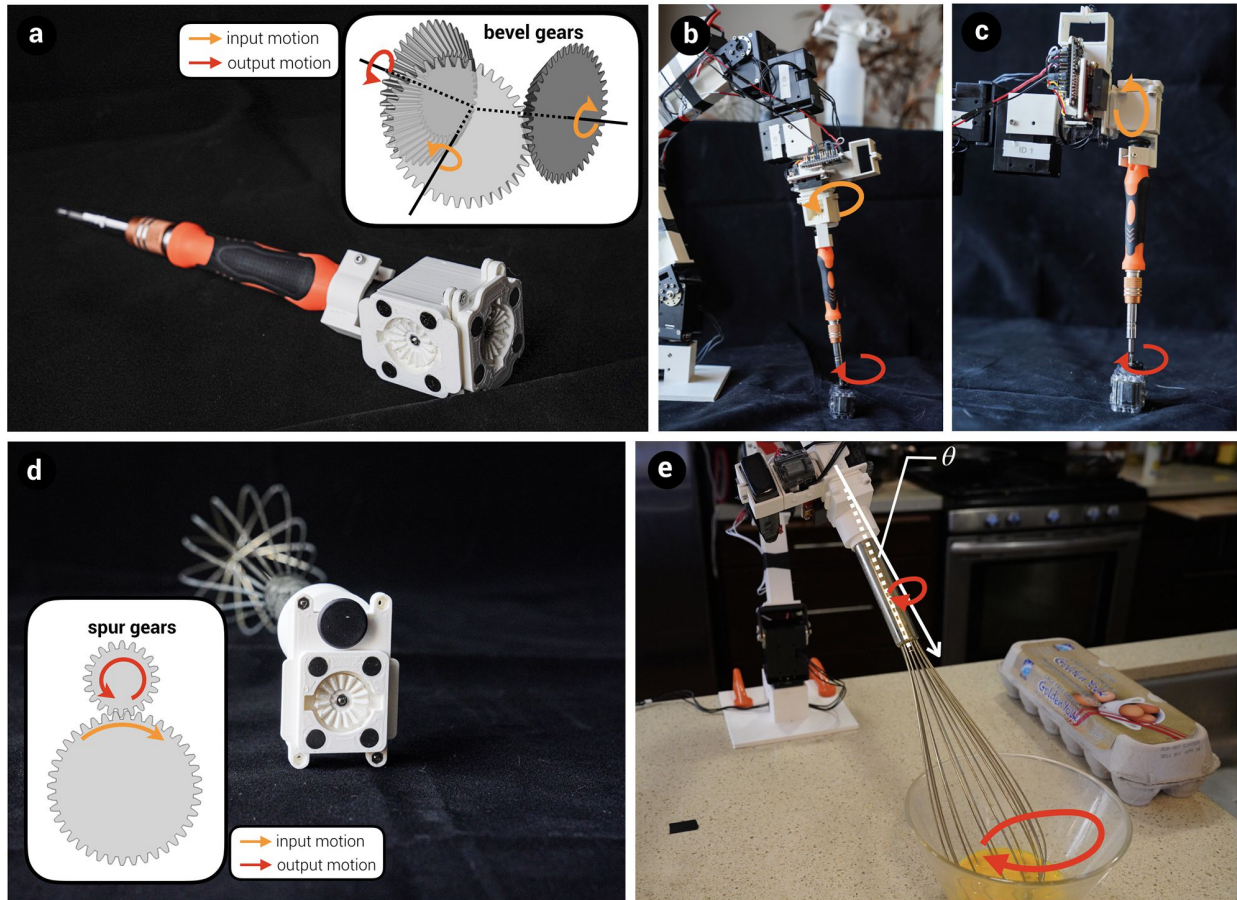


Figure 5.6: Configuration of the bevel gear mechanism to produce rotational motion at an angle on a screwdriver (a), allowing for two attachment configurations (b, c); the spur gear mechanism with a reverse reduction gear ratio produced higher rotation speeds for a whisk (d) and tilting it by a small angle further expands the range of motion at its end (e).

we can expand the range of rotational motion at its end (Figure 5.6de).

5.4.2 Manipulating Objects by Constituent Parts

5.4.2.1 Enabling squeeze & release manipulation

As shown in Figure 5.1, the wire cutter augmented with a gear-rack mechanism can perform the task of cutting a wire by squeezing the handles (Figure 5.1h). Similarly, a gear-rack is used for the spray bottle and thus a robotic arm could fetch the bottle and water the flower

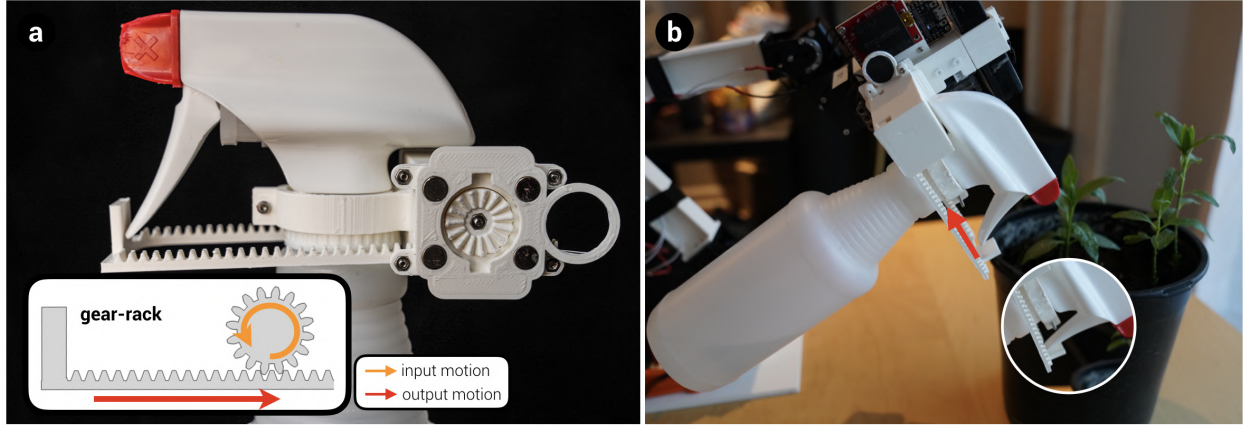


Figure 5.7: Configuration of the gear rack mechanism to produce linear motion (a) in a single direction to squeeze the handle of a spray bottle (b).

automatically (Figure 5.7). As the spray bottle only requires one-directional manipulation, the mechanism only needs to squeeze in one direction before releasing the handle to return to its original position.

The chopsticks require a bi-directional squeezing and releasing manipulation to pick and place food and a spur gear mechanism is used for the manipulation (Figure 5.8). A gear-rack mechanism is utilized for the can opener to perform the squeezing and rotating manipulation to pierce the peanut butter can (Figure 5.12d-g). Specifically, a ratchet design (Figure 5.12e-f) is used to ‘lock’ the squeezing of the handles (Figure 5.12h), which then enables the gripper to detach from the handles, attach to the cutting part (Figure 5.12c), and drive the rotation of the handle to open the can (Figure 5.12h).

5.4.2.2 Enabling twist manipulation

Following the manipulation of piercing the can, the robotic arm is able to twist the handle continuously to open the can by using a spur gear mechanism (Figure 5.12ch). Since both the squeezing and the twisting manipulation of the can opener require a relatively large strength to manipulate, a gear box with ratio of 9:1 is adopted to increase the output torque applied to the target object (Figure 5.19d). Using a pepper grinder is important in a series

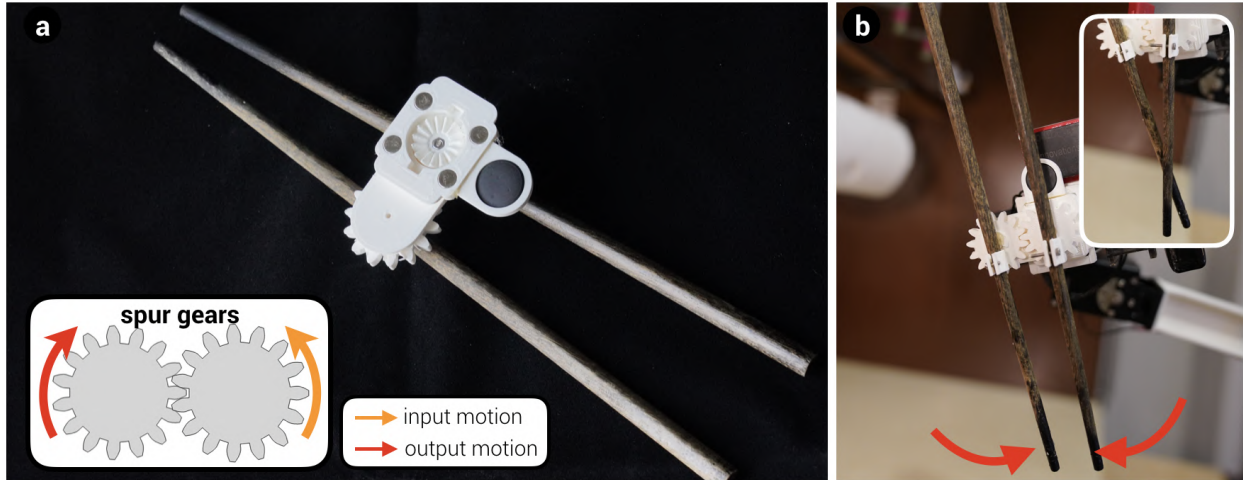


Figure 5.8: Configuration of the spur gear mechanism to produce rotational motion to squeeze together the tips of chopsticks (a), and a practical demonstration of the chopsticks being manipulated with the mechanism attached (b).

of cooking tasks (*e.g.*, making an omelette). With a bevel gear mechanism, the robotic arm can twist the grinder repetitively to sprinkle pepper on the eggs (Figure 5.10c-d). A robotic arm can also collaborate with a human in a cooking task by opening the lid of a starch jar with a bevel gear mechanism (Figure 5.10a-b). Roman also enables a robotic arm to open the door by twisting the door knob with a spur gear mechanism on it (Figure 5.9).

5.4.2.3 Enabling pump manipulation

With a gear-rack mechanism, an oil spray can be manipulated by a robotic arm to help human cook (Figure 5.14). A robotic arm can fetch a bottle of hand sanitizer augmented with a gear-rack mechanism and pump it when the user approaches it (Figure 5.11). To enable a repetitive manipulation of the balloon pump, a gear-rack mechanism is also adopted to perform a bi-directional manipulation (Figure 5.13).

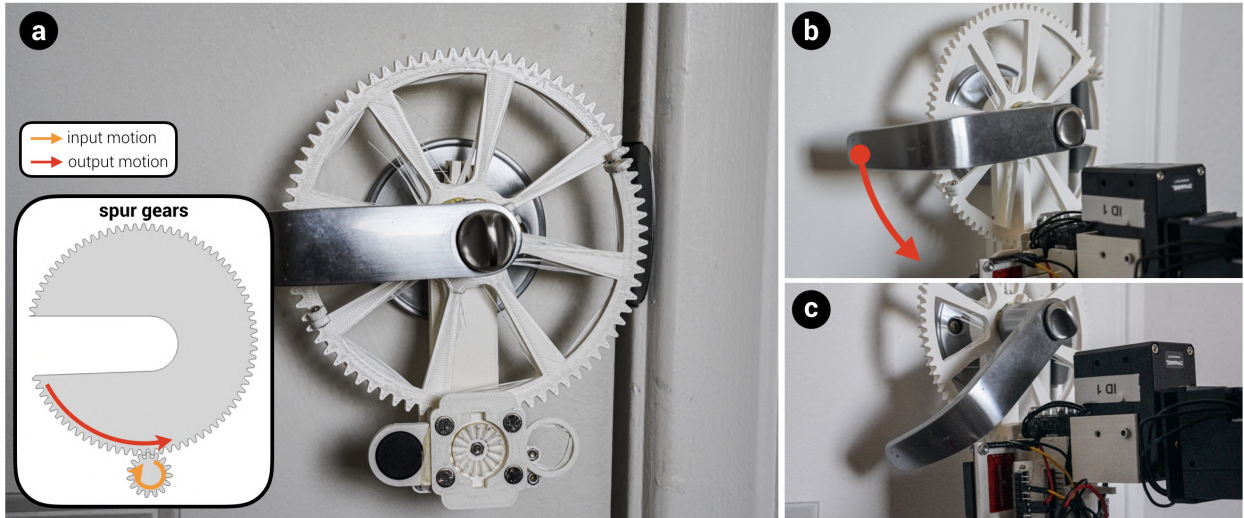


Figure 5.9: Configuration of the spur gear mechanism with a reduction gear ratio to produce rotational motion with high torque in order to rotate the door knob (a), and a practical demonstration of the door knob being twisted with the mechanism attached (b, c).

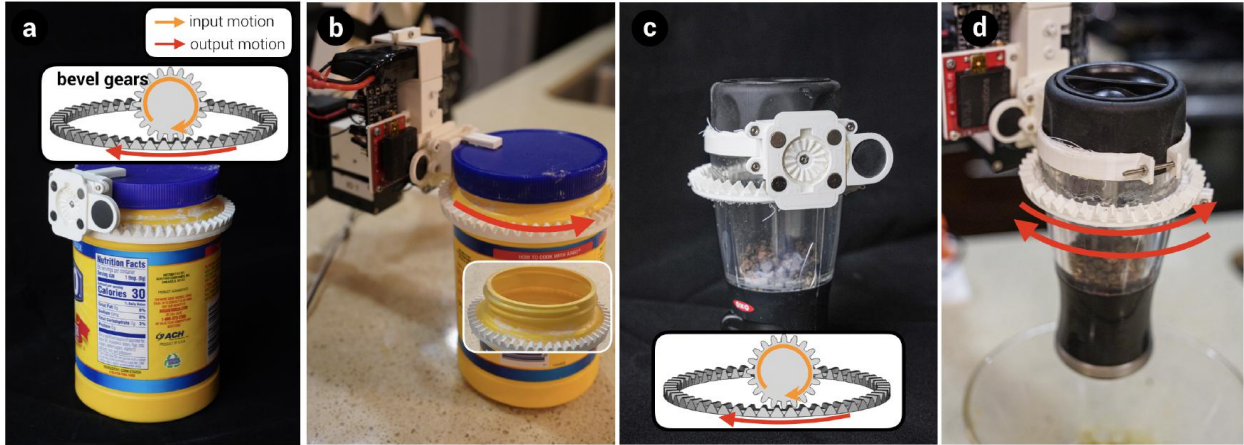


Figure 5.10: Two examples of the bevel gear mechanism to produce rotational motion at an angle to unscrew the lid (a, b) or to rotate the pepper grinder (c, d).

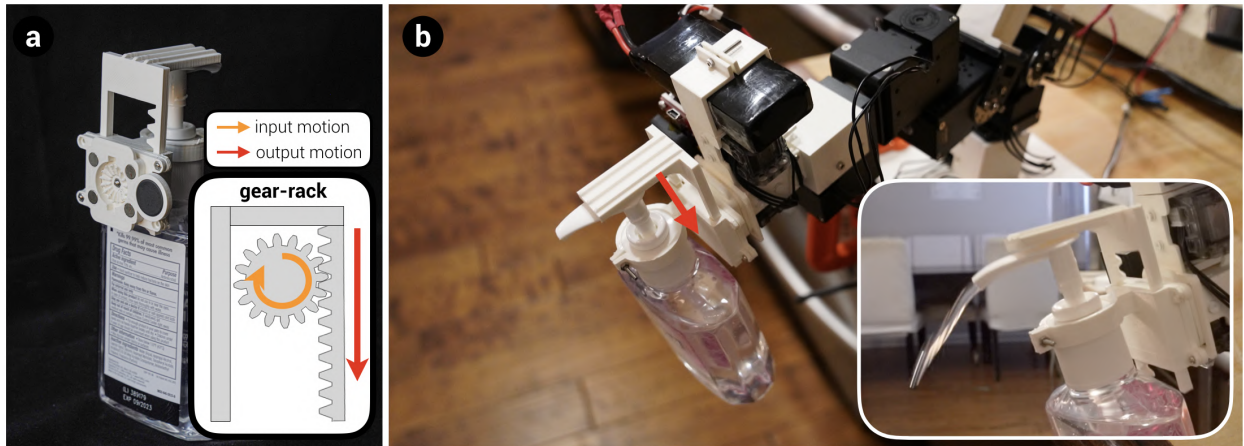


Figure 5.11: Configuration of the gear rack mechanism to produce bi-directional linear motion in order to squeeze a bottle of hand sanitizer (a), and a practical demonstration of the bottle of hand sanitizer being squeezed with the mechanism attached (b, c).

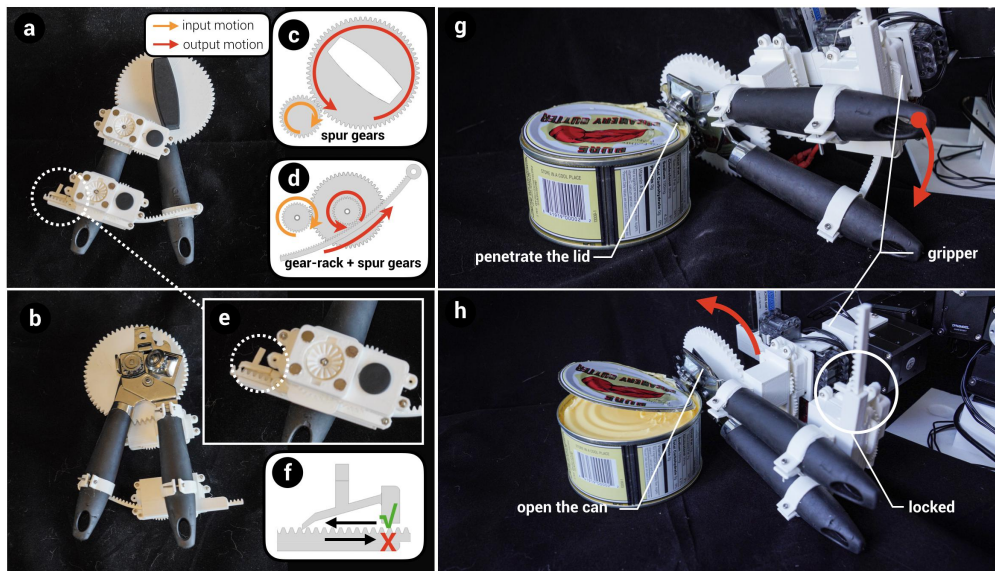


Figure 5.12: Combination of two separate mechanisms with a reduction gear ratio to produce high torque: a spur gear to cut around the can (c), and a gear rack to pierce the can (also using spur gears to increase torque) (d). A ratchet mechanism (f) is used to maintain the position of the piercing mechanism (*i.e.*, keep the handles squeezed), which allows the gripper to switch to the other mechanism (c) for cutting the can open.

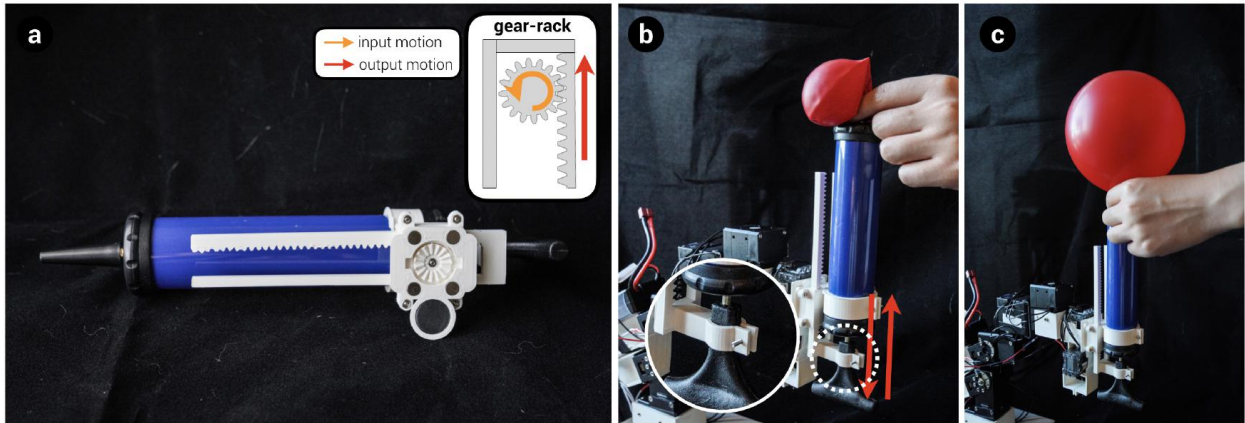


Figure 5.13: Configuration of the gear rack mechanism to produce bi-directional linear motion in order to actuate a pump (a), and a practical demonstration of the pump being actuated with the mechanism attached (b, c).

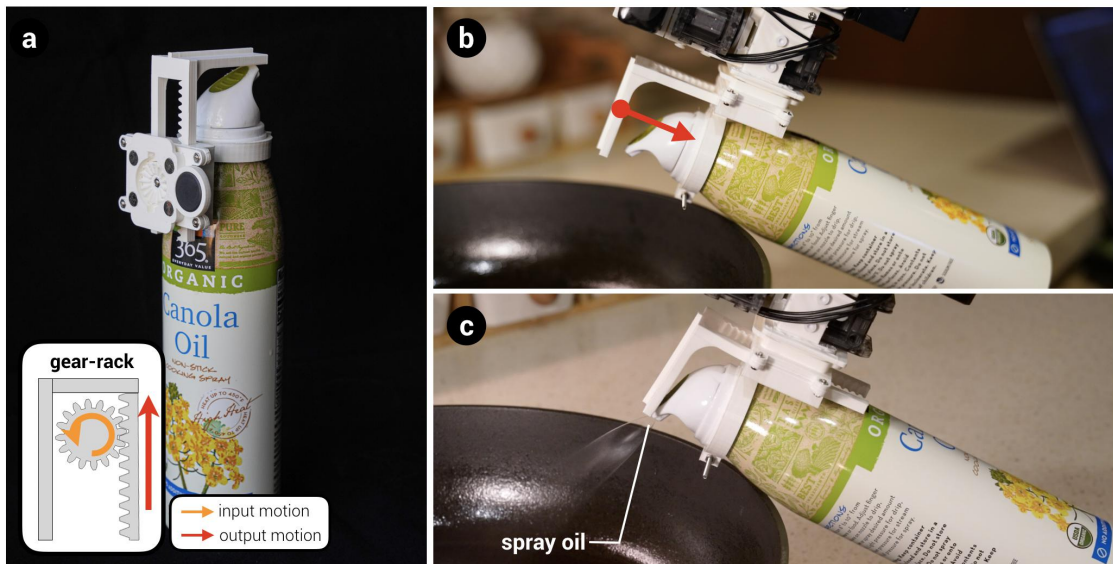


Figure 5.14: Configuration of the gear rack mechanism to produce linear motion in order to depress the spray button (a), and a practical demonstration of the button being squeezed with the mechanism attached (b, c).

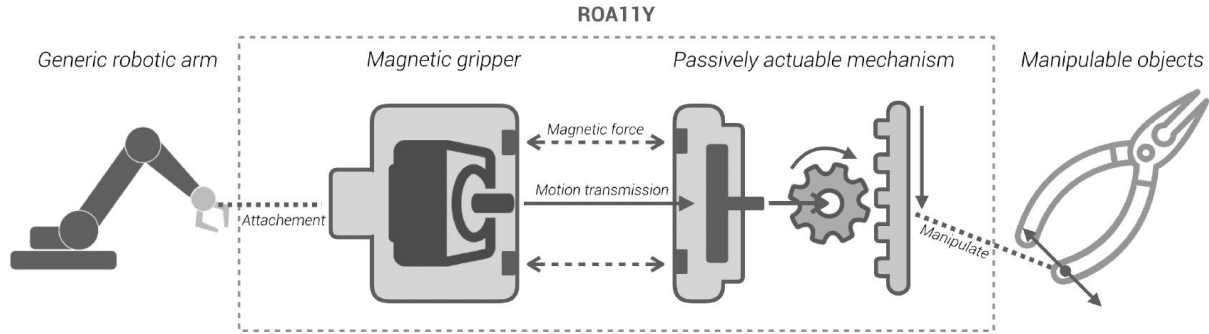


Figure 5.15: Overall hardware structure of Roman

5.5 System Overview of Roman

Roman is an all-in-one solution to make everyday objects manipulable by generic robotic arms and includes both hardware and software support:

- **Hardware modules** consist of (as shown in Figure 5.15)
 - §5.6.1: A modular magnetic gripper that can attach to or detach from an object’s add-on mechanism, recognize the object to retrieve the corresponding control program, and transfer the driving force from the robotic arm’s motor to the mechanism to execute the object-specific manipulation;
 - §5.6.2: 3D-printable powerless mechanisms (spur gear, bevel gear, gear-rack, and pin-in-slot) attached to the object which enables objects to be manipulated as a whole or by their constituent parts. The mechanisms are easy to remove/assemble using screws.
- §5.7: **Software module** is a tool for robotic engineers to interactively specify custom motion profiles for manipulating a specific object (*e.g.*, amplitudes of a signal over time to be sent to the motor that drives the mechanisms to squeeze a wire cutter). We specifically focus on authoring motion profiles, which, to the best of our knowledge, is unsupported by prior work; meanwhile, the task of generating the 3D models of mechanisms—based on the type of motion and an object’s geometry—can be supported

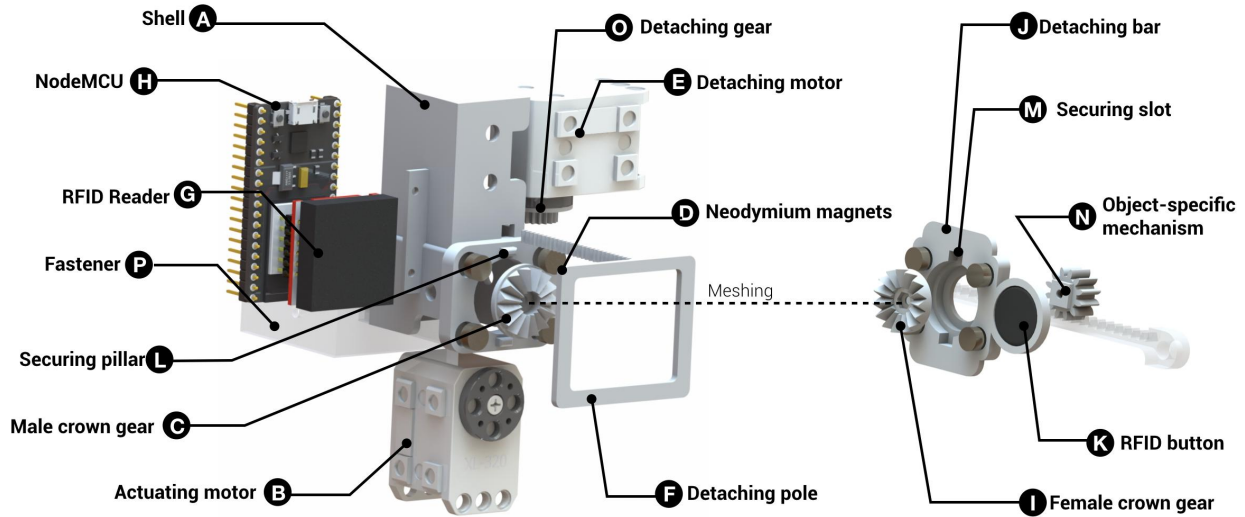


Figure 5.16: Exploded view of the Roman gripper.

by existing tools [116, 118, 28, 25].

5.6 Hardware Implementation

5.6.1 A Magnetic Gripper to Attach to, Recognize, and Transfer Motion to an Object's Add-on Mechanism

The magnetic gripper serves as the intermediary between the robotic arm with two main functionalities: 1) attaching to and detaching from the passively actuatable mechanism on the target objects and 2) driving the mechanisms on the target objects to perform the manipulation. Further, the gripper also contains an RFID reader for recognizing which object it is attached to and running the corresponding control program.

5.6.1.1 Attaching and detaching mechanisms:

The gripper uses four neodymium magnets (Figure 5.16D) to generate the magnetic force for attaching to the mechanism on the target objects. The four neodymium magnets could generate a pull force (the vertical force required to pull the magnets from a steel surface) of

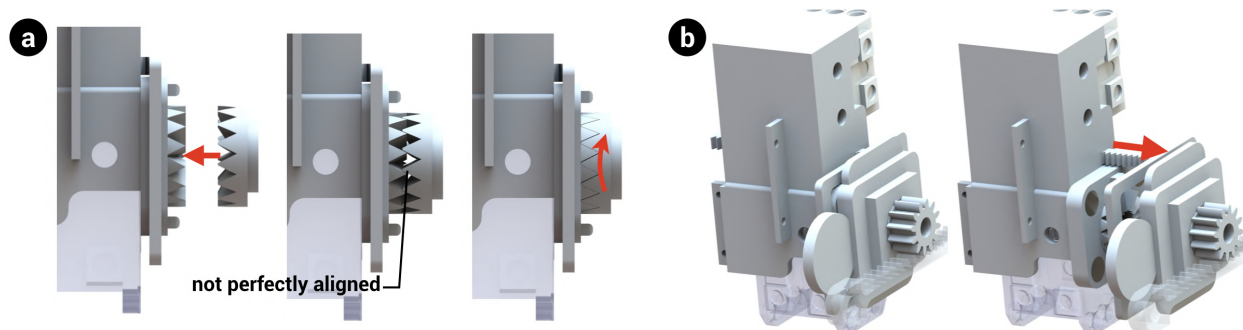


Figure 5.17: Meshing (a) and detaching (b) operations of Roman gripper.

11.2lbs in total, which is sufficient to securely attach to common everyday handheld objects. Furthermore, the strong magnetic force enables the gripper to attach to objects when it is within approximately 1cm of them, which increases the fault tolerance of the robotic arm's manipulation (*e.g.*, a low cost robotic arm³ may have accuracy larger than 5mm). To further strengthen the connection, we designed a pin structure (Figure 5.16L) to counter the lateral force generated during the actuation of the mechanisms.

On the other hand, a stronger connection means larger force required for the detachment. To provide auto-detachment of the mechanisms, Roman employs a Dynamixel XL-320 motor⁴ (Figure 5.16E) and a gear-rack mechanism (Figure 5.16F/O), which could transfer the rotational motion of the motor into linear motion, for the auto-detachment (Figure 5.17b). While the XL-320 motor can generate a maximal torque of 0.39 N·M, we designed the gear of the gear-rack mechanism to have a radius of 5mm which enables it to generate 17.52lbs of force on the rack for detachment (Figure 5.16O).

5.6.1.2 Motion transmission:

The gripper also serves as the actuator of the mechanisms on target objects. Same as the detachment, Roman employs an XL-320 motor (Figure 5.16B) and a pair of crown gears (Figure 5.16C/I). The crown gear could transmit the rotational motion of the motor to the

³PincherX 150 Robotic Arm: <https://www.trossenrobotics.com/pincherx-150-robot-arm.aspx>

⁴<https://www.robotis.us/dynamixel-xl-320/>

mechanism with auto-alignment of the teeth (Figure 5.17a). On the side of the mechanism, modular driving gears (*e.g.*, gears or gear-rack mechanism, Figure 5.16N) can be assembled with the female crown gear to actuate different types of mechanisms.

5.6.1.3 Communicating with target object

Roman employs NodeMCU ESP-12 module⁵ for communicating with the web server (Figure 5.16H). An RFID reader is used for recognizing different objects (Figure 5.16G) and each mechanism on the object comes with an RFID tag (Figure 5.16K) whose ID is associated with a user-defined control program to perform a specific task. Fasteners can also be customized and 3D printed to fit different robotic arms using screws (Figure 5.16P). Finally, the gripper can be powered by an additional 7.4v LiPo battery. The whole system weighs 110g without the battery, which makes it possible to be installed and used on any generic robotic arm.

5.6.2 Mechanisms for Manipulating Handheld Objects with Different Motion Profiles

We first discuss the mechanism design on the objects' side that transfers the rotary input from the motor into task-specific motion profile for different objects.

Objects require different motion profiles in order to perform their tasks. The motion profile is defined as the required output motion for the objects in order to perform an object-specific task. For example, objects in the *squeezing* category may require a curved motion profile (*e.g.*, the legs of the wire cutter move in an arc trajectory for a cutting task Figure 5.1); objects in the *twisting* category require a rotary motion profile (*e.g.*, rotating the door knob to open the door Figure 5.9) and objects in the *pumping* category require a linear motion profile (*e.g.*, linear pushing motion for the balloon pump Fig Figure 5.13).

To address this, Roman adopts four basic types of mechanical mechanisms: spur gears,

⁵<https://www.nodemcu.com/>

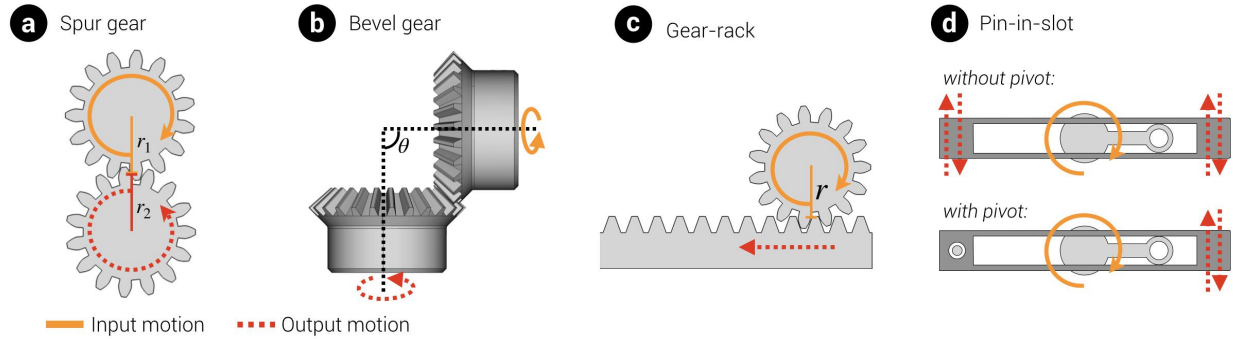


Figure 5.18: Overview of the four types of mechanisms used to achieve target motions: spur gears for rotational motion (a), bevel gears for rotational motion at an angle (b), gear rack for linear motion (c), and a pin-in-slot mechanism to achieve periodic motion (d).

bevel gears, gear-and-rack and pin-in-slot mechanisms (Figure 5.18). As the motor outputs rotary motion, the goal of the selected mechanisms is to transfer the rotary motion into desired motion profiles. Comparing to the mechanisms used in [118], Roman only focuses on the motion output while selecting the mechanisms and uses additional gearbox to tackle the torque requirement.

5.6.2.1 Spur gears

Spur gears are a mechanism in which multiple gears mesh together to transmit the rotary motion from one shaft to another (Figure 5.18a). Therefore this mechanism can transfer the rotary motion from the motor input into a rotary motion profile.

Rotary to rotary While the output motion has the same rotary motion as the input, the gear-pair mechanism can translate the rotary axis to a parallel position, which enables the mechanism to be anchored to a fixed point on the object while manipulating the object. As shown in Figure 5.9, instead of being directly anchored to the door knob, which may get in the way of people using it, the mechanism for driving the knob can be shifted to a position where it does not interfere with regular use of the knob.

Speed and strength Besides the translation of the rotational axis, Roman leverages the property of reduction in gears to generate higher speed or strength than is typically

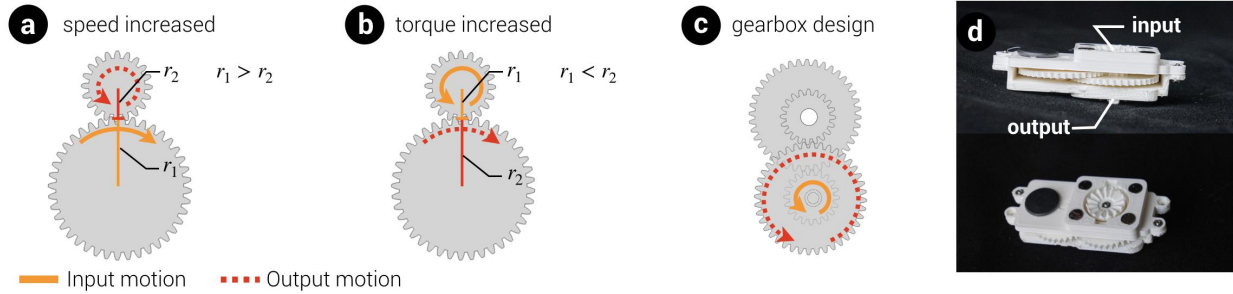


Figure 5.19: Breakdown of the configuration of the gearbox, showing how to produce higher speeds at the expense of torque (a) and to produce higher torque at the expense of speed (b), as well as the physical configuration of the gears (c) and the design of the gearbox (d).

available from a motor. Gear reduction is an arrangement of gears in which an input speed can be raised at the expense of torque, or the output torque can be raised at the expense of speed as explained in the following equations:

$$\text{Speed}_{\text{output}} = \frac{r_{\text{input}}}{r_{\text{output}}} \times \text{Speed}_{\text{input}}$$

$$\text{Torque}_{\text{output}} = \frac{r_{\text{output}}}{r_{\text{input}}} \times \text{Torque}_{\text{input}}$$

With a proper selection of the gears, the mechanism can generate much larger torque than the motor's base ability (Figure 5.19). In the meantime, with a reversed reduction gear, the mechanism can generate higher rotational speed than the motor's typical maximum speed. The whisk is an example of leveraging reversed reduction gears to enable the whisk to rotate at a high speed to beat the egg (Figure 5.6de). Such reduction gears can also be combined with other mechanisms depending on the task (*e.g.*, the can-opener adopts a reduction gear box over its gear-rack mechanism to generate enough force to pierce the can as shown in Figure 5.12d).

5.6.2.2 Bevel gears

Similar to the spur gears, the output of the bevel gears is also rotary motion. Unlike spur gears, bevel gears change the orientation of the rotational axis, which enables robotic arms to execute a twisting task from different angles (Figure 5.18b).

Rotary to rotary Bevel gears can generate a rotary motion profile for target objects. With the property of being able to change the orientation of the rotational axis, bevel gears enable robotic arms to operate twisting tasks with a space-efficient solution. For example, in order to perform a lid opening task, a spur gears design would make the mechanisms bulky as the driving gear will protrude from the jar lid. In contrast, A bevel gears design rotates the protruding gear so that it reduces the overall volume occupied by the entire mechanism (Figure 5.10).

5.6.2.3 Gear-rack

The gear-rack mechanisms are utilized to convert the rotary motion from the gear into the linear motion of the rack (Figure 5.18c). As the gear-rack mechanism generates a linear motion profile, this mechanism can be used by the objects in the *grasp & pump* category. Further, the gear-rack mechanism can also be utilized for generating a curved motion profile, which will be discussed below.

Rotary to linear While the gear-rack mechanism converts the rotary motion from the gear into the linear motion of the rack, it could help the robotic arm to manipulate objects that require pumping (Figure 5.4E). Designers of the mechanism could adjust the output velocity and force by modifying the size of the gear based on the equation:

$$\text{Speed}_{\text{output}} = \text{rpm}_{\text{motor}} \times r$$

$$\text{Force}_{\text{output}} = \frac{\text{torque}_{\text{motor}}}{r}$$

where r represents the radius of the gear attached to the motor. For example, hand sanitizer is an example that requires large torque to squeeze out sanitizer Figure 5.11.

Rotary to curved Besides the linear motion profile, the gear-rack is able to generate a curved motion profile for squeezing objects such as the squeezing leg of can opener (Figure 5.12). Roman leverages the compliant property of the PLA material to design racks that can bend themselves to adapt to the curved motion (Figure 5.12e). As a result, the fastener on the rack (normally attached with the other movable part of the object) would require an

active joint for the rack to rotate relatively, without which the curved motion may generate a large offset at the tip of the rack and break the mechanism.

Bi-directional manipulation The gear-rack mechanism may deal with objects requiring either one or bi-directional manipulation. For objects with only one-directional manipulation such as the oil spray (Figure 5.14), a simple bar is sufficient for generating a one-directional force. However, objects that require bi-directional manipulation such as the balloon pump, require an additional structure fastened to the part to enable motion in two directions (Figure 5.13b).

Single-direction constraints Roman also provides a single direction constraint using the gear-rack mechanism. For example, in order to perform the task of piercing the can and then opening it, the mechanism needs to lock the squeezing mechanism for the opener to continue to pierce through the surface of the can. To achieve this, we employ a design inspired by the ratchet gear, where the rack can move freely in one direction while being prevented from moving in the opposite direction unless a pin is depressed, which releases the mechanism (Figure 5.12d).

5.6.2.4 Pin-in-slot

A pin-in-slot is a mechanism where a pin-joint moves along or parallel to a slide-joint (Figure 5.18d). The pin-joint receive the rotary input from the motor and transfer the motion to the slide-joint. By fixing one side of the slide-joint, the pin-in-slot mechanism can generate a single-sided motion profile (Figure 5.18d bottom) while a double-sided motion profile can also be generated by placing the slide-joint inside of another slider joint (Figure 5.18d top).

The spice bottle is an example of using the pin-in-slot mechanism to mimic the human shaking action by generating a single-sided motion profile (Figure 5.5cd) and the knife is an example of performing the chopping task by generating a double-sided motion profile (Figure 5.5ab).

Periodic motion Besides generating linear and curved motion profiles, the pin-in-slot

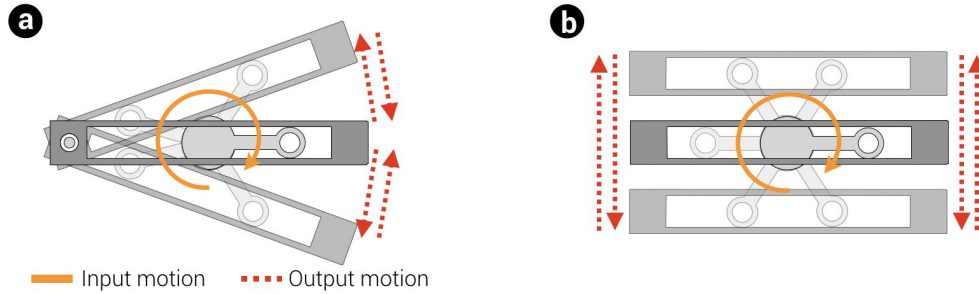


Figure 5.20: Demonstration of the two types of periodic motion that could be accomplished with a pin-in-slot mechanism: angular (a), and up and down (b).

mechanism is good at generating a periodic motion. As shown in Figure 5.20, the continuous rotation of the pin-joint can generate a double-sided or single-sided periodic motion at the slide-joint, *e.g.*, for continuously shaking a spice bottle (Figure 5.5cd). While it is possible for other mechanisms to generate a periodic motion with periodically changed control input, the control signal may experience data loss when changing intensely in short periods. Differently, the periodic motion generated by the pin-in-slot mechanism is easier to control because it relies on the stability of the mechanism components.

5.7 Software Implementation: A Tool for Robotic Engineers to Specify Custom Motion Profiles for Object-Specific Mechanisms

In this section, we illustrate the workflow of a user authoring the control program for a task given some pre-generated mechanisms to manipulate a target object. We assume that the users are robotic engineers with a mechanical or robotic background and the mechanism generation is supported elsewhere by tools similar to [118, 28, 68]. We exclusively focus on the less-supported part where the users author control programs for different object-specific tasks. Roman’s front end is written in JavaScript and the back end is written in Python. Roman communicates with the ESP8266 module on the NodeMCU and stores the motion profile through a Python-based web server. The RFID reader on the gripper reads the ID

of different RFID chips on the objects and accesses the corresponding motion profile via the NodeMCU.

5.7.1 Custom Motion Profiles

As Roman transfers the rotary motion input from the motor to a customized output motion profile, it requires further specification of the motion profiles in order to perform an object-specific task. For example, to design a task for manipulating the wire cutter while collaborating with a human user (Figure 5.1), the wire cutter is expected to *(i)* move to the desired position for the user to hold the wire for cutting, *(ii)* squeeze the handles about half-way and hold the position in order for the user to align and adjust the wire, *(iii)* squeeze fully to cut the wire, and *(iv)* return the handles to the initial configuration. While step *(i)* is conducted by the movement of the robotic arm, the rest are done by Roman hardware and require a custom design of the motion profile. More specifically, the customizability of the motion profiles amounts to specifying the amplitudes u of the control signal over time, which corresponds to the rotational speed of the motor output. The aforementioned task for the wire cutter has a motion profile as the control signal shown in Figure 5.22.

5.7.2 User Interface

As shown in Figure 5.21, to facilitate the design of such custom motion profiles, the Roman software provides a user interface for interactively specifying the custom motion profiles by adding or adjusting the key points in the graph of a control signal (Figure 5.21c). The u for the y-axis is a unitless value ranged from -1 to 1 which represents the rotational speed of the motor relative to the maximum speed under the current load (with negative values corresponding to rotation in the opposite direction). The x-axis represents time and the user has the ability to modify the total time the motor is running at a certain speed by adjusting the length of each period of the control signal. With the combination of different control signals, the user could design a custom motion profile for a object-specific task.

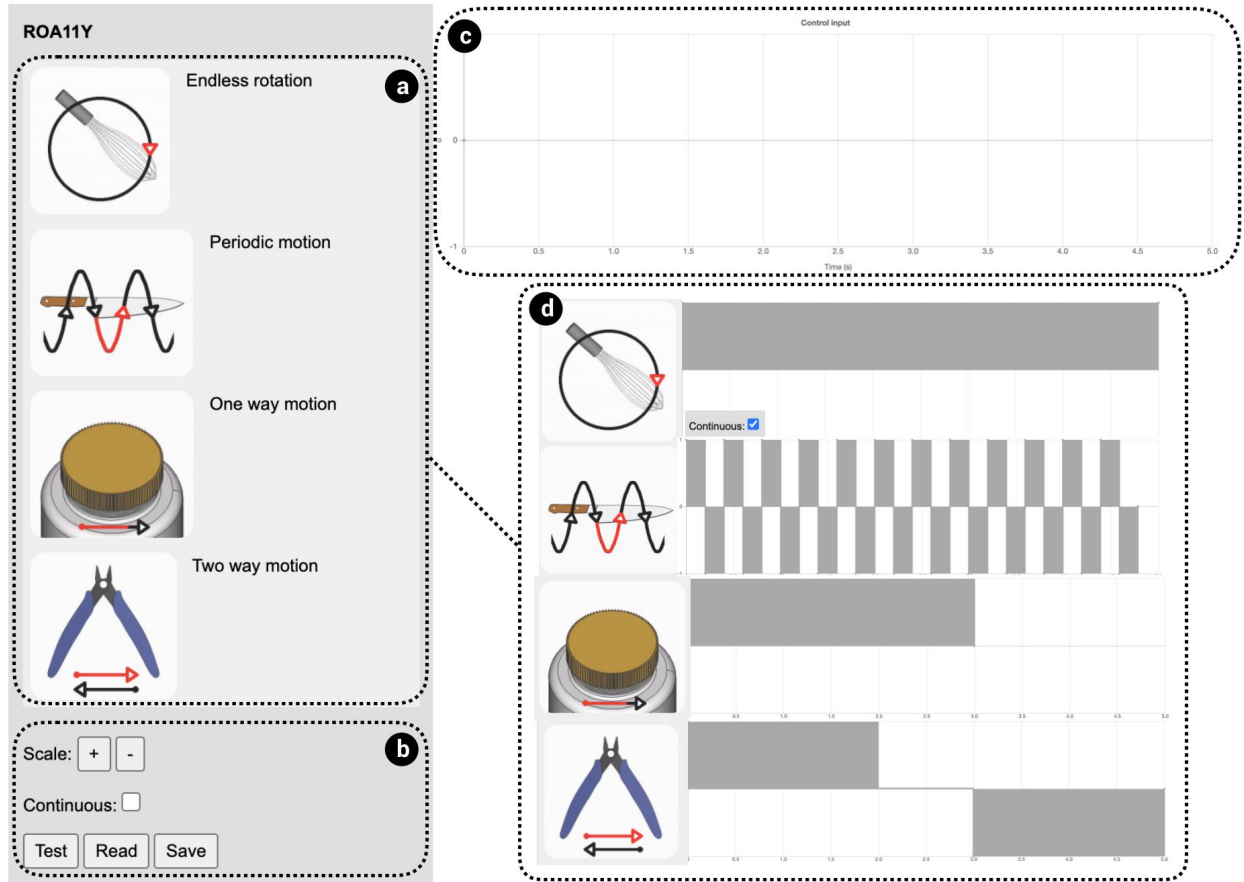


Figure 5.21: The Roman user interface used for selecting motion templates, authoring control programs, and uploading them to the robotic arm.

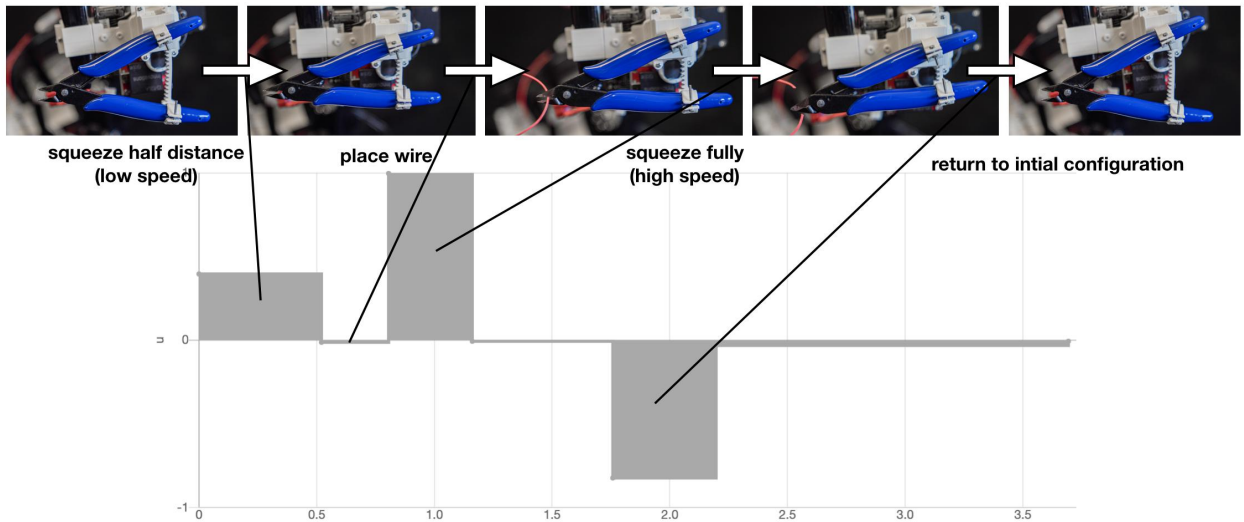


Figure 5.22: A sample motion profile used to control the wire cutter in order to cut a wire.

5.7.2.1 Motion Profile Templates and Interactive Graph Editing

Roman provides different ways to interact with the graph of the control signal. As shown in Figure 5.21a and d, Roman provides four types of motion profile templates for a user to adapt to their needs, including endless rotation (*e.g.*, whisk), periodic motion (*e.g.*, knife), one-way motion (*e.g.*, jar lid), and two-way motion (*e.g.*, wire cutter). By clicking on the corresponding template, the control signal is imported into and visualized on the interactive graph and the user can further adjust the motion profiles.

To start editing the graph, a user simply double-clicks anywhere on the graph and then adjusts the position of an existing point by dragging it. The user can also adjust the range of the x-axis (*i.e.*, increase/decrease time) by clicking on the +/- signs (Figure 5.21b), which either adds or removes 1 second.

5.7.2.2 Real Time Testing

There is uncertainty in manipulating real-world objects, *e.g.*, the manipulation of a wire cutter (the motor's rotational speed) might need to vary with different wires. As such, Roman enables a real-time *testing* mode (Figure 5.21b) that creates a feedback loop in which the user could test and adjust the custom motion profile to see how well the manipulation is performed on the target object. The user could also check the box of 'continuous' to specify a repetitive motion. Once a user has tested and is satisfied with the motion profile, they can click on the 'save' button to store the motion profile to the target object.

5.8 Validation with Expert interview

We conducted an expert interview as an initial step to evaluate Roman. The purpose of this interview is to evaluate the usefulness and practicality of Roman in helping robotic engineers achieve object manipulation tasks, as well as to gather feedback and further suggestions from experts in the field of robotics.

5.8.1 Participants & Procedure

Four participants (all male with an average age of 28.25) with expert knowledge in the domain of robotics and mechanism design were invited for interviews. Three of them are Ph.D. students in the Mechanical Engineering Department with a focus on the research of robotic design and control (P1-P3). Among these, two had prior experience in developing robotic manipulators as a product in a robotic start-up (P1 and P2). The other participant is a post-doc researcher in Electrical Engineering Department with a Ph.D. degree in mechanical engineering and his research focuses on the computational design for origami robots (P4). Although no participants worked in the exact area of robotic manipulation in a human environment, all were familiar with the concept of mechanism design and control of robotic arm manipulation.

Each of the participants was interviewed in-person. First, they were explained the goal and function of Roman hardware and one author showed them demos of using the Roman gripper to manipulate five different objects (wire cutter, hand sanitizer, jar lid, whisk and spice can). These objects covered the four types of mechanisms we used in Roman as well as the four motion profile templates in the user interface. Afterwards, they were introduced to the Roman software and were asked to replicate the custom motion profiles of the wire cutter (which has the most complicated motion profile as shown in Figure 5.22). The motion profiles designed by the participants were test by performing a cutting task (similar to Figure 5.1) of an AWG 18 wire. The participants were able to design the profile and cut the provided wire physically in only 5 minutes (excluding the time needed to become familiar with interacting with the user interface) We then conducted a semi-structured interview, mainly to collect their feedback on the capability and practicality of Roman and potential directions for future research. The entire study lasted on average 25 minutes per participant.

5.8.2 Results and Feedback

All the participants were positive to Roman's features, leaving feedback for its *utility* and *practicality* of Roman.

5.8.2.1 Roman is complementary to current approaches in robotic manipulation in human environment

All the participants considered Roman to be a complement to the current research in the robotic manipulation in situations that require robotic arms in human environment. P2 stated that most of the existing robotic arms have limited functions or they required complicated programming of the control to use different types of tools. He thought that Roman provided a simple solution as the object + mechanism can be pre-designed as a product and delivered to customers. From the perspective of a user, P2 was initially concerned about the practicality of possessing a robotic arm in daily life, which was later answered by himself when he noted that the Roman solution could make a robotic arm adapt to different objects in a task-heavy scenario like cooking. P1 was concerned about the pre-design of the custom motion profiles to be time-consuming, but he also pointed out that in the application scenarios such as cooking, the number of target objects is limited and therefore it is acceptable. P3 considered Roman usable, but he was more concerned about the necessity of the usage of robotic arm in such scenarios versus designing more easy-to-manipulate objects for human (*e.g.*, designing a jar that can be opened by pressing a button instead of adding mechanism to be manipulable by robotic arm). P4 mentioned that it makes more sense to have a strong and functional robotic hand to manipulate different objects without altering them but he agreed that the artificial hand solution is costly (a powerful enough robotic hand normally costs more than \$10k) and Roman could be a low-cost solution.

5.8.2.2 Roman’s mechanism designs and custom motion profiles are replicable by robotic experts

When asked about whether they are able to replicate the hardware design of Roman’s five examples⁶, all the participants agreed and were able to understand the rationale behind the design of each mechanism. P1 thought that it would be easy for a person with fundamental knowledge of mechanical design to replicate all of the examples. P2 also pointed out that with the explanation of different mechanisms as in the previous section (§5.6.1.3) as a guideline, it is a lot easier to select and design the mechanisms for specific objects. P4 stated that it is easy for robotic experts to replicate the examples but it would be great if a parametric design tool is provided for novice users.

All the participants were able to replicate the custom motion profiles for the wire cutter to perform a wire cutting task. All the participants could understand the differences between each motion profile template. P2 also valued the provided templates because, with the templates, he could know the overall movement of the mechanism and would only need to adjust a few parameters.

5.8.2.3 Roman’s hardware could be further improved

The participants suggested that the hardware components can be improved to make the overall structure smaller and more durable including developing new types of mechanism (making the mechanisms more versatile), replacing the neodymium magnets with electromagnets (making the mechanisms smaller) and printing the custom mechanism in carbon (making the mechanisms stronger).

In summary, results of the expert interview indicated that Roman could be a potential solution for robotic manipulation to assist human with daily tasks in certain scenarios. Also, it was considered practical for people with a robotic or mechanical engineering background

⁶Note that participants did not actually replicate those examples, considering the time-consuming process of modeling, printing and assembly.

to design and replicate the objects with mechanisms using Roman. The participants also recommended that a tool for novice users to custom such mechanism for existing objects would be valuable.

5.9 Limitations, TradeOffs, future work

5.9.1 Performance testing

Roman’s main contribution is the idea of adding mechanisms to enhance objects’ manipulability, which complements existing approaches focused on gripper design and manipulation algorithms. As such, we chose to validate Roman by creating various examples to show the idea’s wide applicability and by interviewing four robotic experts to obtain their initial feedbacks and reactions that would set the scene for future work. As next steps, future work that uses Roman on a specific type of objects/tasks (*e.g.*, manipulating a wire cutter) should define metrics for success of each task (*e.g.*, a wire is completely cut in one trial), control pertinent variables (*e.g.*, the thickness of wires), and employ repeated measures to obtain such metrics.

5.9.2 Mechanisms interfering with normal use

As Roman attaches 3D printed mechanism onto the existing object, some of the mechanisms may interfere with the normal use of the objects. For example, the spur gear mechanism on the chopsticks obstruct the way of a human would use them (Figure 5.8) and the bulky mechanism on the can opener also makes it hard for human to manipulate (Figure 5.12). While some of the mechanism can be easily disassembled, *e.g.*, the rack of the hand sanitizer can be easily pulled out (Figure 5.11), future work can focus on making the mechanisms modular and easy to disassemble such as LEGO MindStorm [97]. It is also possible to embed such function during the product design stage that aims at making the overall object + mechanism manipulable by both humans and robotic arms.

5.9.3 Incorporating sensing & perception

Roman focuses on enhancing manipulability by enabling human collaborating with Roman-equipped robotic arm or a Roman-equipped robotic arm to perform tasks independently. However, the latter scenario assumes that the object's position and orientation are known, which is a trade-off of not integrating sensing modules in the current design. Given the plethora of work on sensing and perception (*e.g.*, [188]), future work can add such modules to Roman's hardware components, which are expected to work independently and complementarily to the current set-up.

5.9.4 Generalizability of the design

Currently, Roman only focuses on tasks driven by the motor of the gripper while the robotic arm have to be manually configured. As such, a Roman mechanism only affords manipulation in a limited space and cannot enable large-scale tasks, *e.g.*, holding a spatular to make stir-fry.

Future work may extend Roman to include software support involving the entire robotic arm, *e.g.*, linking the action of the robotic arm with the actuation of the mechanism. With that, the Roman hardware could collaborate with the robotic arm to achieve more complicated tasks *e.g.*, scooping ice cream (enabled by Roman mechanisms) and distributing it into different locations (enabled by the rest of the robotic arm) automatically.

On the control side, Roman uses open-loop control of the mechanisms. This limits Roman to manipulate objects that have dynamic feedback such as tightening screws in different conditions. One of the future directions is adding haptic sensors to the mechanisms and incorporate feedback control while performing the tasks.

5.9.5 Trade-off between torque and the size/complexity of mechanisms

Currently, Roman enables the manipulation of objects by generating motions using a low-cost motor and 3D printed mechanisms. As a result, Roman is limited to supporting objects that

do not require a lot of torque. While Roman provides a partial solution by adding a gearbox in between the gripper and the object to increase the torque (see the can opener example Figure 5.12), there is a trade-off between the size of the mechanism and the maximum torque the gripper could generate. For example, manipulating a hedge trimmer (too large) or a sealed jar (too tight) will require rather bulky mechanisms unrealistic to be attached to the object. Substituting the motors in the current design with stronger models could partially solve the problem. Alternatively, there exist research opportunities to solve this problem with improved mechanisms, *e.g.*, using more durable materials such as metal linkages or cables to increase the generated force.

5.9.6 Manipulating objects with multiple movable parts or multiple consecutive manipulations

Some objects are articulated with multiple movable parts, *e.g.*, multi-functional pliers, Swiss army knife, flexible selfie rods, and Rubik's cubes. Technically, Roman's mechanisms can extend to more parts by enabling one additional motion at a time, yet, practically, too many mechanisms might not be fittingly added to an object and might even interfere with one another. Future work could focus on involving different materials (cable, metal, carbon) to make the mechanism smaller while functional.

Other objects might require consecutive manipulations to perform a task, *e.g.*, a corkscrew requires a twisting motion on the handle to penetrate a wine cork and a subsequent squeezing motion on the two arms, followed by pulling out the same handle. Currently Roman needs to provide separate mechanisms for each motion, which could result in the overall mechanisms too bulky. To address this problem, future work could incorporate interactive trajectory design into the mechanism design of Roman, similar to the approach in [37] that would need only one custom mechanism to accomplish multiple consecutive manipulations.

5.9.7 Opportunities to support novice users

As mentioned by the robotic engineers in the interview, while it is easy for people with knowledge of robotics to replicate examples in Roman or design mechanism for new examples, novice users might struggle to design mechanism and understand which mechanism to use and how to specify the control signals since Roman does not provide 3D modelling support. One possible idea for future work is to instrument sensors on an object similar to the current Roman mechanisms. Then a user can demonstrate how they would manipulate an object, which can be captured by such sensors (*e.g.*, inertial measurement units or reflective markers + external optical tracking) and incorporate the algorithm in Robiot [118] which could generate the mechanism and control program automatically. In this way, Roman would be more usable to novice users.

5.9.8 Areas of hardware improvement

The hardware of Roman can be improved by: *(i)* switching to a half-duplex enabled microcontroller board that could enable the control of the mechanism to be self-adaptive to different loading conditions; *(ii)* replacing the neodymium magnets with electromagnets that can be programmatically controlled for attachment/detachment, thus dispensing with the additional motor currently used for detachment.

CHAPTER 6

Enabling Pose Estimation of Appearance-changing Physical Objects

3D reconstruction is an increasingly popular method to synthesize training data for vision-based object 6D pose estimation. However, these methods are designed for static objects with diffuse colors and do not work well for objects that change their appearance during manipulation, such as deformable objects like plush toys, transparent objects like chemical flasks, reflective objects like metal pitcher, and articulated objects like scissors. To address this limitation, we propose RoCap, a robotic pipeline that emulates human manipulation of target objects while generating data labeled with ground truth pose information. The user first gives the target object to a robotic arm, and the system captures many pictures of the object in various 6D configurations. The system trains a model by using captured images and their ground truth pose information automatically calculated from the joint angles of the robotic arm. We showcase pose estimation for appearance-changing objects by training simple deep-learning models using the collected data and comparing the results with a model trained with synthetic data based on 3D reconstruction via quantitative and qualitative evaluation. The findings underscore the promising capabilities of RoCap.

6.1 Introduction

Accurately predicting the 6D pose (position and orientation) of a physical object is a crucial task for a wide range of applications in the field of robotics and augmented reality (AR) where robots or human users need to interact with these objects in the environment. Vision-

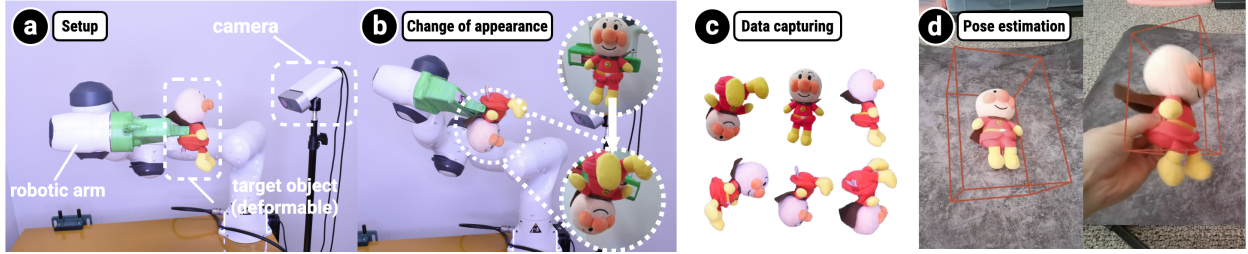


Figure 6.1: The RoCap pipeline is a robotic system designed to collect datasets for the purpose of pose estimation of appearance-changing objects, *e.g.*, a deformable plush toy (a). The system consists of a robotic arm and an RGB camera, which allows for data collection (c) of objects with appearance-changing features (b). Through data augmentation and training on off-the-shelf deep learning models using the collected data, the system can effectively estimate the pose of the plush toy during manipulation, even as it transitions through deformation (d).

based pose estimation has gained popularity in the past few years over tracker or sensor based methods, as it does not require additional hardware, alter the appearance or interfere with the normal use of the objects and it is cost effective and accessible.

However, estimating the 6D pose of an object from an unconstrained RGB image remains highly challenging due to the ambiguity nature of the estimation. To solve this problem, researchers have adopted different approaches including mapping image feature to the 3D model of the object [75] and matching point cloud constructed by depth camera [15]. More recently, data-driven deep learning methods [217, 167] demonstrated accurate predictions of the 6D pose of pre-defined sets of object included in carefully crafted datasets [23, 106]. However, it remains unclear how well they work on objects where carefully labelled data do not exist such as personal objects. To address this issue, some prior work enables end-users to collect datasets for everyday objects 6D pose estimation [137, 172], introducing synthetic approaches to generate a large amount synthetic data given the 3D model of the objects [50], or adopts a few-shot learning method by training on 3D mesh reconstructed from a short clip of video [128].

A limitation of these existing methods is that they mainly focus on objects that are static objects with diffuse colors, with a lesser focus on objects that **change their appearances** when being manipulated, including objects with challenging appearance materials (*e.g.*, transparent and specular objects), deformable objects and articulated objects [204]. a pair of scissors will dramatically change its physical appearance due to mechanical operation and a model trained on the image of a closed pair of scissors might produce lower accuracy at recognizing the same pair in an open configuration. Similarly, a plush toy that changes its shape during manipulation when being affected by gravity will affect the performance of the pose estimation. While one intuitive approach is to capture data while a human user is manipulating the objects, annotating such data at scale would be costly and error-prone.

To address the challenge, we propose RoCap, an automated pipeline to collect image data of appearance-changing object for 6D pose estimation using a robotic arm with minimum human intervention. We deploy a robot arm to mimic human’s hand to manipulate the objects while capturing the image data as shown in Figure 6.1. The 6D poses of the object of each image can be obtained with robotic forward kinematics as each joint of the robotic arm is precisely controlled. Specifically, RoCap performs the data capturing process for eight different appearance-changing objects with deformable, transparent, reflective and articulated properties (Figure 6.3).

We also implemented a simple pose estimation pipeline to quantitatively and qualitatively evaluate the pose estimation performance of the model trained on our collected data comparing against a few-shot learning pose estimation approach based on 3D reconstruction (Gen6D [128]). Both the quantitative and qualitative evaluation results demonstrate that existing work struggles with appearance-changing objects and our approach shows promise in overcoming these limitations with improved pose estimation accuracy.

In summary, our contributions are two-fold:

- **A robotic data collection pipeline** with a 6 DoF robotic arm which captures and annotates 6D pose data for objects that change their appearance during manipulation, addressing limitations in existing data collection methods.

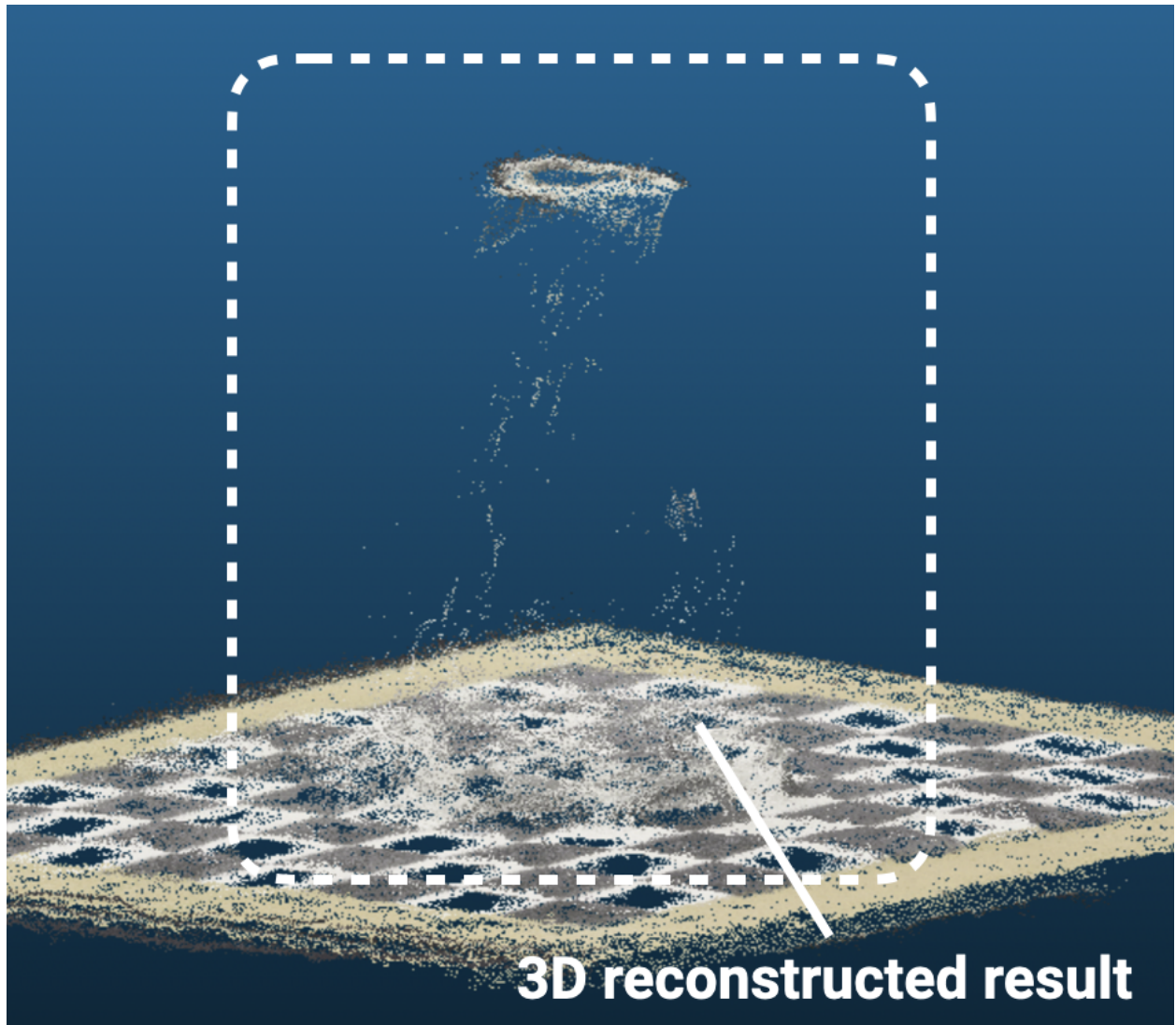


Figure 6.2: 3D reconstructed results for a transparent flask.

- **Quantitative and qualitative evaluations** to demonstrate the feasibility of the pipeline via improved accuracy of appearance-changing objects pose estimation by comparing with an advanced pose estimation method in the field of computer vision.

6.2 Related Work

6.2.1 Object pose estimation

Object pose estimation plays a crucial role in various HCI applications such as augmented reality [75, 15, 74] and robotics and automation [119]. Over recent decades, researchers have explored diverse approaches to predict an object’s pose. These includes sensor applications like IMUs, physical marker techniques such as fiducial markers [88, 207, 62], optic trackers [230] 3D printed embedded QR code [49], computer vision techniques such as color-based tracking, feature point tracking [15] and point cloud alignment [75]. Recent advancement in deep learning has unlocked new challenging tasks such as predicting the poses of hand-object interaction [69, 127, 24], articulated objects [121] and other problem setups [130, 2, 38, 220]. Extending this line of research, RoCap focuses on a new problem setup where the objects will change their appearance during the manipulation. Note that RoCap does not contribute new model architecture or algorithm to improve the performance in the field of deep learning. Instead, RoCap contributes a novel data collection method and the data captured by the system can serve as great resources for researchers in the community of computer vision and machine learning to solve the downstream tracking problems.

6.2.2 Pose data collection

Data-driven deep learning approaches require data annotated with ground truth labels. Yet, annotating 6D pose data is challenging, as it is hard to specify 3D bounding box on a 2D image. To address this, researchers have investigated various methods including three primary strategies: *(i)* training on synthesized data, *(ii)* utilizing publicly available datasets and *(iii)* designing interactive tools for data collection.



Figure 6.3: Example objects for each category that RoCap is focusing on, **Viewing-angle dependent**: (1) flask, (2) water bottle and, (3) pitcher, **Deformable**: (4) flexible frog and (5) stiff anpanman, **Articulated**: (6) scissors, (7) spray head and (8) clamp.

6.2.2.1 Synthetic data

One typical way is synthesizing data with the available resources such as the 3D model of the objects. This approach is commonly used in tasks such as object segmentation [179] and object detection [50]. And a standard way of using synthetic data in pose estimation is to obtain the 3D model and texture of the objects first and then render them with different target background [196]. Although synthetic data can be easily scaled, it comes with the drawback of a disparity between real and virtual data, which might impact model performance. Moreover, as illustrated in Figure 6.2, the necessary step of object reconstruction may fail for our target objects, such as the flask.

6.2.2.2 Real-world data

An intuitive way to bypass the issue of synthesized data is to collect data in the real world. In the recent years, researchers have adopted two major types of data collection methods. The first is “**static object + moving camera**”, where the pose of the object is calculated from the pose of the camera, which can be read from the embedded sensor. Normally it requires a certain level of human labor as first couple frames need to be manually annotated by matching the 3D model to the physical object. For example, several publically available datasets have been collected in this way for benchmarking in the pose estimation domain, such as YCB Video dataset [217], Linemod [76, 20] and T-Less [77]. Additionally, researchers have also developed interactive data collection pipeline to collect data on custom objects (*e.g.*, Label Fusion [137]). However, since the objects remain static, it is challenging to capture the appearance-changing features.

Another approach is “**moving objects + static or moving camera**”. While effective for capturing appearance-changing objects, this approach poses challenging for labeling ground truth. For instance, ARnnotate, used in augmented reality [172], requires users to hold and move the object along a recorded path, leading to potential errors, especially with objects like articulated items or deformed plush toys. RoCap adopts this approach

and ensures the labeled ground truth to be precise by calculating the robotic arm’s forward kinematics while it manipulates the object to capture the appearance-changing features.

6.3 Appearance-changing objects

In this section we define and explain the importance of three categories of appearance-changing objects that we aim to track using RoCap. We collected and captured eight items from the three categories with RoCap.

6.3.1 Deformation

Deformation refers to changes in the shape or size of an object due to external forces applied during manipulation (i.e., force of the hand and gravity). Objects with naturally deformable features can include soft and malleable objects such as fabric materials, clothing and plush toys/stuffed animals. During manipulation, the objects are affected by gravity all the time, leading to the deformation while the user is moving the objects into different orientation. We picked two plush toys of different stiffness, anpanman (stiffer) (Figure 6.3(5)) and frog (more flexible)(Figure 6.3(4)) as examples of the deformable objects.

6.3.2 Viewing-angle dependent

The visual appearance of viewing-angle dependent objects includes two main sub-categories of objects, transparent objects (e.g., glass) and reflective objects (e.g., polished metal). Appearance of transparent objects depends on the background behind them, which may contain the environment and the user’s hands. Tracking and estimating the pose of such transparent objects is a known challenge [55] and hand manipulation may make this even harder. Appearance of reflective objects on the other hand depends on the environment in front and around it. We picked a conical flask and a plastic bottle(Figure 6.3(1, 2)) as representations of transparent objects of different level of translucency (Figure 6.7b). We also included a reflective pitcher to represent reflective object.

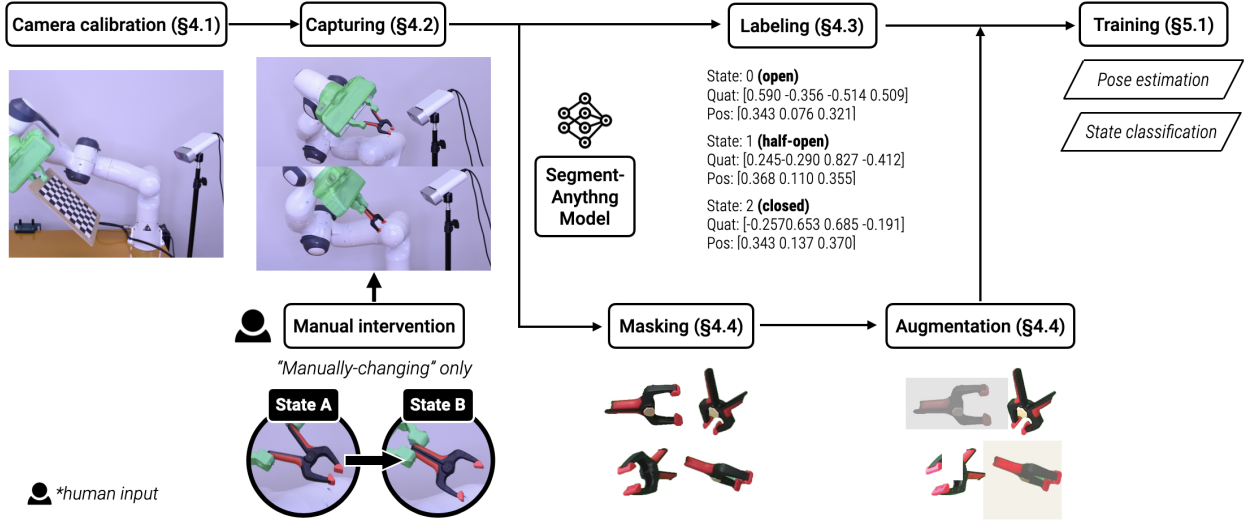


Figure 6.4: *Overview of RoCap*. RoCap pipeline consists of camera calibration (§6.4.1), data capturing (§6.4.2), data labeling (§6.4.3), data processing (§6.4.4) and data augmentation (§6.4.4). By training on an existing deep learning framework, RoCap achieves object segmentation, state classification and pose estimation for appearance-changing objects.

6.3.3 Articulated

Objects with articulated features refer to objects whose appearance changes through manual manipulation or interaction. These changes can occur due to the inherent function of the physical objects. For examples, various handheld tools will change their mechanical forms while being manipulated by human. We selected three manually-changing objects: a clamp, a pair of scissors and, a head of spray bottle to represent two different types of manual gripping and hand operation (holding and pinching).

6.4 RoCap pipeline

In this section, we will introduce the design of the RoCap pipeline, which is easily replicable using any 6-DoF robotic arm, we document the essential knowledge and technical challenges addressed including (i) camera calibration, (ii) data collection, (iii) data labeling and (iv) data pre-processing. Figure 6.4 shows the overview of the RoCap pipeline and we

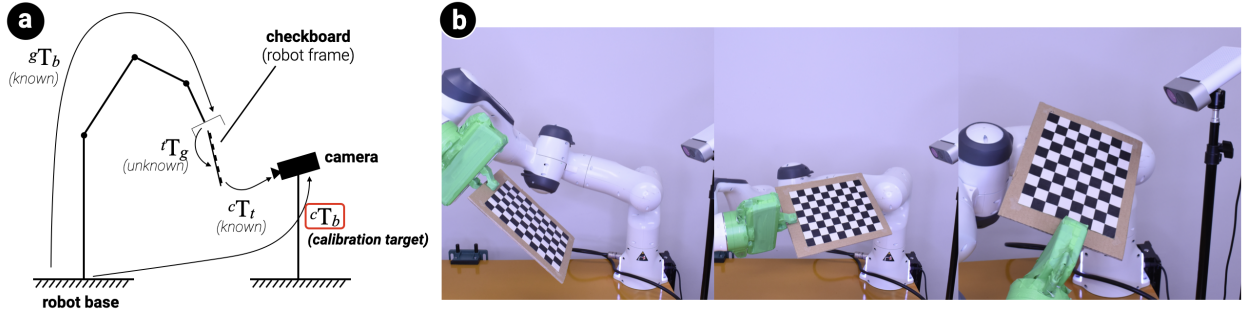


Figure 6.5: Illustration of the eye-to-hand camera calibration (a). The robotic arm grip a checkerboard and move to multiple positions and orientations for an accurate calibration (b).

discuss each step in details as follows.

6.4.1 Eye-to-hand camera calibration

The first step of RoCap pipeline is to calibrate the camera to the robotic arm (Figure 6.4a). During data collection, the robotic arm will hold the target object using a gripper and the camera is standing on the side to capture the images. In this setup, the pose of the object in the image refers to the homogeneous transformation of the object from its reference frame to the camera's reference frame. This is a typical hand-eye calibration problem because as shown in Figure 6.5. Assuming the object has the same pose as the end-effector, the goal is to calculate the transformation matrix of the gripper to the camera: cT_g , which can be calculated from the following equation:

$${}^cT_g = {}^cT_b \cdot {}^bT_g \quad (6.1)$$

In which bT_g refers to the transformation from the gripper to the base of the robotic arm which could be calculated by forward kinematics [41] while cT_b refers to the transformation from the base of the robotic arm to the camera frame, which is unknown.

To calculate cT_b , a camera calibration step is required which can be accomplished by using a checkerboard with known size of the squares, which is illustrated in Figure 6.5b . By moving the robotic arm to multiple configuration, cT_g can be calculated from the following

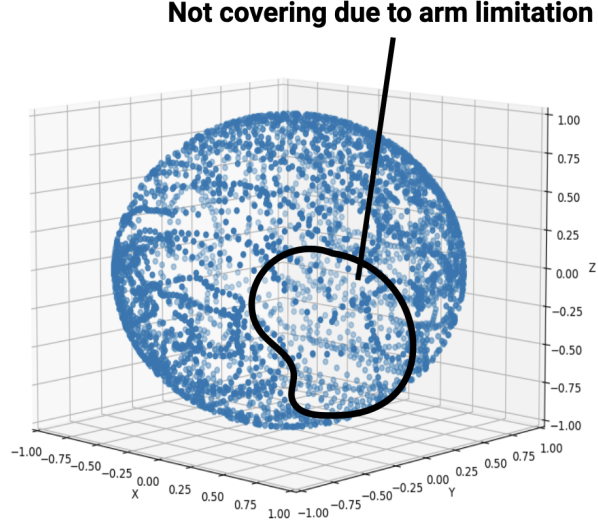


Figure 6.6: Pose coverage in RoCap capturing pipeline.

$AX = XB$ equations:

$$\begin{aligned}
 gT_b^{(1)} bT_c cT_t^{(1)} &= gT_b^{(2)} bT_c cT_t^{(2)} \\
 (gT_b^{(2)})^{-1} bT_g^{(1)} gT_c &= gT_c cT_t^{(2)} (cT_t^{(1)})^{-1} \\
 A_i X &= X B_i
 \end{aligned} \tag{6.2}$$

Here cT_t refers to the transformation from the checkerboard to the camera frame, which can be calculated knowing the size of the pattern [164]. Then the calibration target cT_b can be calculated from Eq. 6.1.

After the camera is calibrated, the next step is to capture the image data of the objects.

6.4.2 Data collection

As mentioned in the previous sections, RoCap collect data of the objects that exhibit the appearance-changing features. More specifically, RoCap collects objects categorized in four types of appearance-changing features: deformable, reflective, transparent and articulated.

6.4.2.1 Pose coverage

The goal of the capturing is simple: capture the images of the objects from as many angles as possible to have a good coverage of all the potential pose. Quaternions possess the advantage of representing each rotation without introducing any ambiguity. However, directly sampling quaternions proves to be a challenging task. To overcome this obstacle and achieve comprehensive coverage of poses, we opt for sampling Euler angles with a specific step of degrees for each yaw, pitch, and roll channel. Once we have obtained the Euler angles, they are converted into quaternions. These quaternions are then utilized to calculate the arc distance between each orientation. This methodology is employed due to the inherent redundancies that can arise from sampling Euler angles. By computing the arc distance of quaternions, we effectively eliminate these redundancies. The threshold for eliminating redundancies is set at 0.35, roughly equivalent to a 20° azimuth angle.

However, due to the hardware limitation of the robotic arm (*e.g.*, the joints may have a limited range of motion), RoCap cannot cover the whole possible poses sampled in this process. We use the inverse kinematics solver and path planners in ROS and achieve the final sampling of the poses RoCap supports. Figure 6.6 visualizes the coverage of the poses in RoCap. Noted that in existing data collection method where the objects are placed on floor, there will be at least half of the poses not capturable because it is occluded by the contacting ground.

6.4.2.2 Capturing process

During the capturing, a human user will be required to hand the target object to the robotic arm and the robotic arm will move along the designed path and the camera capture the RGB images on each sampled point.

For deformable, reflective and transparent objects, there is no further actions from the users as the change of the appearance happens naturally when the object is oriented to different direction while being manipulated by the robotic arm 6.7abc. For articulated

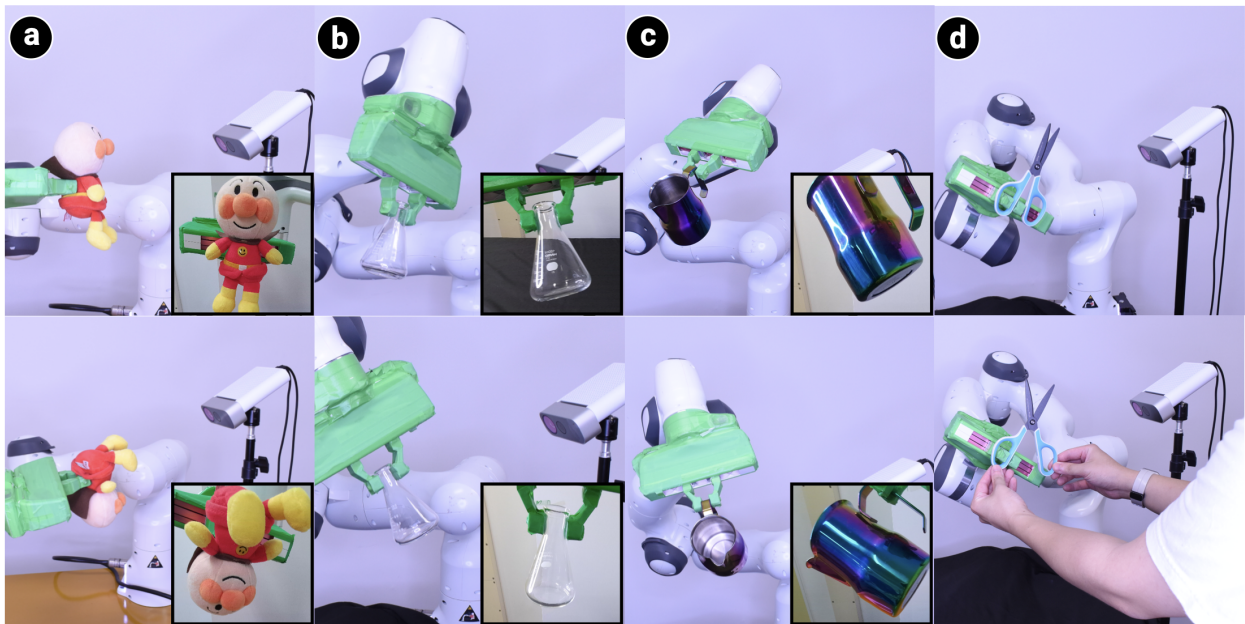


Figure 6.7: RoCap captures the appearance-changing feature of deformable objects (a), viewing-angle dependent objects including transparent objects (b) and reflective objects (c), and objects with articulated features (d). Human operator is needed if the robotic arm is not able to change the states automatically (d).

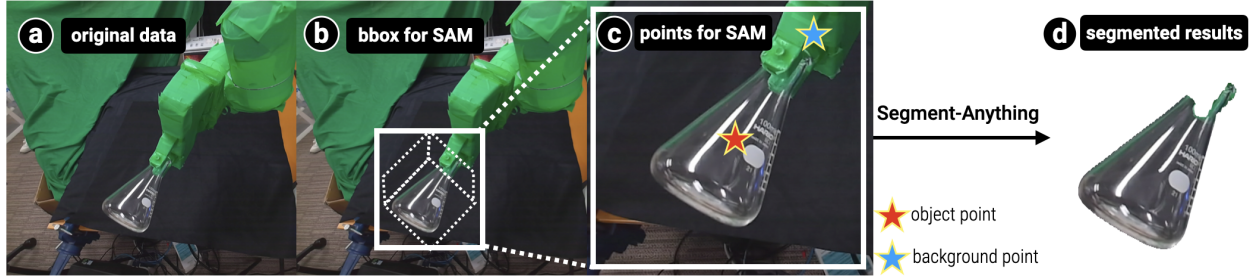


Figure 6.8: Data processing of data collected in by RoCap. RoCap generates mask for each image (d) by prompting SAM with bounding box (b) and points (c).

objects, actions need to be taken in order to change the mechanical states of the objects. The manually-changing action can be achieved either by human or the robotic arm automatically depending on the capability of the robotic arm to change the appearance. As is shown in Figure 6.1b, the size of the clamp is small enough to be grasped by the gripper. And the clamp is expected to have multiple states such as closed, open, and mid-open states. Without the help of human, the gripper could be able to change the states of the clamp by applying different forces on the parallel grippers. However, for the pair shown in Figure 6.7d, the robotic gripper is not able to automatically change the states because the handle is too wide for the parallel gripper when it is in open state. This is a typical robotic manipulability problem as mentioned in [119]. For the case that a robotic arm cannot establish firm gripping on the object, a human operator will be required to manually change the opening angle of the scissors in the interval between the capturing of different states.

6.4.3 Data labeling

The transformation from the base frame of the robotic arm to the target object is logged for each captured image in a 4×4 homogeneous transformation matrix. Then the transformation from camera frame to the object could be calculated using Equation 6.1. The rotation and the translation serve as the 6D pose label for the object in each image as shown in Figure 6.4c left.

6.4.4 Data processing and augmentation

After capturing the data with the ground truth labels of objects using RoCap, crucial processing step must be performed to facilitate subsequent pose estimation training. A typical object pose estimation task comprises two subtasks: *(i)* segmenting the object from the scene, and *(ii)* predicting the orientation of the segmented object. Therefore, the processing steps involves generating object masks for each label and augmenting the data to adapt to various environment in application.

6.4.4.1 Generating masks

RoCap leverages the recent emergence of Segment-Anything Model (SAM) [96] which is capable of producing high quality segmentations given points or bounding boxes as prompts. For each image captured by RoCap, the subsequent procedures must be executed:

Bounding box As the camera is calibrated to the robotic arm’s coordinate frame, we generate the initial bounding box of the object by assuming the robotic arm is holding a 15x15x15 cm cube. We then project the cube’s coordinates onto the camera’s 2D plane to obtain the bounding box (Figure 6.8b). Generally, this method yields satisfactory masks for objects that are distinct and easily identifiable in the image. However, complications arise when objects are partially obscured by the robotic arm, difficult to distinguish (e.g., a flask whose appearance is influenced by the background), or even entirely invisible. To address these challenges, an additional filtering process is introduced. This process either segments the semi-occluded objects or discards the invisible data.

Filtering To improve the quality of the masked objects, we leverages the interaction with SAM by providing additional prompts (points) to specify the objects and background (Figure 6.8c). Specifically, we incorporated two additional steps: *(i)* we provide additional prompts for the SAM to highlight the object’s location and *(ii)* we wrap the gripper in green tape to reduce its potential interference with segmentation performance.

- **Providing additional prompts for the SAM to highlight the object’s location.** Given that the gripper consistently holds the objects, we can infer that the center of the 15x15x15 cm cube corresponds to the object. Thus, we add the projected pixel coordinate of this center as a point prompt for the SAM, indicating the object’s location.
- **Removing green background.** Given that the gripper can partially obscure the object, it might predominantly appear within the bounding box. This could lead the SAM to mistakenly segment the gripper as the target object. To counteract this, we detect the green regions in the image, which are presumed to represent the gripper. We then calculate the geometric center of these regions and use its coordinates to provide the SAM with a prompt, pointing out the undesired areas.

6.4.4.2 Data augmentation

We augment each masked image of the object with random exposure, contrast, saturation, etc. via Albumentations [21] to achieve better generalizability.

6.5 Evaluation

To demonstrate the feasibility of our data collection pipeline, we conducted both quantitative and qualitative evaluation of the model trained on our data to compare with a few-shot learning pose estimation approach Gen6D [128]. Gen6D has shown competitive performance on any custom object by using a single video as input for 3D reconstruction (via COLMAP [190]) and performed feature matching based on the image and resulting pointcloud.

We evaluated the model in two settings, *controlled* setting where the ground truth can be reliably obtained for quantitative evaluation, and *application* setting, where the user manipulates the object during pose estimation for qualitative evaluation, as the ground truth pose cannot be obtained easily.

6.5.1 Implementation of pose estimation pipeline

Before we delve into the result of quantitative evaluation, We will discuss the pose estimation pipeline first. As mentioned earlier, the pose estimation pipeline should consists of one model for segmenting the target object and another for predicting the orientation based on the segmented output. For objects with manually-modifiable states (e.g., scissors), an additional state classifier is employed.

For the segmentation task, we leveraged a recent advancement based on SAM: HQTrack [232]. It is a zero-shot approach and requires no training while being able to consistently produce high-quality segmentation of target objects in videos.

For the orientation estimation, the model is a VGG16 model, pretrained on ImageNet [42], followed by a fully connected layer outputting the quaternion and the 2D pixel location of the object. The loss function is a combined loss of the Geodesic Loss on the quaternion prediction and the MSE Loss of the displacement prediction. We train the model on the augmented data for 120 epochs, using the Adam optimizer with a learning rate of 0.0001.

For the state classification, the model is a MobileNet V3 [80], pretrained on ImageNet [42], followed by a fully connected layer, and the output dimension is equivalent to the number of the states of the object. We train the model on the augmented data for 120 epochs, using the Adam optimizer with a learning rate of 0.0001.

6.5.2 Quantitative evaluation

For quantitative evaluation, we modified the environment by changing the camera angle and updating the background (Figure 6.9). Using a newly designed trajectory for the robotic arm, we sampled 1041 data entries. The accuracy threshold for pose estimation remained consistent with our training data parameters: set at 0.35 or an azimuth angle of 20° .

To clarify, our evaluation only focused on the accuracy of the orientation prediction, since RoCap rely on prior to determine the position of the object. For Gen6D, we could not modify its training pipeline to incorporate HQTrack to enhance its object detection.

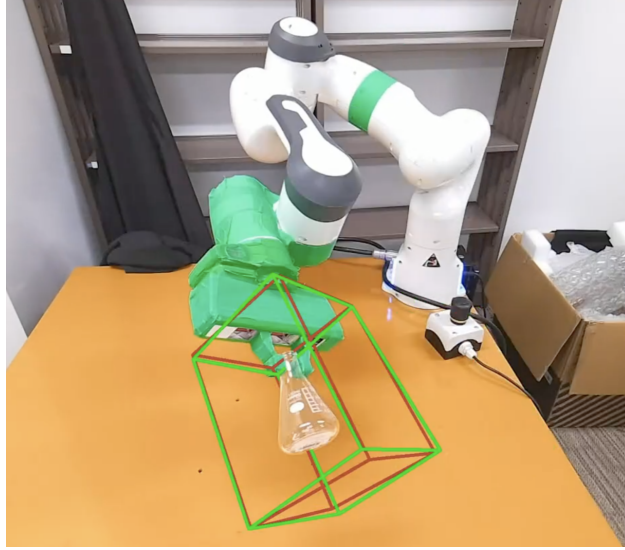


Figure 6.9: Quantitative evaluation setup. The green bounding box represents the ground truth and the read bounding box represents the predicted pose.

Instead, we adhered to the guidelines provided for pose estimation on custom objects as outlined in Gen6D’s guidelines¹. Specifically, we performed 3D reconstruction of the object using COLMAP [190] and followed the preprocessing procedure in the guideline.

	anpanman	frog	pitcher	flask	bottle	scissors	clamp	spray
RoCap	91.9	61.9	73.7	87.1(66.9)	71.9	83.4	42.0	87.6
Gen6D [128]	19.6	12.9	12.7	16.2	16.9	38.3	19.4	28.4

*The number in parentheses indicates flask accuracy in a different background (Figure 6.9).

Table 6.1: Quantitative evaluation result. The numbers indicate the average precision at 20° azimuth error.

For objects with multiple manual states, test data is gathered for each state, and the model is evaluated accordingly. The results represent the mean accuracy across all states. Specifically, the flask was tested against two different backgrounds: its original setting (a black background) and an alternate setting with a typical orange-colored desk surface. Table

¹https://github.com/liuyuan-pal/Gen6D/blob/main/custom_object.md

6.1 shows the accuracy comparison between our approach and Gen6D.

We note that the accuracy is much lower in our testing result as compared to the result Gen6D demonstrated in their paper. This could be due to several factors:

- 3D reconstruction failures (*e.g.*, Figure 6.2).
- Data collection with objects in static positions, leading to challenges when the object’s unseen side becomes visible during manipulation (*e.g.*, a plush toy might be placed face-up on a table during data collection).
- Gen6D’s documented issue with size-changing objects in frames (as the object moves closer and further away from the camera), as mentioned in their GitHub issues².

The results indicate that a simple pose estimator trained with data from RoCap can deliver relative working pose estimation performance. However, the quantitative findings also reveal some limitations. For example, the accuracy of clamp is relatively low compared to other objects due to ambiguity caused by its symmetry. Additionally, objects whose appearances are environment-dependent demonstrate inconsistent performance under varying backgrounds. More details are discussed in the limitation in Sec. 6.6.1.

6.5.3 Qualitative evaluation

To test the performance in the application setting, due to the difficulty in collecting ground truth, we conducted a qualitative evaluation on videos of humans manipulating the objects. Figure 6.10 shows the qualitative comparison between model trained on RoCap data and Gen6D. For example, RoCap recorded both closed and open states during data collection for the pair of scissors. This allowed it to provide viable pose estimation for the open state (Gen6D which struggled with the unobserved state). Please refer to the supplementary materials for the video.

²<https://github.com/liuyuan-pal/Gen6D/issues/29>

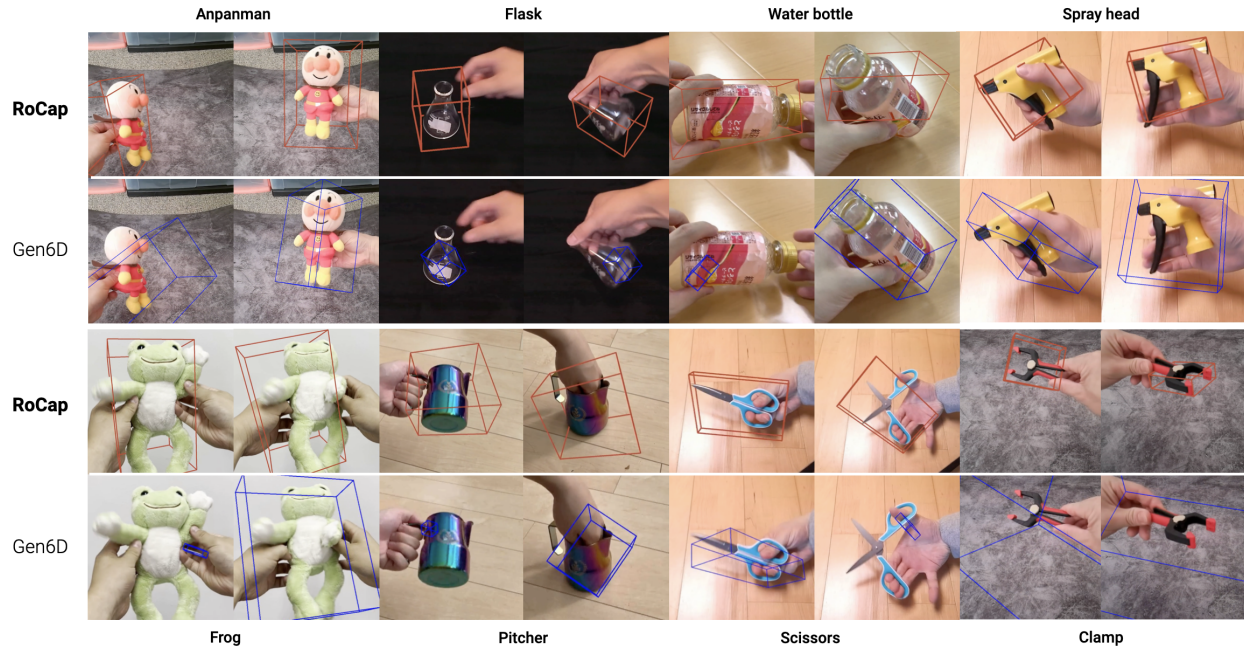


Figure 6.10: Qualitative evaluation of the eight objects.

6.6 Discussion

6.6.1 Limitations

The quantitative and qualitative evaluation has demonstrated the feasibility and potential of our data collection method. However, the result also shows certain limitations. Below, we will discuss the limitations from the perspectives of data capturing, model performance and other constraints.

Data capturing While RoCap addresses the data capturing of appearance-changing objects, it requires the objects have distinct appearances in different defined poses. One typically example that is challenging for RoCap is cloth, which is highly deformable. Its extreme flexibility results in a loss of the pose information when being manipulated by the robotic arm. As illustrated in Figure 6.11, the piece of cloth is nearly identical in two different poses manipulated by the robotic arm.

On the other hand, as currently we target objects that people can easily change their

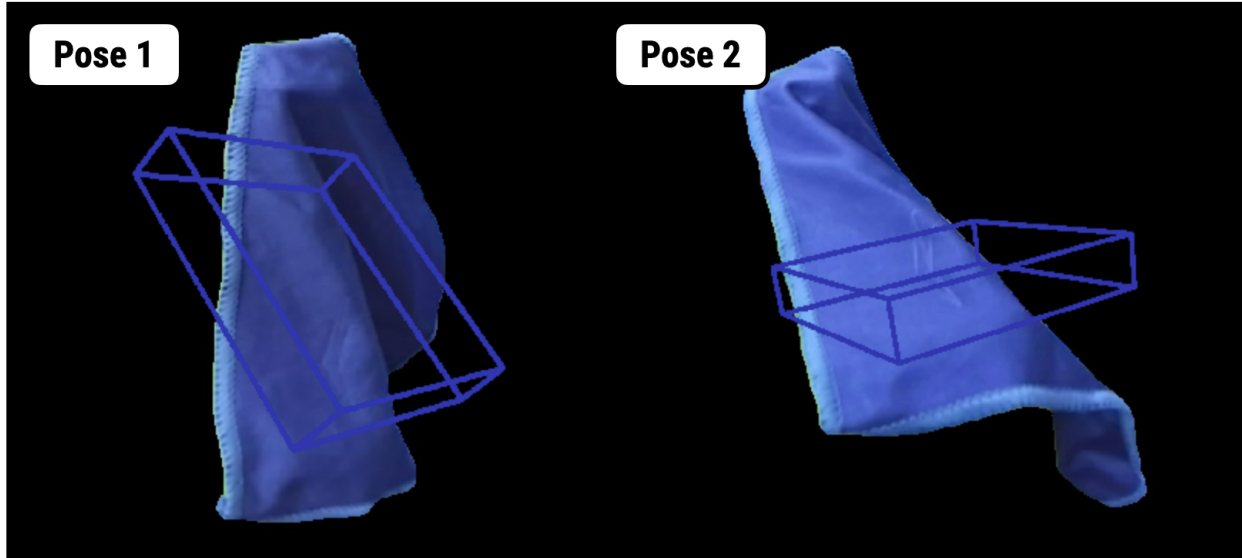


Figure 6.11: Failure case for a highly deformable cloth.

appearances with hands, leading to the target object size ranging from $0.5x \sim 1.5x$ of a palm size. Additionally, our robotic arm’s mechanical gripper, with a maximum gripping width of 80mm, further constrains the size of objects it can handle. However, this limitation can be resolved when a system applies our method to a larger scale robotic arm (*e.g.*, in a mass manufacturing setting).

Model performance Indicated in the evaluation results, the pose estimation pipeline does not handle symmetric object well. While this has been an open challenge in object pose estimation [204], recent work have proposed different network architecture to address this issue [217, 194]. While addressing symmetry is beyond the purview of this paper, future enhancements could incorporate a sophisticated pose estimator or gather supplementary data like depth via depth cameras.

Furthermore, variations in the environment from the capturing stage may also affect the model performance, especially for viewing-angle dependent objects. While it is feasible to maintain an environment similar to the capturing setup (*e.g.*, using a black background when operating a transparent flask), future improvements could include varying environmental factors such as different lighting conditions [40, 161]. Additionally, future work could introduce

other augmentation method such as [234] to adapt to various environment.

Other constraints Currently the need of a robotic arm may require a lab setting. However, the pipeline can also be applied to scenarios such as *(i)* product manufacturers collect data and train a model for their product, and include it as part of their solution package and *(ii)* home users asks their robot to train a model for their own object when robots are more accessible in the future, and later the user uses the model to estimation the pose in specific applications.

6.6.2 Handling Occlusion

Occlusion happens in different scenarios, including the objects being manipulated by hands during interaction, or the objects being held by the robotic arm during data capturing. While the hand occlusion does have an impact on the model performance trained on RoCap data, our pipeline is less impacted compared to Gen6D as shown in the qualitative results. This is due to the fact that during the capture phase, the robotic arm may partially obscure the object throughout the capturing process, which simulates hand occlusion in the training data.

To further address the occlusion problem, one possible approach is to introduce a hand-like robotic hand during the data capturing process. For example, anthropomorphic robotic hands, such as those presented in [59], can closely mimic human hand movements and provide more realistic interaction scenarios for data collection. By using a robotic hand, it is possible to better account for occlusions that occur during human-object interactions and develop models that can better predict user intent in such cases. Additionally, to address occlusion caused by the robotic arm during data capturing, multiple cameras could be employed to ensure the complete visibility of the objects being captured.

6.6.3 Automatic Changing of Mechanical States

As mentioned in the paper, certain articulated objects necessitate human intervention to change their states, as they cannot be manipulated by the parallel gripper of the robotic arm [119]. Examples of such objects include those that require a large range of motion or those that demand bi-manual operation. Recent research in HCI has proposed different methods of attaching mechanisms to the physical object to automatically actuate the motion without human intervention [119, 118, 116], which can be potentially leveraged by future data collection system using robotic arms to automatically collect a large amount of data. By automating the data collection process, it is possible to scale up the dataset and sample object states at smaller intervals. For instance, instead of having discrete states of a clamp, we can sample from a continuous parameter space evenly while capturing. This would enable the prediction of the continuous parameter such as the angle of a pair of scissors, thus opening up a wider range of applications. Future direction should include how to design mechanisms that will not affect the appearance of the objects during capturing while being able to actuate the objects.

6.6.4 Leveraging Robots for Large-Scale Data Collection

Robots possess the capability to perform repetitive tasks consistently and efficiently. Researchers in computer vision and HCI have explored various approaches to employing robots for data collection across a diverse range of applications [33, 60, 133]. This has opened up new opportunities for augmenting tasks that necessitate a substantial amount of repetitive work, such as data collection for multiple objects, through the integration of robotic systems. By leveraging robotic systems, researchers can not only streamline the data collection process but also minimize human error and fatigue. This can lead to the acquisition of more accurate and reliable datasets, which are critical for the development and evaluation of advanced algorithms and models.

In addition to automating repetitive tasks, robotic systems can be equipped with various

sensors and end effectors to collect multimodal data, such as visual, tactile, and auditory information. This can significantly enrich the datasets and provide researchers with a more comprehensive understanding of the objects and environments being studied. As robotics technology continues to advance, we can expect even more sophisticated and versatile robotic systems to be employed in the data collection process. This will ultimately lead to more robust, accurate, and diverse datasets, which will contribute to the improvement of various computer vision and HCI applications.

CHAPTER 7

Summary

Smart homes have been a popular topic in both the research community and the industry for decades since the 1970s, with the goal of elevating the quality of life for users by automating various applications. With a large amount of legacy objects have not yet become *smart*, researchers and practitioners have devoted considerable efforts towards incrementally enhancing these legacy items. The aim is to enable them to autonomously execute *tasks* for users, using diverse methods that are both financially and environmentally sustainable. Such tasks involve physical tasks, which play a vital role in assisting users in their daily lives, especially for those with limited mobility, or people in a situational impairments (*e.g.*, occupied hands while holding groceries).

While there has been only limited support for the automation of the physical tasks, the goal of this dissertation is to enable legacy objects to perform physical tasks for users. In order to achieve this, two challenges have to be addressed:

C1: *How can we enable users to customize the augmentation of physical objects to be smart?*

The vast diversity of legacy objects makes crafting a universal solution for their smart transformation challenging. Consequently, user-driven customization becomes imperative. Yet, the intricacies of augmentation—spanning mechanical design, motion planning, electronic structuring, and personal fabrication—demand specialized expertise. This presents a gap between users’ limited knowledge and the complexity of the design process. It’s crucial, therefore, to simplify this complex procedure, making it accessible for general users to personalize the smart adaptations of their physical objects.

C2: *How can we enable physical objects to be easier to perform physical tasks?*

Legacy objects were predominantly designed with human interaction in mind and not for autonomous functionality. When introducing smart capabilities to these objects, the challenge lies in determining how to streamline task automation. This can be through either direct actuation of the object or intervention via an external robotic arm. Thus, it becomes crucial to introduce designs that render these physical objects more adept at autonomously conducting physical tasks.

During my PhD, I have explored different approaches to address these challenges. Foremost, I propose two methods to automate the physical tasks for legacy objects: *(i)* directly augmenting the legacy objects with robotic actuation to perform physical tasks without external assistance and *(ii)* augmenting the legacy objects to be more easily manipulated by external robotic arms.

In Chapter 3, I developed a computational design tool to enable users to create customized add-on mechanisms to automate the physical tasks of legacy objects (*e.g.*, adjusting the angle of a desk lamp automatically). This tool allows users to use only a video demonstrating the manipulation as the input and outputs the 3D printable mechanisms automatically.

Extending the work in Chapter 3, I developed another computational tool to enable users to design embedded mechanism in 3D models of physical objects (Chapter 4). These embedded mechanisms are designed to augment the default functionalities of the physical objects. By specifying the highlevel motion points of the target tasks, users are able to use the tool to automatically generate 3D printable components of the final physical objects.

Beyond directly augmenting the legacy objects to perform physical tasks without external assistance, I also explored the approach of augmenting the legacy objects to be more easily manipulated by external robotic arms. Augmenting the legacy objects to be more manipulable by robotic arms consists of two aspects: *(i)* better to be sensed by the robotic arms (Chapter 6) and *(ii)* better to be manipulated by the robotic arms (Chapter 5).

In Chapter 5, I identified the issues of the manipulability of existing physical handheld objects and proposed a design of a magnetic gripper along with mechanical mechanisms to augment the manipulability of physical objects. The magnetic gripper serves as a versatile

interface between a generic robotic arm and the target physical objects that the robotic arm is able to actuate the mechanisms on the physical objects automatically.

One of the prerequisite to enable the actuation in Chapter 5 is the ability to sense the spatial configuration (i.e., the pose) of the target objects. It is a known challenge to perform pose estimation on objects that change their appearance during the manipulation (*e.g.*, articulated objects in Chapter 5). Thus in Chapter 6, I proposed a method to collect datasets of these appearance-changing objects and enable the pose estimation of such objects by training a deep learning network.

7.1 Limitations and Future Work

7.1.1 Enabling More Complex Tasks

The tools and methodologies presented in this dissertation primarily address a select subset of physical tasks. While they have proven effective in their targeted applications, the realm of automation encompasses a wider range of complexities.

Currently, the design of the mechanical mechanisms only addresses single-DoF (degree of freedom) motion of the physical objects. While this approach simplifies the design and operational parameters, it inherently limits the range and complexity of tasks that these objects can undertake. Covering a wide array of physical tasks with just a single-DoF is certainly commendable, but to truly harness the potential of smart augmentation, multi-DoF systems need to be explored and integrated.

Handheld tools, for instance, often employ multi-DoF systems to offer users a wide range of motions. A classic example is the 3-DoF ball joint found in many modern-day accessories such as selfie rods or camera tripods.

Beyond these examples, the potential of multi-DoF systems becomes even more evident when we look at objects with multiple 1-DoF joints. For example, some objects are articulated with multiple movable parts, *e.g.*, multi-functional pliers, Swiss army knife, Rubik's

cube. Technically, mechanisms presented in the thesis can be extended to more parts by enabling additional motion at a time. However, practically, too many mechanisms might not be fittingly added to an object and might even interfere with one another.

In essence, while single-DoF designs have paved the way for the initial exploration of smart object augmentation, the future envisions a more comprehensive integration of multi-DoF systems. Such an evolution will not only enhance the capabilities of smart objects but also bring them a step closer to mimicking the dexterity and adaptability of human beings. Furthermore, this also highlights the need for a more robust design tool that can support the creation of multi-DoF mechanisms.

7.1.2 Adapting to Humans and Environments

Legacy objects, once augmented, need to not just perform tasks but also able to adapt to changing environments or unpredicted scenarios. Furthermore, as users engage with such augmented objects, they may have feedback or preferences on how tasks are executed. Thus it is also crucial to create a system where user feedback can be seamlessly integrated to refine task performance.

Incorporating machine learning techniques throughout the design and operational phases can harness this adaptability. By analyzing user interactions, these intelligent systems can tailor designs that are more attuned to user expectations. Moreover, post-augmentation, continuous learning from both environmental cues and direct user feedback is essential. This iterative process ensures that the system's interactivity remains dynamic, evolving in tandem with user needs and environmental nuances.

7.1.3 Studying Human Behavior

The introduction of augmented functionalities to everyday objects ushers in a new paradigm in human-object interactions. When inanimate objects begin to exhibit "alive" qualities, it inevitably alters the way humans perceive, interact with, and rely on them. A shift from

passive objects to active, responsive entities poses fascinating questions for future research:

Research questions on the social science or psychology may arise, for example, will users begin attributing human-like traits or emotions to these objects? Or how will 'alive' objects fit into our social interaction. Exploring these areas will not only enhance the design and functionality of such augmented objects but will also pave the way for a more harmonious integration of these entities into our daily lives.

BIBLIOGRAPHY

- [1] Gift a lot. Smart wifi electric curtains tracks and motorized roller blinds. <https://giftalot.com/>, 2019. (Accessed on 04/01/2019).
- [2] Adel Ahmadyan, Liangkai Zhang, Artsiom Ablavatski, Jianing Wei, and Matthias Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7822–7831, June 2021.
- [3] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *American Statistician*, 1992.
- [4] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.
- [5] Abul Al Arabi, Jiahao Li, Xiang’Anthony Chen, and Jeeun Kim. Mobiot: Augmenting everyday objects into moving iot devices using 3d printed attachments generated by demonstration. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2022.
- [6] Rahul Arora, Rubaiat Habib Kazi, Tovi Grossman, George Fitzmaurice, and Karan Singh. SymbiosisSketch. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI ’18*, pages 1–15, New York, New York, USA, 2018. ACM Press.
- [7] Daniel Ashbrook, Shitao Stan Guo, and Alan Lambie. Towards augmented fabrication: Combining fabricated and existing objects. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1510–1518. ACM, 2016.
- [8] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.
- [9] Vincent Babin and Clément Gosselin. Picking, grasping, or scooping small objects lying on flat surfaces: A design approach. *The International Journal of Robotics Research*, 37(12):1484–1499, 2018.
- [10] Vincent Babin and Clément Gosselin. Mechanisms for robotic grasping and manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 4, 2020.
- [11] Vincent Babin, David St-Onge, and Clément Gosselin. Stable and repeatable grasping of flat objects on hard surfaces using passive and epicyclic mechanisms. *Robotics and Computer-Integrated Manufacturing*, 55:1–10, 2019.

- [12] M Bacher, S Coros, and B Thomaszewski. LinkEdit: Interactive Linkage Editing using Symbolic Kinematics. *Acm Transactions on Graphics*, 2015.
- [13] Moritz Bächer, Bernd Bickel, Doug L James, and Hanspeter Pfister. Fabricating articulated characters from skinned meshes. *ACM Transactions on Graphics (TOG)*, 31(4):1–9, 2012.
- [14] Moritz Bächer, Stelian Coros, and Bernhard Thomaszewski. Linkedit: interactive linkage editing using symbolic kinematics. *ACM Transactions on Graphics (TOG)*, 34(4):1–8, 2015.
- [15] Connelly Barnes, David E Jacobs, Jason Sanders, Dan B Goldman, Szymon Rusinkiewicz, Adam Finkelstein, and Maneesh Agrawala. Video puppetry: a performative interface for cutout animation. In *ACM SIGGRAPH Asia 2008 papers*, pages 1–9. 2008.
- [16] Ron Berenstein, Averell Wallach, Pelagie Elimbi Moudio, Peter Cuellar, and Ken Goldberg. An open-access passive modular tool changing system for mobile manipulation robots. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 592–598. IEEE, 2018.
- [17] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. *Springer handbook of robotics*, pages 1371–1394, 2008.
- [18] Aude Billard and Danica Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446):eaat8414, 2019.
- [19] Mayara Bonani, Raquel Oliveira, Filipa Correia, André Rodrigues, Tiago Guerreiro, and Ana Paiva. What My Eyes Can’t See, A Robot Can Show Me. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS '18*, pages 15–27, New York, New York, USA, 2018. ACM Press.
- [20] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014.
- [21] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Alumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.
- [22] Jacques Cali, Dan A Calian, Cristina Amati, Rebecca Kleinberger, Anthony Steed, Jan Kautz, and Tim Weyrich. 3d-printing of non-assembly, articulated models. *ACM Transactions on Graphics (TOG)*, 31(6):1–8, 2012.
- [23] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

- [24] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, et al. Dexycb: A benchmark for capturing hand grasping of objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9044–9053, 2021.
- [25] Xiang ‘Anthony’ Chen. Making Fabrication Real. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 17–20. ACM, 2016.
- [26] Xiang ‘Anthony’ Chen, Stelian Coros, and Scott E Hudson. Medley: A Library of Embeddables to Explore Rich Material Properties for 3D Printed Objects. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 162. ACM, 2018.
- [27] Xiang ‘Anthony’ Chen, Stelian Coros, Jennifer Mankoff, and Scott E Hudson. 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 and Technology*, pages 73–82. ACM, 2015.
- [28] Xiang ‘Anthony’ Chen, Jeeun Kim, Jennifer Mankoff, Tovi Grossman, Stelian Coros, and Scott E Hudson. Reprise: A Design Tool for Specifying, Generating, and Customizing 3D Printable Adaptations on Everyday Objects. In *the 29th Annual ACM Symposium on User Interface Software and Technology*. ACM, 2016.
- [29] Xiang ‘Anthony’ Chen, Ye Tao, Guanyun Wang, Runchang Kang, Tovi Grossman, Stelian Coros, and Scott E Hudson. Forte: User-Driven Generative Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 496. ACM, 2018.
- [30] Keene Chin, Tess Hellebrekers, and Carmel Majidi. Machine learning for soft robotic sensing and control. *Advanced Intelligent Systems*, 2(6):1900171, 2020.
- [31] Hyungjun Cho, Han-Jong Kim, JiYeon Lee, Chang-Min Kim, Jinseong Bae, and Tek-Jin Nam. Iotizer: A versatile mechanical hijacking device for creating internet of old things. In *Designing Interactive Systems Conference 2021*, pages 90–103, 2021.
- [32] Changhyun Choi, Yuichi Taguchi, Oncel Tuzel, Ming-Yu Liu, and Srikumar Ramalingam. Voting-based pose estimation for robotic assembly using a 3d sensor. In *2012 IEEE International Conference on Robotics and Automation*, pages 1724–1731. IEEE, 2012.
- [33] Toby Chong, I-Chao Shen, Nobuyuki Umetani, and Takeo Igarashi. Per garment capture and synthesis for real-time virtual try-on. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 457–469, 2021.
- [34] Andrea S Ciullo, Janne M Veerbeek, Eveline Temperli, Andreas R Luft, Frederik J Tonis, Claudia JW Haarman, Arash Ajoudani, Manuel G Catalano, Jeremia PO Held, and Antonio Bicchi. A novel soft robotic supernumerary hand for severely affected

- stroke patients. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(5):1168–1177, 2020.
- [35] Gabe Cohn, Sidhant Gupta, Tien-Jui Lee, Dan Morris, Joshua R. Smith, Matthew S. Reynolds, Desney S. Tan, and Shwetak N. Patel. An ultra-low-power human body motion sensor using static electric field sensing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 99–102, New York, NY, USA, 2012. ACM.
- [36] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. Computational design of mechanical characters. *ACM Transactions on Graphics*, 32(4):1, jul 2013.
- [37] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W Sumner, Wojciech Matusik, and Bernd Bickel. Computational design of mechanical characters. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.
- [38] Anton Konushin Danila Rukhovich, Anna Vorontsova. Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. pages 2397–2406, 2022.
- [39] Scott Davidoff, Nicolas Villar, Alex S Taylor, and Shahram Izadi. Mechanical hijacking: how robots can accelerate ubicomp deployments. In *UbiComp*, pages 267–270, 2011.
- [40] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 145–156, 2000.
- [41] Jacques Denavit and Richard S Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. 1955.
- [42] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [43] Ruta Desai, Fraser Anderson, Justin Matejka, Stelian Coros, James McCann, George Fitzmaurice, and Tovi Grossman. Geppetto: Enabling semantic design of expressive robot behaviors. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2019.
- [44] Ruta Desai, James McCann, and Stelian Coros. Assembly-aware Design of Printable Electromechanical Devices. In *The 31st Annual ACM Symposium on User Interface Software and Technology - UIST '18*, pages 457–472, New York, New York, USA, 2018. ACM Press.
- [45] Ruta Desai, Margarita Safonova, Katharina Muelling, and Stelian Coros. Automatic design of task-specific robotic arms. *arXiv preprint arXiv:1806.07419*, 2018.

- [46] Ruta Desai, Ye Yuan, and Stelian Coros. Computational abstractions for interactive design of robotic devices. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1196–1203. IEEE, 2017.
- [47] dezeen. Reconfigurable apartment allows residents to transform their living spaces. <https://www.dezeen.com/2018/02/15/video-white-arkitekter-dream-home-reconfigurable-apartment-movie/>, 2018. (Accessed on 04/01/2019).
- [48] Giovanni Diraco, Alessandro Leone, and Pietro Siciliano. A radar-based smart sensor for unobtrusive elderly monitoring in ambient assisted living applications. *Biosensors*, 7(4):55, 2017.
- [49] Mustafa Doga Dogan, Ahmad Taka, Michael Lu, Yunyi Zhu, Akshat Kumar, Aakar Gupta, and Stefanie Mueller. Infraredtags: Embedding invisible ar markers and barcodes using low-cost, infrared-based 3d printing and imaging tools. In *CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2022.
- [50] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1301–1310, 2017.
- [51] Mark Edmonds, Feng Gao, Xu Xie, Hangxin Liu, Siyuan Qi, Yixin Zhu, Brandon Rothrock, and Song-Chun Zhu. Feeling the force: Integrating force and pose for fluent discovery through imitation learning to open medicine bottles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3530–3537. IEEE, 2017.
- [52] Markus Ehrenmann, Oliver Rogalla, Raoul Zöllner, and Rüdiger Dillmann. Teaching service robots complex tasks: Programming by demonstration for workshop and household environments. In *Proceedings of the 2001 International Conference on Field and Service Robots (FSR)*, volume 1, pages 397–402, 2001.
- [53] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.
- [54] Erik D Engeberg, Nancy Pollard, Maximo Roa, and Zeyang Xia. Robotic grasping and manipulation competition: task pool. *Robotic Grasping and Manipulation: First Robotic Grasping and Manipulation Challenge, RGMC 2016, Held in Conjunction with IROS 2016, Daejeon, South Korea, October 10–12, 2016, Revised Papers*, 816:1, 2018.
- [55] Heng Fan, Halady Akhilesha Miththanthaya, Siranjiv Ramana Rajan, Xiaoqiong Liu, Zhilin Zou, Yuewei Lin, Haibin Ling, et al. Transparent object tracking benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10734–10743, 2021.

- [56] T. Feix, J. Romero, H. B. Schmiebmayer, A. M. Dollar, and D. Kragic. The grasp taxonomy of human grasp types. *IEEE Transactions on Human-Machine Systems*, 46(1):66–77, 2016.
- [57] David Fischinger, Peter Einramhof, Konstantinos Papoutsakis, Walter Wohlking, Peter Mayer, Paul Panek, Stefan Hofmann, Tobias Koertner, Astrid Weiss, Antonis Argyros, et al. Hobbit, a care robot supporting independent living at home: First prototype and lessons learned. *Robotics and Autonomous Systems*, 75:60–78, 2016.
- [58] Jodi Forlizzi and Carl DiSalvo. 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*, pages 258–265, 2006.
- [59] Erika Nathalia Gama Melo, Oscar Fernando Aviles Sanchez, and Darlo Amaya Hurtado. Anthropomorphic robotic hands: a review. *Ingeniería y desarrollo*, 32(2):279–313, 2014.
- [60] Ruohan Gao, Zilin Si, Yen-Yu Chang, Samuel Clarke, Jeannette Bohg, Li Fei-Fei, Wenzhen Yuan, and Jiajun Wu. Objectfolder 2.0: A multisensory object dataset for sim2real transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10598–10608, 2022.
- [61] Akash Garg, Alec Jacobson, and Eitan Grinspun. Computational design of reconfigurables. *ACM Trans. Graph.*, 35(4):90–1, 2016.
- [62] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.
- [63] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [64] Chaitanya P. Gharpure and Vladimir A. Kulyukin. Robot-assisted shopping for the blind: issues in spatial cognition and product selection. *Intelligent Service Robotics*, 1(3):237–251, jul 2008.
- [65] Yoav Golan, Amir Shapiro, and Elon Rimon. Jamming-free immobilizing grasps using dual-friction robotic fingertips. *IEEE Robotics and Automation Letters*, 5(2):2889–2896, 2020.
- [66] Anhong Guo, Jeeun Kim, Xiang ‘Anthony’ Chen, Tom Yeh, Scott E Hudson, Jennifer Mankoff, and Jeffrey P Bigham. Facade: Auto-generating tactile interfaces to appliances. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5826–5838. ACM, 2017.

- [67] David Gyimothy and Andras Toth. Experimental evaluation of a novel automatic service robot tool changer. In *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1046–1051. IEEE, 2011.
- [68] Sehoon Ha, Stelian Coros, Alexander Alspach, James M Bern, Joohyung Kim, and Katsu Yamane. Computational design of robotic devices from high-level motion specifications. *IEEE Transactions on Robotics*, 34(5):1240–1251, 2018.
- [69] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3196–3206, 2020.
- [70] Richard S Hartenberg and Jacques Denavit. A kinematic notation for lower pair mechanisms based on matrices. 1955.
- [71] Alaa Hassan and Mouhammad Abomoharam. Design of a single dof gripper based on four-bar and slider-crank mechanism for educational purposes. *Procedia CIRP*, 21:379–384, 2014.
- [72] Liang He, Huaishu Peng, Michelle Lin, Ravikanth Konjeti, François Guimbretière, and Jon E Froehlich. Ondulé: Designing and controlling 3d printable springs. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 739–750, 2019.
- [73] Liang He, Xia Su, Huaishu Peng, Jeffrey Ian Lipton, and Jon E Froehlich. Kinergy: Creating 3d printable motion using embedded kinetic energy. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, pages 1–15, 2022.
- [74] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 3d puppetry: A kinect-based interface for 3d animation. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST ’12, page 423–434, New York, NY, USA, 2012. Association for Computing Machinery.
- [75] Anuruddha Hettiarachchi and Daniel Wigdor. Annexing reality: Enabling opportunistic use of everyday objects as tangible proxies in augmented reality. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1957–1967, 2016.
- [76] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision—ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11*, pages 548–562. Springer, 2013.

- [77] Tomáš Hodan, Pavel Haluza, Štěpán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE, 2017.
- [78] Guy Hoffman and Wendy Ju. Designing Robots With Movement in Mind. *Journal of Human-Robot Interaction*, 2014.
- [79] Google Home. Smart speaker & home assistant - google store. https://store.google.com/us/product/google_home?hl=en-US, 2019. (Accessed on 04/04/2019).
- [80] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.
- [81] Zhengtao Hu, Weiwei Wan, and Kensuke Harada. Designing a mechanical tool for robots with two-finger parallel grippers. *IEEE Robotics and Automation Letters*, 4(3):2981–2988, 2019.
- [82] Xiaojun Huang, Liang Wang, Junjun Huang, Dongxiao Li, and Ming Zhang. A depth extraction method based on motion and geometry for 2d to 3d conversion. In *2009 Third International Symposium on Intelligent Information Technology Application*, volume 3, pages 294–298, Nov 2009.
- [83] Yi-Jheng Huang, Shu-Yuan Chan, Wen-Chieh Lin, and Shan-Yu Chuang. Making and animating transformable 3d models. *Computers & Graphics*, 54:127 – 134, 2016. Special Issue on CAD/Graphics 2015.
- [84] Nathaniel Hudson, Celena Alcock, and Parmit K Chilana. Understanding newcomers to 3d printing: Motivations, workflows, and barriers of casual makers. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 384–396, 2016.
- [85] Philips Hue. Smart home automation light. <https://www2.meethue.com/en-us/smart-home-automation-light>, 2019. (Accessed on 04/04/2019).
- [86] Jeffrey Ichnowski, Yahav Avigal, Justin Kerr, and Ken Goldberg. Dex-nerf: Using a neural radiance field to grasp transparent objects. *arXiv preprint arXiv:2110.14217*, 2021.
- [87] Jianbo Shi and Tomasi. Good features to track. 2002.
- [88] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, Anand Ambrosi, Camden Whitehead, and Nikolaos Vitzilaios. Fiducial markers for pose estimation. *Journal of Intelligent & Robotic Systems*, 101(4):1–26, 2021.
- [89] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. *ACM Trans. Graph.*, 29(4), July 2010.

- [90] Shohei Katakura, Yuto Kuroki, and Keita Watanabe. A 3d printer head as a robotic manipulator. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 535–548, 2019.
- [91] Shohei Katakura and Keita Watanabe. Printmotion: Actuating printed objects using actuators equipped in a 3d printer. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings*, pages 137–139, 2018.
- [92] Rubaiat Habib Kazi, Tovi Grossman, Hyunmin Cheong, Ali Hashemi, and George Fitzmaurice. DreamSketch: Early Stage 3D Design Explorations with Sketching and Generative Design. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology - UIST '17*, 2017.
- [93] Rubaiat Habib Kazi, Tovi Grossman, Hyunmin Cheong, Ali Hashemi, and George W Fitzmaurice. Dreamsketch: Early stage 3d design explorations with sketching and generative design. In *UIST*, volume 14, pages 401–414, 2017.
- [94] Charles C Kemp, Aaron Edsinger, and Eduardo Torres-Jara. Challenges for robot manipulation in human environments [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):20–29, 2007.
- [95] Han-Jong Kim, Chang Min Kim, and Tek-Jin Nam. SketchStudio: Experience Prototyping with 2.5-Dimensional Animated Design Scenarios. *Proceedings of the 2018 on Designing Interactive Systems Conference 2018*, 2018.
- [96] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.
- [97] Frank Klassner and Scott D Anderson. Lego mindstorms: Not just for k-12 anymore. *IEEE robotics & automation magazine*, 10(2):12–18, 2003.
- [98] Kilian Kleeberger, Richard Bormann, Werner Kraus, and Marco F Huber. A survey on learning-based robotic grasping. *Current Robotics Reports*, 1:239–249, 2020.
- [99] Hikmet Kocabas. Gripper design with spherical parallelogram mechanism. *Journal of Mechanical Design*, 131(7), 2009.
- [100] Gerd Kortuem, Fahim Kawsar, Vasughi Sundramoorthy, and Daniel Fitton. Smart objects as building blocks for the internet of things. *IEEE internet computing*, 14(1):44–51, 2009.
- [101] Ioannis Kostavelis, Dimitrios Giakoumis, Georgia Peleka, Andreas Kargakos, Evangelos Skartados, Manolis Vasileiadis, and Dimitrios Tzovaras. Ramcip robot: A personal robotic assistant; demonstration of a complete framework. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018.

- [102] Robert Kovacs, Alexandra Ion, Pedro Lopes, Tim Oesterreich, Johannes Filter, Philipp Otto, Tobias Arndt, Nico Ring, Melvin Witte, Anton Synytsia, et al. Trussformer: 3d printing large kinetic structures. In *The 31st Annual ACM Symposium on User Interface Software and Technology*, pages 113–125. ACM, 2018.
- [103] Yuki Koyama, Shinjiro Sueda, Emma Steinhardt, Takeo Igarashi, Ariel Shamir, and Wojciech Matusik. Autoconnect: computational design of 3d-printable connectors. *ACM Transactions on Graphics (TOG)*, 34(6):1–11, 2015.
- [104] Yuki Koyama, Shinjiro Sueda, Emma Steinhardt, Cal Poly, and I D C Herzliya. Auto-Connect : Computational Design of 3D-Printable Connectors. *ACM Transactions on Graphics*, 2015.
- [105] Eric Krotkov, Douglas Hackett, Larry Jackel, Michael Perschbacher, James Pippine, Jesse Strauss, Gill Pratt, and Christopher Orłowski. The darpa robotics challenge finals: Results and perspectives. *Journal of Field Robotics*, 34(2):229–240, 2017.
- [106] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In *Proceedings of the IEEE international conference on computer vision*, pages 954–962, 2015.
- [107] Andrey Kurenkov. Deepcrop : Directed object segmentation with deep learning. In *arXiv*, 2016.
- [108] Maria Kyrarini, Fotios Lygerakis, Akilesh Rajavenkatanarayanan, Christos Sevastopoulos, Harish Ram Nambiappan, Kodur Krishna Chaitanya, Ashwin Ramesh Babu, Joanne Mathew, and Fillia Makedon. A survey of robots in healthcare. *Technologies*, 9(1):8, 2021.
- [109] TP-Link Laos. Smart wi-fi light switch. https://www.tp-link.com/la/products/details/cat-5622_HS200.html, 2019. (Accessed on 04/01/2019).
- [110] Gierad Laput, Karan Ahuja, Mayank Goel, and Chris Harrison. Ubicoustics: Plug-and-play acoustic activity recognition. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pages 213–224, 2018.
- [111] Gierad Laput, Yang Zhang, and Chris Harrison. Synthetic sensors: Towards general-purpose sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 3986–3999, 2017.
- [112] Kiju Lee, Yanzhou Wang, and Chuanqi Zheng. Twister hand: Underactuated robotic gripper inspired by origami twisted tower. *IEEE Transactions on Robotics*, 36(2):488–500, 2020.
- [113] Danny Leen, Nadya Peek, and Raf Ramakers. Lamifold: fabricating objects with integrated mechanisms using a laser cutter lamination workflow. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 304–316, 2020.

- [114] LEGO. Homes - Mindstorms LEGO.com, 2019.
- [115] Honghua Li, Ruizhen Hu, Ibraheem Alhashim, and Hao Zhang. Foldabilizing furniture. *ACM Trans. Graph.*, 34(4):90–1, 2015.
- [116] Jiahao Li, Meilin Cui, Jeeun Kim, and Xiang’Anthony’ Chen. 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*, pages 897–911, 2020.
- [117] Jiahao Li, Jeeun Kim, and Xiang ’Anthony’ Chen. 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, UIST 2019, New Orleans, LA, USA, October 20-23, 2019*, pages 673–685, 2019.
- [118] Jiahao Li, Jeeun Kim, and Xiang’Anthony’ Chen. 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*, pages 673–685, 2019.
- [119] Jiahao Li, Alexis Samoylov, Jeeun Kim, and Xiang ’Anthony’ Chen. Roman: Making everyday objects robotically manipulable with 3d-printable add-on mechanisms. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems, CHI ’22, New York, NY, USA, 2022*. Association for Computing Machinery.
- [120] Nianlong Li, Han-Jong Kim, LuYao Shen, Feng Tian, Teng Han, Xing-Dong Yang, and Tek-Jin Nam. Haplinkage: Prototyping haptic proxies for virtual hand tools using linkage mechanism. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 1261–1274, 2020.
- [121] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3706–3715, 2020.
- [122] Z-C Lia and C-H Menq. The dexterous workspace of simple manipulators. *IEEE journal on Robotics and Automation*, 4(1):99–103, 1988.
- [123] TV Lift. Motorized tv lift & tv automation system - nexus 21. <https://www.tvlift.com/>, 2019. (Accessed on 04/01/2019).
- [124] Chao Liu and Lauren Christopher. Depth map estimation from motion for 2d to 3d conversion. In *2012 IEEE International Conference on Electro/Information Technology*, pages 1–4, May 2012.
- [125] Chih-Hsing Liu, Ta-Lun Chen, Chen-Hua Chiu, Mao-Cheng Hsu, Yang Chen, Tzu-Yang Pai, Wei-Geng Peng, and Yen-Pin Chiang. Optimal design of a soft robotic gripper for grasping unknown objects. *Soft robotics*, 5(4):452–465, 2018.

- [126] Hangxin Liu, Chi Zhang, Yixin Zhu, Chenfanfu Jiang, and Song-Chun Zhu. Mirroring without overimitation: Learning functionally equivalent manipulation actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8025–8033, 2019.
- [127] Shaowei Liu, Hanwen Jiang, Jiarui Xu, Sifei Liu, and Xiaolong Wang. Semi-supervised 3d hand-object poses estimation with interactions in time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14687–14697, 2021.
- [128] Yuan Liu, Yilin Wen, Sida Peng, Cheng Lin, Xiaoxiao Long, Taku Komura, and Wenping Wang. Gen6d: Generalizable model-free 6-dof object pose estimation from rgb images. *arXiv preprint arXiv:2204.10776*, 2022.
- [129] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. *Robotics*, 1981.
- [130] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines, 2019.
- [131] Kalle Lyytinen and Youngjin Yoo. Ubiquitous computing. *Communications of the ACM*, 45(12):63–96, 2002.
- [132] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences Blair. *ACM Transactions on Graphics*, 2005.
- [133] Ajay Mandlekar, Jonathan Booher, Max Spero, Albert Tung, Anchit Gupta, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1048–1055. IEEE, 2019.
- [134] M. Manti, A. Pratesi, E. Falotico, M. Cianchetti, and C. Laschi. Soft assistive robot for personal care of elderly people. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 833–838. IEEE, jun 2016.
- [135] Mariangela Manti, Andrea Pratesi, Egidio Falotico, Matteo Cianchetti, and Cecilia Laschi. Soft assistive robot for personal care of elderly people. In *2016 6th IEEE international conference on biomedical robotics and biomechatronics (BioRob)*, pages 833–838. Ieee, 2016.
- [136] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpm: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv:1903.06684*, 2019.

- [137] Pat Marion, Peter R Florence, Lucas Manuelli, and Russ Tedrake. Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3235–3242. IEEE, 2018.
- [138] Justin Matejka, Ali Hashemi, Michael Glueck, Tovi Grossman, Erin Bradner, and George Fitzmaurice. Dream Lens. 2018.
- [139] Christoforos Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld, and Jean Oh. Core challenges of social robot navigation: A survey. *ACM Transactions on Human-Robot Interaction*, 12(3):1–39, 2023.
- [140] Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. Interactive design of 3d-printable robotic creatures. *ACM Transactions on Graphics (TOG)*, 34(6):1–9, 2015.
- [141] Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. Interactive design of 3D-printable robotic creatures. *ACM Transactions on Graphics*, 34(6):1–9, oct 2015.
- [142] Ankur Mehta, Nicola Bezzo, Peter Gebhard, Byoungkwon An, Vijay Kumar, Insup Lee, and Daniela Rus. A design environment for the rapid specification and fabrication of printable robots. In *Experimental Robotics*, pages 435–449. Springer, 2016.
- [143] Ankur Mehta, Joseph DelPreto, and Daniela Rus. Integrated codesign of printable robots. *Journal of Mechanisms and Robotics*, 7(2), 2015.
- [144] Ankur M Mehta, Joseph DelPreto, Kai Weng Wong, Scott Hamill, Hadas Kress-Gazit, and Daniela Rus. Robot creation from functional specifications. In *Robotics Research*, pages 631–648. Springer, 2018.
- [145] Vincent Mendez, Francesco Iberite, Solaiman Shokur, and Silvestro Micera. Current solutions and future trends for robotic prosthetic hands. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:595–627, 2021.
- [146] Kalaitzakis Michail, Brennan Cain, Sabrina Carroll, Ambrosi Anand, Whitehead Camden, and Vitzilaios Nikolaos. Fiducial markers for pose estimation. *Journal of Intelligent & Robotic Systems*, 101(4), 2021.
- [147] Justinas Mišeikis, Pietro Caroni, Patricia Duchamp, Alina Gasser, Rastislav Marko, Nelija Mišeikienė, Frederik Zwilling, Charles De Castelbajac, Lucas Eicher, Michael Früh, et al. Lio-a personal robot assistant for human-robot interaction and care applications. *IEEE Robotics and Automation Letters*, 5(4):5339–5346, 2020.
- [148] Paul Motzki, Frank Khelfa, Lukas Zimmer, Marvin Schmidt, and Stefan Seelecke. Design and validation of a reconfigurable robotic end-effector based on shape memory alloys. *IEEE/ASME Transactions on Mechatronics*, 24(1):293–303, 2019.

- [149] Stefanie Mueller, Alexander Teibrich, Stefan Neubert, Patrick Baudisch, François Guimbretière, and Robert Kovacs. Patching Physical Objects. 2015.
- [150] R. Mutlu, G. Alici, M. in het Panhuis, and G. Spinks. Effect of flexure hinge type on a 3d printed fully compliant prosthetic finger. In *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 790–795, 2015.
- [151] MySmartBlinds. Make your blinds smart. <https://www.mysmartblinds.com/>, 2019. (Accessed on 04/01/2019).
- [152] Ken Nakagaki, Joanne Leong, Jordan L Tappa, João Wilbert, and Hiroshi Ishii. 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*, pages 882–896, 2020.
- [153] Ken Nakagaki, Yingda Liu, Chloe Nelson-Arzuaga, and Hiroshi Ishii. Trans-dock: Expanding the interactivity of pin-based shape displays by docking mechanical transducers. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 131–142, 2020.
- [154] Kento Nakayama, Weiwei Wan, and Kensuke Harada. Designing grasping tools for robotic assembly based on shape analysis of parts. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 1–7. IEEE, 2019.
- [155] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. ProtoAR. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*, pages 1–12, New York, New York, USA, 2018. ACM Press.
- [156] Francesca Negrello, Werner Friedl, Giorgio Grioli, Manolo Garabini, Oliver Brock, Antonio Bicchi, Máximo A Roa, and Manuel G Catalano. Benchmarking hand and grasp resilience to dynamic loads. *IEEE Robotics and Automation Letters*, 5(2):1780–1787, 2020.
- [157] BBC News. Nissan’s self-parking robot chairs tidy up offices. <https://www.youtube.com/watch?v=FLEgvD7iG-M>, 2016. (Accessed on 04/04/2019).
- [158] RT News. Smart slippers: Nissan extends its self-parking technology. <https://www.youtube.com/watch?v=BtGqiOC2SSU>, 2018. (Accessed on 04/04/2019).
- [159] H. Nishimura, A. Kakogawa, and S. Ma. Development of an underactuated robot gripper capable of retracting motion. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2161–2166, 2012.
- [160] Illah Nourbakhsh, Rob Powers, and Stan Birchfield. Dervish an office-navigating robot. *AI magazine*, 16(2):53–53, 1995.
- [161] Makoto Okabe, Kenshi Takayama, Takashi Ijiri, and Takeo Igarashi. Light shower: a poor man’s light stage built with an off-the-shelf umbrella and projector. In *ACM SIGGRAPH 2007 sketches*, pages 62–es. 2007.

- [162] Allison M Okamura, Niels Smaby, and Mark R Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262. IEEE, 2000.
- [163] Kaveh Pahlavan, Prashant Krishnamurthy, and Yishuang Geng. Localization challenges for the emergence of the smart world. *IEEE Access*, 3:3058–3067, 2015.
- [164] Frank C Park and Bryan J Martin. Robot sensor calibration: solving $ax=xb$ on the euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, 1994.
- [165] Georgia Peleka, Ioannis Kostavelis, Andreas Kargakos, Dimitrios Giakoumis, Manolis Vasileiadis, Dimitrios Tzovaras, and Evangelos Skartados. RAMCIP Robot: A Personal Robotic Assistant; Demonstration of a Complete Framework. pages 96–111. Springer, Cham, sep 2019.
- [166] Huaishu Peng, Jimmy Briggs, Cheng-Yao Wang, Kevin Guo, Joseph Kider, Stefanie Mueller, Patrick Baudisch, and François Guimbretière. Roma: Interactive fabrication with augmented reality and a robotic 3d printer. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–12, 2018.
- [167] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019.
- [168] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.
- [169] Adam Pettinger, Conner Dimoush, and Mitch Pryor. Passive tool changer development for an elastic and compliant manipulator. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1200–1205. IEEE, 2019.
- [170] C Piazza, G Grioli, MG Catalano, and AJAROC Bicchi. A century of robotic hands. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:1–32, 2019.
- [171] Steffen Puhlmann, Jason Harris, and Oliver Brock. Rbo hand 3: A platform for soft dexterous manipulation. *IEEE Transactions on Robotics*, 38(6):3434–3449, 2022.
- [172] Xun Qian, Fengming He, Xiyun Hu, Tianyi Wang, and Karthik Ramani. Arnnotate: An augmented reality interface for collecting custom dataset of 3d hand-object interaction pose estimation. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, pages 1–14, 2022.
- [173] Raf Ramakers, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. RetroFab. pages 409–419, 2016.

- [174] Raf Ramakers, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 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*, pages 409–419, 2016.
- [175] Ahunzyanov Rasim and Tropchenko Alex. Hand detection based on skin color segmentation and classification of image local features, 1999.
- [176] Autodesk Research. Project dreamcatcher. <https://www.autodeskresearch.com/projects/dreamcatcher>, 2019. (Accessed on 04/01/2019).
- [177] Miguel Ribo, Axel Pinz, and Anton L Fuhrmann. A new optical tracking system for virtual and augmented reality applications. In *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No. 01CH 37188)*, volume 3, pages 1932–1936. IEEE, 2001.
- [178] Vincent Riquebourg, David Menga, David Durand, Bruno Marhic, Laurent Dela-hoche, and Christophe Loge. The smart home concept: our immediate future. In *2006 1st IEEE international conference on e-learning in industrial electronics*, pages 23–28. IEEE, 2006.
- [179] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.
- [180] Ali Roshanianfard and Noboru Noguchi. Pumpkin harvesting robotic end-effector. *Computers and Electronics in Agriculture*, 174:105503, 2020.
- [181] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut"- Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics – Proceedings of ACM SIGGRAPH 2004*, 2004.
- [182] Thijs Roumen, Ingo Apel, Jotaro Shigeyama, Abdullah Muhammad, and Patrick Baudisch. Kerf-canceling mechanisms: making laser-cut mechanisms operate across different laser cutters. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 293–303, 2020.
- [183] Thijs Jan Roumen, Willi Mueller, and Patrick Baudisch. Grafter: Remixing 3d-printed machines. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 63. ACM, 2018.
- [184] Iason Sarantopoulos, Yannis Koveos, and Zoe Doulgeri. Grasping Flat Objects by Exploiting Non-Convexity of the Object and Support Surface. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6. IEEE, may 2018.

- [185] Iason Sarantopoulos, Yannis Koveos, and Zoe Doulgeri. Grasping flat objects by exploiting non-convexity of the object and support surface. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5606–5611. IEEE, 2018.
- [186] Munehiko Sato, Ivan Poupyrev, and Chris Harrison. Touché: Enhancing touch interaction on humans, screens, liquids, and everyday objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '12*, pages 483–492, New York, NY, USA, 2012. ACM.
- [187] Ashutosh Saxena, Justin Driemeyer, Justin Kearns, and Andrew Ng. Robotic grasping of novel objects. *Advances in neural information processing systems*, 19, 2006.
- [188] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008.
- [189] Philipp Schoessler, Daniel Windham, Daniel Leithinger, Sean Follmer, and Hiroshi Ishii. Kinetic blocks: Actuated constructive assembly for interaction and display. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 341–349, 2015.
- [190] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.
- [191] Adriana Schulz, Cynthia Sung, Andrew Spielberg, Wei Zhao, Robin Cheng, Eitan Grinspun, Daniela Rus, and Wojciech Matusik. Interactive robogami: An end-to-end system for design of robots with ground locomotion. *The International Journal of Robotics Research*, 36(10):1131–1147, 2017.
- [192] Eric Schweikardt and Mark D. Gross. Learning about Complexity with Modular Robots. In *2008 Second IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, pages 116–123. IEEE, 2008.
- [193] Masahiro Shiomi, Takayuki Kanda, Dylan F Glas, Satoru Satake, Hiroshi Ishiguro, and Norihiro Hagita. Field trial of networked social robots in a shopping mall. In *2009 IEEE/RSJ international conference on intelligent robots and systems*, pages 2846–2853. IEEE, 2009.
- [194] Chen Song, Jiaru Song, and Qixing Huang. Hybridpose: 6d object pose estimation under hybrid representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 431–440, 2020.
- [195] Evan Strasnick, Jackie Yang, Kesler Tanner, Alex Olwal, and Sean Follmer. shiftio: Reconfigurable tactile elements for dynamic affordances and mobile interaction. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI '17*, pages 5075–5086, New York, NY, USA, 2017. ACM.

- [196] Yongzhi Su, Jason Rambach, Alain Pagani, and Didier Stricker. Synpo-net—accurate and fast cnn-based 6dof object pose estimation using synthetic training. *Sensors*, 21(1):300, 2021.
- [197] Yuta Sugiura, Daisuke Sakamoto, Anusha Withana, Masahiko Inami, and Takeo Igarashi. Cooking with robots: designing a household system working in open environments. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2427–2430, 2010.
- [198] Ryo Suzuki, Rubaiat Habib Kazi, Li-Yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. 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*, pages 166–181, 2020.
- [199] Haruki Takahashi and Jeeun Kim. 3d pen+ 3d printer: Exploring the role of humans and fabrication machines in creative making. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.
- [200] Takeshi Takaki and Toru Omata. 100g-100n finger joint with load-sensitive continuously variable transmission. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 976–981. IEEE, 2006.
- [201] Xiang Zhi Tan and Aaron Steinfeld. Using Robot Manipulation to Assist Navigation by People Who Are Blind or Low Vision. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction - HRI '17*, pages 379–380, New York, New York, USA, 2017. ACM Press.
- [202] Md Farhan Tasnim Oshim, Julian Killingback, Dave Follette, Huaishu Peng, and Tauhidur Rahman. Mechanobeat: Monitoring interactions with everyday objects using 3d printed harmonic oscillators and ultra-wideband radar. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 430–444, 2020.
- [203] Alexander Teibrich, Stefanie Mueller, François Guimbretière, Robert Kovacs, Stefan Neubert, and Patrick Baudisch. Patching physical objects. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 83–91, 2015.
- [204] Stefan Thalhammer, Dominik Bauer, Peter Hönig, Jean-Baptiste Weibel, José García-Rodríguez, and Markus Vincze. Challenges for monocular 6d object pose estimation in robotics. *arXiv preprint arXiv:2307.12172*, 2023.
- [205] Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. Computational design of linkage-based characters. *ACM Transactions on Graphics (TOG)*, 33(4):1–9, 2014.
- [206] Jun Ueda, Masahiro Kondo, and Tsukasa Ogasawara. The multifingered naist hand system for robot in-hand manipulation. *Mechanism and Machine Theory*, 45(2):224–238, 2010.

- [207] Jiří Ulrich, Ahmad Alsayed, Farshad Arvin, and Tomáš Krajník. Towards fast fiducial marker with full 6 dof pose estimation. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pages 723–730, 2022.
- [208] Francisca Gil Ureta, Chelsea Tymms, and Denis Zorin. Interactive modeling of mechanical objects. In *Computer Graphics Forum*, volume 35, pages 145–155. Wiley Online Library, 2016.
- [209] Markus Vincze, Tobias Koertner, Astrid Weiss, Walter Wohlkinger, Peter Einramhof, Paul Panek, Konstantinos Papoutsakis, Antonis Argyros, David Fischinger, Stefan Hofmann, and Peter Mayer. Hobbit, a care robot supporting independent living at home: First prototype and lessons learned. *Robotics and Autonomous Systems*, 75(PA):60–78, jan 2014.
- [210] Luke Vink, Viirj Kan, Ken Nakagaki, Daniel Leithinger, Sean Follmer, Philipp Schoessler, Amit Zoran, and Hiroshi Ishii. Transform as adaptive and dynamic furniture. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 183–183, New York, NY, USA, 2015. ACM.
- [211] Philipp Wacker, Simon Voelker, Adrian Wagner, and Jan Borchers. Physical Guides. In *Proceedings of the Symposium on Spatial User Interaction - SUI '18*, pages 25–35, New York, New York, USA, 2018. ACM Press.
- [212] Shaohua Wan and Sotirios Goudos. Faster r-cnn for multi-class fruit detection using a robotic vision system. *Computer Networks*, 168:107036, 2020.
- [213] Chen Wang, Daniel Freer, Jindong Liu, and Guang-Zhong Yang. Vision-based automatic control of a 5-fingered assistive robotic manipulator for activities of daily living. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 627–633. IEEE, 2019.
- [214] Weitian Wang, Rui Li, Yi Chen, Z. Max Diekel, and Yunyi Jia. Facilitating Human-Robot Collaborative Tasks by Teaching-Learning-Collaboration From Human Demonstrations, 2018.
- [215] Zeli Wang, Heng Li, and Xintao Yang. Vision-based robotic system for on-site construction and demolition waste sorting and recycling. *Journal of Building Engineering*, 32:101769, 2020.
- [216] Mark Weiser. The computer for the 21 st century. *Scientific american*, 265(3):94–105, 1991.
- [217] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

- [218] Jingren Xu, Weiwei Wan, Keisuke Koyama, Yukiyasu Domae, and Kensuke Harada. Selecting and designing grippers for an assembly task in a structured approach. *Advanced Robotics*, pages 1–17, 2021.
- [219] Zeyu Yan and Huaishu Peng. Fabhydro: Printing interactive hydraulic devices with an affordable sla 3d printer. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 298–311, 2021.
- [220] Hao Yang, Chen Shi, Yihong Chen, and Liwei Wang. Boosting 3d object detection via object-focused image fusion. *arXiv preprint arXiv:2207.10589*, 2022.
- [221] Xiaoying Yang and Yang Zhang. Cubesense: Wireless, battery-free interactivity through low-cost corner reflector mechanisms. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2021.
- [222] Jonny Yeu. Convert single-color stl files into multi-body models using meshmixer, 2019.
- [223] Christopher Yu, Keenan Crane, and Stelian Coros. Computational design of telescoping structures. *ACM Transactions on Graphics (TOG)*, 36(4):1–9, 2017.
- [224] Shenli Yuan, Austin D Epps, Jerome B Nowak, and J Kenneth Salisbury. Design of a roller-based dexterous hand for object grasping and within-hand manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8870–8876. IEEE, 2020.
- [225] Ye Yuan, Changxi Zheng, and Stelian Coros. Computational design of transformables. In *Computer Graphics Forum*, volume 37, pages 103–113. Wiley Online Library, 2018.
- [226] Azlan Zahid, Long He, Lihua Zeng, Daeun Choi, James Schupp, and Paul Heinemann. Development of a robotic end-effector for apple tree pruning. *Transactions of the ASABE*, 63(4):847–856, 2020.
- [227] Daohui Zhang, Xingang Zhao, Jianda Han, Xiaoguang Li, and Bi Zhang. Active modeling and control for shape memory alloy actuators. *IEEE Access*, 7:162549–162558, 2019.
- [228] Yang Zhang and Chris Harrison. Tomo: Wearable, low-cost electrical impedance tomography for hand gesture recognition. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 167–173, 2015.
- [229] Yang Zhang, Yasha Irvantchi, Haojian Jin, Swarun Kumar, and Chris Harrison. Sozu: Self-powered radio tags for building-scale activity sensing. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 973–985, 2019.
- [230] Qian Zhou, Sarah Sykes, Sidney Fels, and Kenrick Kin. Gripmarks: Using hand grips to transform in-hand objects into mixed reality input. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2020.

- [231] Yahan Zhou, Shinjiro Sueda, Wojciech Matusik, and Ariel Shamir. Boxelization: Folding 3d objects into boxes. *ACM Trans. Graph.*, 33(4), July 2014.
- [232] Jiawen Zhu, Zhenyu Chen, Zeqi Hao, Shijie Chang, Lu Zhang, Dong Wang, Huchuan Lu, Bin Luo, Jun-Yan He, Jin-Peng Lan, et al. Tracking anything in high quality. *arXiv preprint arXiv:2307.13974*, 2023.
- [233] Simon Zimmermann, Ghazal Hakimifard, Miguel Zamora, Roi Poranne, and Stelian Coros. A multi-level optimization framework for simultaneous grasping and motion planning. *IEEE Robotics and Automation Letters*, 5(2):2966–2972, 2020.
- [234] Douglas E Zongker, Dawn M Werner, Brian Curless, and David H Salesin. Environment matting and compositing. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 205–214, 1999.
- [235] Victor Zykov, Andrew Chan, and Hod Lipson. Molecubes: An open-source modular robotics kit. In *IROS-2007 Self-Reconfigurable Robotics Workshop*, pages 3–6, 2007.