

DESIGN AND ANALYSIS OF OPEN-SOURCE
EDUCATIONAL HAPTIC DEVICES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MECHANICAL
ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Melisa Orta Martinez

June 2020

© 2020 by Melisa Orta Martinez. All Rights Reserved.

Re-distributed by Stanford University under license with the author.



This work is licensed under a Creative Commons Attribution-Noncommercial 3.0 United States License.

<http://creativecommons.org/licenses/by-nc/3.0/us/>

This dissertation is online at: <http://purl.stanford.edu/vf931cp5479>

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Allison Okamura, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jo Boaler

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mark Cutkosky

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Sean Follmer

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Open-source hardware significantly impacts the development of technology by allowing communities of users with varied expertise to share, customize, and collectively improve on designs. Most haptic hardware available outside of the research community is proprietary and expensive. This prevents communities of users from easily obtaining, building, modifying, and learning from the devices. We propose that the ability to easily obtain, assemble and customize a haptic device is especially important for educational applications. In this work we present three open-source haptic devices, for which we carefully considered trade-offs in cost, ease of fabrication and assembly, and performance – toward making haptic devices more accessible for educational applications.

We first present Hapkit 3.0: an open-source, customizable, 3D-printed, one-degree-of-freedom (1-DOF), kinesthetic haptic device developed for science, engineering and math learning. The design of Hapkit 3.0 built on the design of two previous versions of the device (Hapkit 1.0 and Hapkit 2.0). We performed an analysis of the mechanical components of the Hapkit family of devices as well as a stiffness discrimination study using the three versions of Hapkit. From the study we found that Hapkit 3.0 outperformed the previous two versions. We applied the Hapkit family of devices in several educational environments: a middle school classroom, an online class, and undergraduate and graduate courses. We also investigated the use of Hapkit to illustrate abstract mathematical concepts for high school students, and found evidence that haptic feedback could be used to visualize mathematical functions.

We then present Haplink: an open-source, 3D-printed, kinesthetic haptic device that can be used as a 1- or 2-DOF device, and where the kinematics of the 2-DOF

device build on the kinematics of the 1-DOF device. Haplink was designed with the idea that students benefit from learning concepts incrementally, and that a device that itself increments from one to two degrees of freedom will aid in this process. We analyze the resolution and force capabilities of Haplink throughout its workspace and the effects of the resolution and force capabilities in rendering different virtual environments. We also describe the use of Haplink in a university freshman course on haptics, where we demonstrated the use of Haplink to teach concepts incrementally.

Finally we present HapCaps: an open-source, 3D-printed, tactile haptic device for finger sense training. HapCaps are haptic buttons designed to sense a press and give a tactile cue in the form of a vibration. We used 10 HapCaps to build a system (the HapCaps System) to combine finger sense training and math learning. Using the HapCaps System, we performed a four-week study in which first graders underwent finger sense training at the same time as they were learning math. The purpose of the study was to evaluate our device in a classroom environment as well as understand the logistics of using the HapCaps System hardware in a school setting. At the same time, we looked for any improvement in finger sense and math that would emerge from a short study. As a result of the study, we found evidence that the HapCaps System can be used to improve finger perception in first graders. We also improved the design based on the results of the study.

Acknowledgements

There have been a LOT of people who have helped me and made sure that I succeeded. As there is not enough space to list everything that you have done for me, I want to acknowledge some specific actions without which I would not have been able to obtain a PhD. Please realize that you all have done much more than listed below. In that vein, I would like to give thanks for the following (in chronological order, mostly).

First of all to my mom and dad, who have been slaves to my education, driving me to violin lessons, singing lessons, gymnastics, swimming, anything that they knew was available, they made sure I got to experience. They supported me financially and morally, and even went so far as to help build whatever device I was building at the time whenever they came to visit me at Stanford.

To my brother for inspiring me to come study in the US and believing in me.

To Lassie, Lucky, Chicharito, and Laila for making sure I stayed grounded and cared about the important things.

To the rest of my family, for all their support. My grandparents allowed me to live with them for a while so I could go to the college of my choosing, my uncle Stefan would hire me to give after-class lessons to my cousins whenever I needed money, my uncle Toño would take me tennis shoe-shopping whenever he judged my shoes were too old, my uncle Petercito always gave me great research advice, and my uncles and aunts Oscar, Klaus, Ale, Michelle, Palin and Elizabeth always gave me a fun sanctuary.

To my friends Diana, Lu, Ale, Sofi, Tico, Piojito, Tibu, Martin, Kafu, Angi, Pao, Vichi, and Lucy, for being so understanding and doing all of the heavy lifting and maintaining our friendship throughout these six long years.

To my friend Dani, for teaching me to program, getting me through undergrad, making sure we talked every day, being the keeper of memories, and for recommending great board games.

To Professors Alma Corral-Lopez and Jaime Gomez, for being excellent teachers and mentors.

To Paco Gutierrez, for making sure I was able to secure an internship in a German research lab.

To my friend Professor Antony Fraser-Smith, for welcoming me to Stanford, being a great mentor, making sure I ate well, teaching me magnetics, and telling great stories.

To my friends Behram, Kristen, Jon, Jeff, Reeve, Kim, Wade, Ceci, and Smootie thank you for all of your kindness, helping me navigate Stanford, doing homework together even though you clearly did not need to, and helping me study.

To my friend Suraj for telling great jokes, trying to teach me how to tell jokes, and teaching me writing and machine learning.

To my friend Alex Haas, for teaching me dynamics, optimization, analysis, and for being a great dog co-dad.

To Noe Lozano, for giving me the chance to be a mentor and course assistant for the ACE program and funding the first year of my PhD.

To my advisor Professor Allison Okamura, for giving me a chance to do a PhD, reading and correcting my awful writing, guiding me for the last 6 years, and making sure I succeeded.

To my friend Professor Heather Culbertson, for giving me great advice, always looking out for me, and giving me the opportunity to come to USC.

To my labmates Sean, Cara N., Giada, Sucho, Kyle, Julie, Mike, Laura, Cole, Margaret K., Margaret C., Sophia, Mine, Fabio, Ming, Cara W., Ilana, Nick, Kirk, and Nathan for all of their help with writing and presenting, collaborating on projects, brainstorming, building HapCaps, helping out with the HapCaps studies, and being great fun people.

To my collaborators Richard Davis, Paulo Blikstein, Karon Maclean, Oliver Schneider, and Cathy Williams, for giving me the opportunity to work with them and learn

from them.

To my rotators and interns Annalisa, Kaitlyn, Michal, Sean, Heather, Claus, and Bethany, for all of their hard work, inspiration, and the opportunity to be a mentor.

To Professors Sean Follmer and Mark Cutkosky, for agreeing to be on my committee, brainstorming, and giving me very helpful feedback on my projects and dissertation.

To Professor Jo Boaler, for giving me the opportunity to work with her, being a great collaborator, and including me in the amazing work she does at youcubed.

And finally to my husband Arni, thank you for all of your love and support. For staying up late with me, helping me with my grammar, checking my equations, brainstorming ideas, teaching me Solidworks, taking care of all of the housework while I was writing my thesis, building buttons for the HapCaps project, and giving up every weekend to just be there for me as I worked on the PhD.

To everyone thank you, I would never have been able to accomplish a PhD without you.

This work was supported in part by National Science Foundation grant 1441358, the 2019 Stanford HAI Seed Grant Program, and private donor support.

Contents

Abstract	iv
Acknowledgements	vi
1 Introduction	1
1.1 Motivation	1
1.2 Educational Haptics	2
1.3 Open-Source Haptic Devices	7
1.4 Contributions	8
1.5 Dissertation Overview	9
2 Hapkit 3.0	11
2.1 Introduction	12
2.2 Design Evolution of Hapkit	15
2.2.1 Design Goals	16
2.2.2 Structural Material	17
2.2.3 Motor	19
2.2.4 Transmission	20
2.2.5 Control Board	22
2.3 Analysis of Hapkit Mechanics	22
2.3.1 System Model	23
2.3.2 Frequency Domain System Identification	24
2.3.3 Other System Identification Approaches Tested with Hapkit 1.0	28
2.3.4 Cogging Analysis of Mabuchi Motor	33

2.4	Experiment: Stiffness Discrimination	
	Sensitivity using Hapkit	37
2.4.1	Methods	37
2.4.2	Data Analysis	38
2.4.3	Results	39
2.5	Use in the Classroom	41
2.5.1	Haptics Online Class	41
2.5.2	Haptics Workshop in a Middle School Classroom	46
2.5.3	Undergraduate Freshman Haptics Seminar: ME 20N	50
2.5.4	Graduate Haptics Class: ME 327	52
2.5.5	Teaching Trigonometry using Haptics	57
2.5.6	Discussion	63
2.6	Conclusions	64
3	Haplink	67
3.1	Introduction	68
3.2	Haplink Design	70
3.2.1	Design Goals	70
3.2.2	Haplink Design Implementation	72
3.3	Haplink Analysis	84
3.3.1	Forward Kinematics	84
3.3.2	Differential Kinematics	85
3.3.3	Haplink Workspace	86
3.3.4	Inverse Kinematics	88
3.3.5	Kinematic Conditioning	89
3.3.6	End-Effector Position Resolution	89
3.3.7	Drive Electromechanical System Characterization	90
3.3.8	Output Force Analysis	92
3.4	Software and Control	94
3.4.1	Startup Calibration	96
3.4.2	Rendering Virtual Environments	97

3.5	Haplink Use in the Classroom	100
3.6	Conclusions	108
4	HapCaps	109
4.1	Introduction	110
4.1.1	Motivation	110
4.1.2	Open-Source Tactile Haptic Devices and Tactile Haptic Devices used for Educational Applications	111
4.2	HapCap Design	112
4.2.1	HapCap Design Goals	114
4.2.2	HapCap Design Implementation	115
4.3	HapCaps System Design	120
4.3.1	Design Goals	121
4.3.2	HapCaps System Design Implementation	123
4.4	HapCaps System Control	127
4.5	HapCaps System Haptic Feedback	131
4.6	HapCaps System Use in the Classroom	133
4.6.1	Methods	135
4.6.2	Results and Discussion	141
4.7	Conclusion	146
5	Conclusion	147
5.1	Summary of Contributions	147
5.2	Future Work	149
5.2.1	Educational Software for Haptics	149
5.2.2	Studying the effects of haptic device customizations in educa- tional applications	150
5.2.3	Using Haptic Feedback to Represent Abstract Mathematical Concepts	151
5.2.4	Improving Study Design on the Role of Haptics for Educational Applications	152
5.2.5	Finger Sense Development and Math Learning	152

A	Education Evaluations	154
A.1	Math Evaluation	154
A.2	Finger Gnosis Evaluation	163
A.2.1	Procedure	163
A.2.2	Scoring Sheets	166
B	Math Activities	171
B.1	Identify Number	171
B.2	Identify Number Name on Screen	172
B.3	Identify Number in Ten Frame	172
B.4	Identify Number in Number Line	173
B.5	Single Digit Addition	173
B.6	Addition with Missing Second Number	174
B.7	Simple Subtraction	174
B.8	Simple Addition with Ten Frame	175
B.9	Addition with Missing Second Number with Ten Frame	175
B.10	Simple Addition with Number Line	176
B.11	Addition with Missing Second Number with Number Line	176
B.12	Number Flexibility	177
B.13	Number Flexibility with Two Numbers	177
B.14	Number Flexibility with Three Numbers	178
B.15	Counting up to a Number	178
B.16	Counting down from a Number	179
B.17	Which Group is Bigger	179
B.18	Which Number is Missing	180
B.19	What Numbers are Missing	180
B.20	How Many are There	181
B.21	How Many are Orange	181
B.22	What Comes Next	182
B.23	How Many Cubes in Cup	182
B.24	Pair of Cards Addition	183

B.25 How Many Steps High	183
B.26 Which Shape has More Sides	184
B.27 Which Shape has the Most Sides	184
B.28 How Many have More or Less than X Sides	185
Bibliography	186

List of Tables

1.1	Studies of the Role of Haptics in Education	3
1.2	Studies of the Role of Haptics in Education (Continued)	4
1.3	Studies of the Role of Haptics in Education (Continued)	5
2.1	Hapkit Design Evolution. (© 2019 IEEE.)	15
2.2	Frequency Analysis System Identification Results. (© 2019 IEEE.) .	26
2.3	Comparison of System Identification Results. (© 2019 IEEE.)	32
2.4	Modules of the Online Haptics Course	44
2.5	Modules of the Middle-School Haptics Workshop	48
2.6	Modules of the Freshman Haptics Seminar Course	53
2.7	Modules of the Graduate Haptics Course	56
3.1	Modules of the 2017 Freshman Haptics Seminar Course.	104

List of Figures

2.1	Hapkit 3.0.	11
2.2	Three versions of Hapkit.	13
2.3	Hapkit components.	14
2.4	Hapkit 3.0 Sector Pulley and Base features.	19
2.5	Hapkit 3.0's transmission assembly process.	21
2.6	Hapkit's free body diagram.	23
2.7	Bode plots of experimental data and fitted transfer function for Hapkits 1.0, 2.0, and 3.0 with added virtual spring.	27
2.8	Methods used for calculating the inertia of Hapkit's paddle.	28
2.9	Hapkit 1.0's underdamped response to a step input with an added virtual spring.	30
2.10	Cogging torque measurement setup for Mabuchi motor RF-370CA.	34
2.11	Cogging torque analysis results.	35
2.12	Desired and rendered forces differ due to cogging torque while rendering a spring of stiffness $k = 0.06$ N/mm. (© 2019 IEEE.)	36
2.13	Sigmoid function fits based on one user's responses during the stiffness discrimination test, for all three versions of Hapkit. (Adapted from [1] © 2019 IEEE.)	39
2.14	Weber fraction calculations along with the truncated mean and 95% confidence interval for Hapkits 1.0, 2.0, and 3.0.	40
2.15	A student in middle school workshop moved the Sector Pulley back and forth to feel a virtual spring with Hapkit 2.0. (© 2016 IEEE.)	49

2.16	Design of a haptic keyboard that uses three modified Hapkit 3.0 devices to provide feedback to three different keys. (© 2019 IEEE)	57
2.17	Trigonometry Explorer software.	60
3.1	Haplink	67
3.2	Haplink assembly process	71
3.3	Processor used by Hapkit and Haplink	73
3.4	Hapkit and Haplink position sensing solutions	75
3.5	Haplink's additional electronics	75
3.6	Haplink's structural components for both 1-DOF and 2-DOF assemblies	77
3.7	Hapkit's structural components compared to Haplink's in its 1-DOF version	78
3.8	Kinematic model of Haplink	81
3.9	Haplink mechanism rotational modes	82
3.10	Haplink footprint and workspace	88
3.11	Haplink Jacobian condition number.	90
3.12	Haplink position resolution throughout its workspace	91
3.13	Haplink torque output at different duty cycle values	92
3.14	Haplink force output throughout its workspace at a given handle location (r_x, r_y)	93
3.15	Haplink force output measured at three different locations in its workspace	95
3.16	Haplink startup calibration procedure	97
3.17	Three areas in Haplink's workspace where the rendering capabilities of the device are analyzed.	98
3.18	Rendering a virtual wall horizontally across Haplink's workspace . . .	99
3.19	Rendering an inside of a box and inside of a circle virtual environments in Area 2 of Haplink's workspace centered at $x = -50$ mm and $y = 100$ mm	101
3.20	Rendering an inside of a box and inside of a circle virtual environments in Area 3 of Haplink's workspace centered at $x = 0$ mm and $y = 100$ mm	102

3.21	Gear-shifter created by students in ME20N using Haplink	106
3.22	Operation game created by students in ME20N using Haplink	107
4.1	HapCaps System.	109
4.2	A HapCap shown next to a quarter for size comparison.	113
4.3	HapCap Elements.	115
4.4	HapCaps hardware components.	116
4.5	HapCap Button Pressed	118
4.6	HapCaps 3D printed structural components.	119
4.7	Student using a HapCaps System to perform math activities on a com- puter.	122
4.8	HapCaps System Components.	123
4.9	HapCaps System Button Enclosure components.	125
4.10	HapCaps System Circuit Enclosure components.	125
4.11	HapCaps Circuit electronic components	126
4.12	HapCaps System circuit diagram.	128
4.13	HapCap System Button-Press Flow Chart	129
4.14	HapCaps System Control Firmware Flowcharts	130
4.15	Haptic cue measured on one HapCap using an accelerometer without a finger on the HapCap.	133
4.16	Haptic cue measured on one HapCap using an accelerometer while a person is pressing the button.	134
4.17	HapCaps Workstation components	136
4.18	HapCaps Graphical User Interface.	138
4.19	Student participating in a finger sense test.	140
4.20	HapCaps Systems were used in a pilot study to develop finger percep- tion in first graders as they perform math activities.	142
4.21	Finger perception and math pre-and post-test results by student. . .	144
4.22	Finger perception and math improvement by group.	145

Chapter 1

Introduction

1.1 Motivation

Hands-on educational activities help students understand abstract concepts by connecting mathematical models to physical interactions in a manner that exploits students' intuition and experiences with the physical world [2]. A haptic device enables interactions through the sense of touch and provides a versatile, flexible, programmable interface capable of simulating a wide range of environments. Robotics is already a popular tool used in STEM education, and we propose that haptic devices have an even greater potential to impact STEM learning because they are inherently designed for interaction and display of information. Haptic devices can be used to teach courses where haptics or robotics is the topic of the course and where an understanding of the device functionality is important to learning, or used as an interactive media to understand concepts not inherently linked to understanding how the device functions. Haptic technology for education could also facilitate fundamental research related to hands-on learning, including cognitive embodiment, the role of physical experiences in understanding physics concepts, and the influence of hands-on labs on engagement and retention. In this chapter, we discuss haptic devices being used as an interactive media to display STEM concepts, and motivate the creation of open-source educational haptic devices. In Chapters 2, and 3 we discuss examples of haptic devices being used to teach haptics and robotics.

1.2 Educational Haptics

Haptic devices can be used to simulate touch-based interactions with physical objects, as well as abstract concepts such as mathematical functions, with the goal of enhancing STEM learning. Studies into the role of haptics as an interactive media to understand STEM concepts have revealed instances where the use of haptic simulations has positively impacted students' STEM learning [3, 4, 5, 6, 7, 8, 9, 10, 11, 12] as well as cases where it has had no or even a negative effect on learning [13, 14, 15, 16]. Table 1.1 presents a summary of these studies, with a focus on the reported findings of the effects of haptic feedback on learning, and the haptic devices used in each study. Zacharia [17] presents a similar analysis focusing on a comparison between the theoretical perspectives of embodied cognition and additional sensory channel, as well as a presentation of the empirical evidence presented by the educational studies that could support each theory, and does not focus on the haptic devices.

It is difficult to draw conclusions from the prior work on the effects of haptic feedback as an interactive media to understand STEM concepts due to the studies' small sample sizes, limited treatment time (i.e. most of the studies were performed as one-day interventions), and unrealistic learning environments (i.e. instead of being integrated into a class curriculum, learning happened as an intervention in front of a computer). An example of a pair of studies where it is difficult to draw definite conclusions as to the effects of haptics in learning are [7] and [16]. Both Hallman et al. [7] and Moore et al. [16] used a haptic joystick to understand the effects of using haptics in the learning of physics concepts. Hallman et al. [7] used a Microsoft Sidewinder Force Feedback 2 Joystick and Moore et al. [16] used a Logitech Force 3D Pro Haptic Joystick. Both studies were conducted as an intervention in which students were presented with a simulation of a physics concept on a computer. The students were then divided into two groups: haptics and no-haptics. The haptics group performed the activities in the intervention using the simulation enhanced with haptic feedback, and the students in the no-haptics condition performed the same activities using the same visual simulation but no haptic feedback. The exact duration of the intervention is not reported by the authors, but from the papers

Table 1.1: Studies of the Role of Haptics in Education

Study	Haptic Device	Participants and Groups	Methods	Evaluation	Conclusions
Brooks et al. [3]	GROPE-III system [18, 19, 20]	12 Expert biochemists. All received haptics and no haptics conditions.	Participants used a 6-DOF device to manipulate biomolecules with and without haptics.	Experiment performance and observations.	Haptic feedback aided in the understanding of molecular force fields.
Reiner [4]	IPO tactile trackball [21, 22]	12 undergraduate students. All received haptics.	Participants used a tactile trackball to explore force fields using haptic feedback. There was no visual feedback.	Interviews in which participants qualitatively described the virtual environment as they explored the different force fields. Drawings of representations of the different virtual environments.	Haptic feedback enabled students to identify and draw diagrams that represented force fields.
Hamza et al. [6]	Phantom Omni [23]	55 students from the 11th and 12th grades in high school. 23 received extra intervention with haptic feedback.	All of the students received instruction in Pascal's principles. Selected participants received an additional intervention using a haptics simulation of hydraulics.	Concept evaluation of Pascal's principle and interaction recordings.	The haptic simulator facilitated student understanding of the concepts and enhanced motivation.
Hallman et al. [7]	Microsoft Sidewinder Force Feedback 2 Joystick	28 graduate students divided into two groups: virtual simulation, and virtual simulation with haptics.	Participants used a virtual simulation of gears and forces to understand the concepts of how gears work. Half of the participants received haptic feedback.	Pre- and Post-test of physics concepts relating to gears, and motivation and usability questionnaires.	Haptic feedback enabled students to better understand the gear concepts. Haptic feedback had a negative impact on motivation.
Minogue & Jones [12]	Phantom Desktop [24]	80 middle school students divided into two equal groups: virtual simulation with and without haptics.	Students interacted with a virtual simulation of the cell membrane during passive transport with or without haptic feedback.	Pre- and post- concept evaluations of the animal cell structure.	Both groups improved the understanding of the concepts. Participants in the haptics group were better able to integrate concepts into a coherent whole and developed better understanding of membrane permeability.

Table 1.2: Studies of the Role of Haptics in Education (Continued)

Study	Haptic Device	Participants and Groups	Methods	Evaluation	Conclusions
Bivall et al. [8]	Phantom Omni [23]	20 post-graduate students divided into two equal groups experienced virtual simulations with and without haptic feedback.	Participants interacted with a virtual simulation of biomolecule interaction. Half of the participants received haptic feedback as forces when manipulating and docking the molecules.	Pre- and post-tests about the process of protein-ligand recognition. Qualitative analysis of students' reasoning during the learning phase.	Haptic feedback helped students learn more about the process of protein-ligand recognition and changed the way they reasoned about molecules to include more force-based explanations.
Han & Black [10]	Microsoft Sidewinder Force Feedback 2 Joystick	175 5th graders divided into three groups with the conditions: virtual simulation, virtual simulation plus kinesthetic haptic feedback, and virtual simulation plus kinesthetic and force feedback haptics.	Participants received 3 interventions once a week over 3 weeks. In the first intervention, participants received a pre-test on gear concepts, used a virtual simulation of gears, and then received a post-test on recall and inference of gears concepts. The second week, participants received a retention test on gears. The third week, participants were tested on transfer of concepts to the topic of incline planes.	Pre-and Post-Test on gears concepts. A retention evaluation on gears concepts. A post-test on incline planes.	Students who received kinesthetic and force feedback performed better on the post-test on gears as well as on the retention evaluation and the transfer test over the other two groups.
Jones et al. [11]	A Phantom Desktop [24], a Microsoft Sidewinder Force Feedback 2 Joystick, and a Mouse	36 middle and high-school students divided into three groups.	Participants interacted with a simulation of viruses in one of three conditions: no haptics, haptics with a Microsoft Sidewinder Force Feedback 2 Joystick, or haptics using a Phantom Desktop Device.	Participants were evaluated on their ability to describe the differences in viruses through a questionnaire.	Participants who used haptic feedback performed better than those who did not. Participants who used the higher fidelity haptics tool (Phantom Desktop) were capable of describing the viruses with more terms. Researchers concluded that haptics prompted the generation of more analogies and that the quality of the haptic feedback matters in outcomes.

Table 1.3: Studies of the Role of Haptics in Education (Continued)

Study	Haptic Device	Participants and Groups	Methods	Evaluation	Conclusions
Sato et al. [25]	SPIDAR-G: a haptic interface that is manipulated by a grip with 8 strings	19 undergraduate students divided into two groups	Participants received a lecture on Intermolecular Concepts. After the lecture, half of the participants were tested on their understanding and then interacted with a simulation using SPIDAR-G on inter-molecular forces. The other half first interacted with the simulation and then were tested.	Concept evaluation of intermolecular concepts and questionnaire of usability and engagement.	Students who used the simulation before testing scored much higher. Both groups gave positive feedback on the simulation with haptic feedback.
Jones et al. [13]	Phantom Premium [26] with a custom attachment to place the index finger of the user	50 highschool students divided into two conditions: virtual simulation with haptics and virtual simulation without haptics	Participants interacted with a virtual simulation of a virus with or without haptic feedback over one week by pushing, cutting and poking at it.	Pre- and post knowledge assessments on viruses, opinion questionnaires, and interviews.	During the week-long intervention, all participants improved their understanding of viruses. Haptic feedback condition had no effect on participants' scores.
Wiebe et al. [15]	Phantom Omni [23]	33 middle school students divided into two groups: haptics plus virtual simulation and virtual simulation	Participants interacted with a simulation of levers by manipulating fulcrum location, beam length and load placement with and without haptic feedback.	Pre- and post- conceptual tests on levers, an assessment of knowledge embedded in the simulation, eye tracking data to evaluate engagement, and video analysis of interaction with the simulation.	Haptic feedback increased the time that participants took to interact with the simulation. There was no statistical significant difference between the performance of both groups in the post-conceptual evaluation, however the visual only group scored higher in the embedded assessment.
Moore et al. [16]	Logitech Force 3D Pro haptic joystick [27]	51 undergraduate students enrolled in an advanced dynamics class divided into haptics and non-haptics group	Participants interacted with physics simulations of forces and inclined planes. Half of the participants received haptic feedback as part of the interaction and half did not.	A conceptual test embedded in the simulation of forces and inclined planes and a qualitative questionnaire measuring motivation and engagement.	Participants in the haptics condition performed significantly worse in conceptual evaluations than participants in the non-haptics condition. The questionnaire revealed that haptics could aid in motivation.

we infer that it was a one-session intervention of no more than a few hours. Both studies evaluated the effectiveness of the intervention in learning. Hallman et al. [7] used a pre-and post-test where students were asked questions about the concepts in the simulation and Moore et al. [16] incorporated factual and conceptual questions into the simulation itself. Both studies also included a qualitative questionnaire which evaluated motivation and used engineering students as participants. Hallman et al. [7] conducted the intervention using a virtual simulation of gears with 28 graduates students and Moore et al. [16] used a virtual simulation of inclined planes and forces with 51 undergraduate students.

Despite both studies using very similar devices and procedures, they came to different conclusions. Hallman et al. [7] concluded that haptics aided students in understanding the concepts but had a negative impact on motivation. Moore et al. [16] on the other hand, found that haptics had a negative effect on learning but a positive effect on motivation. There are possible explanations for the difference in these results. One theory is that haptics is more appropriate for teaching certain physics concepts. Another is that haptics is more effective for certain age groups. In addition, the results are likely population specific, so studies should be conducted with the target population and conclusions can only be made about that specific population.

A potential issue is that the haptic devices used in existing studies are not well suited for the classroom due to their high costs [23, 24, 26, 27], limited availability [18, 19, 20, 21, 22], and lack of customizability. Currently, applications used in these studies are tailored to the device rather than the device being customized to the application. Also, studies are conducted as interventions, instead of being integrated into the curriculum, since these devices are too expensive to be brought into a classroom. Research and education would benefit from devices that are inexpensive, openly accessible, and customizable for learning-specific applications. In the next section we motivate the creation of open-source educational haptic devices. Chapters 2, and 3 discuss haptic devices being used to teach concepts such as haptics and robotics where an understanding of the device functionality is important to learning.

1.3 Open-Source Haptic Devices

Open-source hardware impacts the development of technology by allowing communities of users with varied expertise to share, customize, and collectively improve on designs [28, 29]. The success of the open-source model is evidenced by the history of the first microcomputers, whose designs evolved through the collaborative efforts of the Homebrew Computing Club [29]. It was in this club that schematics and designs for the Apple I products were “passed around freely” [29] in the 1970s, and members helped each other in building their own systems. Open-source hardware projects are increasingly prevalent due to the availability of low-cost manufacturing methods such as 3D printing, free online Computer Aided Design (CAD) tools, open electronics platforms (e.g., Arduino [30]), and the growth of online communities enabled by the rapid expansion of internet access. Recent examples of open-source projects include the Kilobot Project [31], the soft robotics toolkit [32], the RepRap open 3D printer [33], and numerous other robotics projects (e.g., [34]).

Haptic devices have also been made open source [35, 36, 37]. Since these open-source haptic devices vary in type of haptic feedback and number of degrees of freedom we will discuss them in chapters 2, 3, and 4. Although open-source haptic device designs are being developed, most users (both inside and outside of the haptics research community) continue to rely on proprietary devices, such as the Phantom Omni [23], which prevents certain communities of users from easily modifying, sharing designs and evolving the design of the devices for new applications, limiting the potential impact of haptics in many arenas. One of the fields that we hypothesize could benefit from open-source haptic devices is education. We propose that the ability to easily obtain, assemble, and customize a haptic device is important for educational applications. Students can learn from the assembly process and likely feel a sense of ownership after customizing a device, which could motivate further learning [38]. Additionally, educators and researchers could benefit from a haptic device they can re-design to fit the needs of a particular application. Moreover, research into the role of haptics in education will benefit from the diverse community enabled by the open-source model in which users and researchers are free to share designs

and build-on each others' discoveries; this includes psychologists, education and haptics researchers, neuroscientists, teachers, and students. The challenge lies in making open-source hardware that is inexpensive and accessible to manufacture and build, while still maintaining the specifications necessary for high-fidelity haptic rendering (e.g., high bandwidth). In this thesis, we present the design of three open-source, 3D-printed, customizable haptic devices designed for educational applications and classroom use. We present an analysis of their rendering capabilities and assess their suitability for a classroom setting. Background into previous open-source educational haptic devices will be provided in each chapter of the thesis.

1.4 Contributions

The major contributions of this dissertation are:

- We present the design of Hapkit 3.0: an open-source, 3D printed, kinesthetic, one-degree-of-freedom (1-DOF), customizable haptic device designed for educational applications and classroom use. Hapkit was used in numerous educational environments, including an online class on haptics and undergraduate and graduate classes at Stanford University in dynamics, controls and haptics.
- We present a characterization of Hapkit dynamics across three versions of Hapkit in order to inform future design changes for Hapkit and other kinesthetic haptic devices.
- We introduce a novel application of haptic feedback in educational applications to substitute graphics in order to “visualize” mathematical functions.
- We present the design of Haplink: an open-source, 3D printed, customizable haptic device that can be used as a 1- and a 2- DOF device to teach science, math and engineering concepts incrementally. Haplink was used in an undergraduate freshman seminar on haptics where students learned kinematics and the rendering of virtual environments incrementally transitioning from 1 to 2 DOF.

- We present the design of a novel 2-DOF serial mechanism used in Haplink that allows for serial kinematics while permitting both motors to be grounded.
- We present the design of HapCaps, an open-source, 3D printed haptic button designed for finger sense training, as well as the design of HapCaps System, a device that uses ten HapCaps designed to combine finger sense training and math learning for first grade students. The HapCaps System was used in a pilot study in a first-grade classroom. During this pilot study, 19 first-graders used the HapCaps System to perform math activities 3 times a week over 4 weeks.
- We found evidence that a HapCaps System can be used to improve finger sense in first graders.

1.5 Dissertation Overview

In this chapter (**Chapter 1**), we presented the motivation for our research in the design of open-source haptic devices for education and reviewed past studies assessing the role of haptics as an interactive media to understand STEM concepts. We also presented the importance of open-source projects as a tool for technological advancement.

Chapter 2 discusses prior one-degree-of-freedom (1-DOF) kinesthetic haptic devices and presents the design of an open-source, 3D-printed, kinesthetic, 1-DOF, customizable haptic device designed for educational applications: Hapkit 3.0. This chapter also includes a characterization of the mechanical components of the Hapkit family of devices and compares the results to similar analyses performed on another 1-DOF open-source haptic device: the Stanford Haptic Paddle. We report findings from using the Hapkit family of devices in different educational settings: graduate classes, undergraduate seminars, a middle school classroom, and an online class. This chapter also discusses a design study with high school students to understand whether haptic feedback can be used as a tool to visualize abstract math concepts.

In **Chapter 3** we describe prior higher-degree-of-freedom open-source kinesthetic haptic devices and introduce Haplink: an open source, 3D printed, kinesthetic haptic

device that can be used as a 1- or 2-DOF device. We analyze the resolution and force capabilities of Haplink throughout its workspace and the effects of the resolution and force capabilities on rendering different virtual environments using Haplink. We also describe the use of Haplink in a freshman seminar on haptics.

Chapter 4 describes prior tactile devices for educational applications as well as the design of HapCaps: an open-source, 3D printed, tactile haptic device. We also present the design of HapCaps System: an open-source device that uses 10 HapCaps to combine finger sense training and math activities for first grade students. We also describe an analysis of the haptic feedback of a HapCaps System as well as a pilot study over four weeks in which students performed math activities and finger perception training simultaneously using HapCaps Systems.

Finally, **Chapter 5** summarizes the results of this work and reviews the contributions of this dissertation. It concludes by discussing future research projects that could derive from this work.

Chapter 2

Hapkit 3.0: Customizable 1-DOF Device



Figure 2.1: Hapkit 3.0: An open-source, 3-D printed, kinesthetic haptic device for educational applications. (Adapted from [1].)

Portions of this chapter are reprinted from [1] © 2019 IEEE, [39] © 2016 IEEE, and [40].

2.1 Introduction

One of the first open-source haptic devices designed specifically for education was the Haptic Paddle [36], which was developed at Stanford University in the mid-1990s for use in an undergraduate dynamic systems laboratory. The Haptic Paddle is a kinesthetic 1-DOF haptic device that uses a capstan drive for transmission. Many other groups have iterated on this design and used their own implementations in educational environments, typically undergraduate engineering courses [41, 42, 43, 44, 45, 46, 47, 48]. These educational haptic devices vary in materials, transmission method, actuation, sensing, user interface, microprocessor, cost, and performance. One notable change from the original Haptic Paddle implemented in many devices is the transition from a capstan drive to a friction drive for the transmission [42, 43, 44, 48], which sacrifices haptic performance for ease of assembly and maintenance.

In 2003, Gillespie et al. [49] introduced the iTouch: a 1-DOF haptic device that extends the educational capabilities of the Haptic Paddle by using a custom motor meant to be built by students from an arrangement of magnets and enameled wire. Gillespie et al. [49] also developed The Box. The Box is a kinesthetic haptic device that uses a wheel as the interface to the user and was developed for an embedded control course at the University of Michigan. In contrast to the iTouch, which was modeled after the Haptic Paddle and built out of inexpensive components, The Box uses a 1024 count per revolution encoder, a Maxon Re35 motor, and an embedded microprocessor, resulting in a cost of \$600 per device as reported by the authors.

Most of the devices discussed above were designed primarily to be used as laboratory equipment in courses, and used with dedicated and stationary computers, data acquisition boards, power supplies and motor amplifiers. Work has also aimed to make educational haptic devices more accessible and useful in a variety of learning environments. In 2002, Verplank et al. [50] designed The Plank: an open-source 1-DOF educational haptic device made from an old disk drive by using the disk drive

armature motors as the actuator. The Plank does not require a dedicated computer. It uses an ATmega163 microcontroller for processing instead of a data acquisition card, which significantly reduced the cost of the device. However, the device's accessibility is constrained by the use of external motor amplifiers and availability of old disk drives. In 2013, Morimoto et al. [48] developed Hapkit 1.0 [51] as a low-cost version of the Haptic Paddle to be used in a new online class on haptics for 100 students on the topic of haptics. Hapkit 1.0 was also open source and all the source files and code are available at <https://hapkit.stanford.edu/>. For this online class, Hapkit components were mailed to the students for them to assemble.

With the aim of enabling students and educators to manufacture Hapkit themselves, which was not possible with Hapkit 1.0 due to its laser-cut acrylic structural components and expensive motor, we took Hapkit 1.0 as a starting point, and evolved the design in terms of its structural materials, drive mechanism, and mechatronic components.

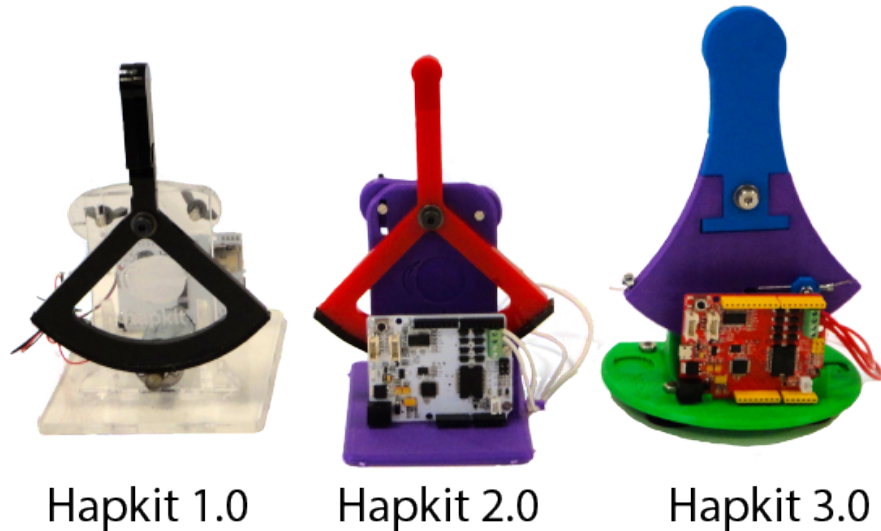


Figure 2.2: Three versions of low-cost, single-degree-of-freedom haptic devices for education: Hapkit 1.0 was designed in 2013 and uses a friction drive and laser-cut acrylic structural elements. Hapkit 2.0 was designed in 2014 and uses a friction drive and a combination of acrylic and 3-D printed plastic structural elements. Hapkit 3.0 was designed in 2015 and uses a capstan drive and 3-D printed plastic structural elements. (© 2019 IEEE.)

In this chapter, we present the design evolution and analysis of Hapkit [51]. The design of Hapkit evolved from lessons learned in several educational environments from a laser-cut device that uses a friction drive transmission (Hapkit 1.0 [48]) to a 3D-printed, friction-drive device that uses an inexpensive motor (Hapkit 2.0), and finally to a 3D-printed, customizable device that uses a capstan drive transmission (Hapkit 3.0 [39]). Figure 2.2 shows the three versions of the device. The remainder of this chapter is organized as follows: Section 2.2 discusses the evolution of the design and components of Hapkit. Section 2.3 presents an analysis and comparison of the mechanical components of the Hapkit family of devices and an analysis of the cogging torque of Hapkit’s motor, while Section 2.4 presents a stiffness discrimination study using the three devices. Section 2.5 discusses the use of Hapkit in a middle school classroom, undergraduate haptics seminar, graduate classes and an online class, focusing on the insights learned during the class use which informed the design evolution of Hapkit. We also present a pilot study in which we used Hapkit 3.0 to aid students in understanding the relationship between trigonometric functions and their graphs. Portions of this chapter were published in [39], [40], and [1].

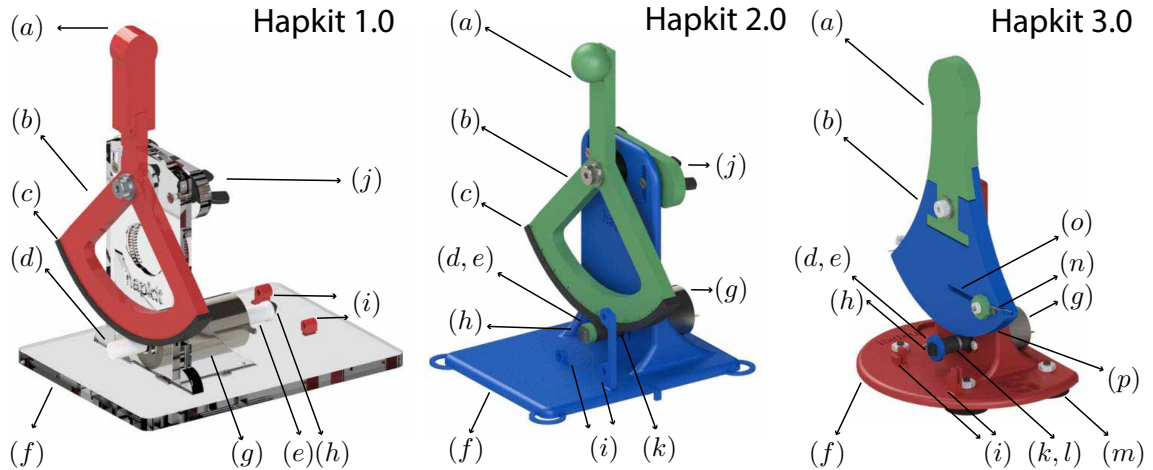


Figure 2.3: Components of Hapkit: (a) User Handle. (b) Sector Pulley. (c) Neo-prene strip. (d) Drive Wheel. (e) Magnet holder. (f) Base. (g) Motor. (h) Magnet. (i) Hapkit Board attachment pieces. (j) Height adjustment bar. (k) Neoprene tube. (l) Capstan cable. (m) Suction cups. (n) Cable routing feature. (o) Cable adjustment slot. (© 2019 IEEE.)

2.2 Design Evolution of Hapkit

The starting point for the Hapkit series of devices (Hapkits 1.0, 2.0, and 3.0, shown in Figure 2.3 with their components) was the Haptic Paddle [36]. As mentioned in the previous section, Hapkit 1.0 was created in 2013 by Morimoto et al. [48] at Stanford University for use in a new online class on haptics for 100 students. This class later evolved into a self-paced Massive Open Online Course (MOOC). Hapkit evolved in design through lessons learned at Stanford and in online courses, and perceptual and educational studies performed in collaboration with the Stanford School of Education and the University of British Columbia [39, 40, 51, 52]. Table 2.1 outlines the main features of Hapkit 1.0, Hapkit 2.0 and Hapkit 3.0 and compares them to the original Stanford Haptic Paddle. In the following sections, we describe our main goals when designing Hapkit, as well as how Hapkit evolved in terms of its structural material, transmission, and control board in order to meet those goals.

Table 2.1: Hapkit Design Evolution. (© 2019 IEEE.)

	Structural Material	Motor	Transmission	Control Board
Haptic Paddle [53]	acrylic	Maxon (surplus)	Capstan	data acquisition card
Hapkit 1.0	acrylic	Maxon (surplus)	Friction	Hapkit Board
Hapkit 2.0	3-D printer PLA and acrylic	Mabuchi (\$3.49)	Friction	Hapkit Board
Hapkit 3.0	3-D printer PLA	Mabuchi (\$3.49)	Capstan	Hapkit Board

2.2.1 Design Goals

Our design goals stem from manufacturing and assembly requirements, as well as dynamic and rendering requirements. When designing the Hapkit series of devices, one of our most important goals was that all devices be open source, meaning that the designs should be available and free to use and modify. We achieved this by placing all the designs, as well as assembly instructions, modifiable CAD files, parts lists, and instructional videos on our website: <http://hapkit.stanford.edu>.

Manufacturing and Assembly

Because Hapkit is meant to be used in educational applications, students and educators must be able to obtain and/or manufacture the parts themselves. Our Hapkit series of devices evolved in structural material and drive mechanism in order to become more accessible both in cost and assembly; the latest version of the device (Hapkit 3.0) costs around \$45 dollars to manufacture and we observed students in a graduate haptics class assemble it in under 10 minutes. The evolution of Hapkit from Hapkit 1.0 to Hapkit 3.0 is further described in Sections 2.2.2, 2.2.3, 2.2.4, and 2.2.5.

Hapkit is a “kit” to be assembled by the user. We propose that the assembly process gives students a better understanding of rendering algorithms and device kinematics. In order for Hapkit to be assembled by students, it is comprised of a minimal number of parts with an unambiguous (i.e. *poka-yoke* [54]) and robust assembly process. Finally, we aimed for Hapkit to be an “uncovered” device that could be used in various educational environments such as middle school and graduate classrooms, as well as in-home learning. Hapkit is designed such that students can see various parts of the device moving and interacting during use. We also ensured that the device had its own electronics board with limited power such that it was safe and no specialized laboratory equipment was needed for operation.

Kinematics and Rendering

Our aim was to create a kinesthetic, 1-DOF haptic device. A 1-DOF device limits cost and complexity, and is sufficiently versatile to simulate simple mathematics, physics, and dynamics concepts such as sinusoids, springs, and dampers. We chose to design

a kinesthetic (as opposed to tactile) device due to the integrated recording of motion and display of grounded forces enabled by kinesthetic haptic devices. Hapkit uses rotational motion because we have found in previous devices that it is acceptable to display nominally linear systems along the large-radius arc traveled by the handle. The alternative, linear motion, requires sliding mechanisms that usually result in high friction.

Hapkit is an impedance-type device such that users input position and the device outputs a force corresponding to a rendered impedance or other force-displacement relationship. This requires the device to be backdrivable and capable of providing users with a consistent feel throughout its workspace – indicating the need for a transmission with minimal friction, moving parts with low inertia, a motor with low cogging torque, and a loop rate of at least several hundred Hz.

2.2.2 Structural Material

Hapkit 1.0 was primarily made out of laser-cut acrylic pieces, with corresponding tabs and slots for connecting pieces. Two pieces, the Drive Wheel and a holder for the magnet used with the magnetoresistive sensor, were made with a ProJet 3-D printer (3-D Systems).

Hapkit 2.0 was designed to be more accessible to students by allowing them to manufacture or buy all the parts themselves. Few students have access to a laser cutter, whereas 3-D printers have become more readily available. Thus, the structural components were designed to be made on widely available, low-cost 3-D printers. Specifically, we chose the Makerbot Replicator 2 due to its price (\sim \$2000 in 2017), availability in public spaces and commercial venues, ease of use, and reliability. New challenges accompanied this transition to 3-D printing. In the first iteration of the new design, we faced several problems associated with the low strength and quality of the 3-D printed parts. Small pieces of the 3-D printed material used to screw the Hapkit Board to the base regularly broke off during the attachment process (when this happened, students used tape to hold the board down). The 3-D printed Drive Wheel (now made with a Makerbot 3-D printer using PLA as opposed to a ProJet

3-D printer using VisiJet white plastic) was not smooth enough to provide consistent contact in the friction drive transmission. Neoprene tubing was added to the drive wheel outer diameter to provide a smoother, more compliant interface surface (at the expense of some rolling resistance).

Most problematic was that the 3-D printed Sector Pulleys warped during printing, resulting in circular arcs with eccentricity greater than 1 mm. Because of this small eccentricity, the Sector Pulley was not able to maintain contact with the drive wheel throughout the workspace. We made Sector Pulleys with several low-cost 3-D printers (Makerbot Replicator 2, Flashforge Creator, and Afinia H480) to test if it was a problem specific to our unit, but none of the other printers in the same price category were able to print a sufficiently circular Sector Pulley. However, a Sector Pulley printed on a higher quality ProJet 3-D printer was able to create a smooth friction drive transmission, suggesting that printer capability was the issue. Thus, we released two different designs, with the options for the Sector Pulley to be laser cut or 3-D printed in Hapkit 2.0. Our recommendation was to laser cut the component since the material on the ProJet is very expensive. For our education studies and classroom use, we implemented the laser cut design. To align this thesis with that work, the rest of the analysis uses the properties of a laser cut paddle (where *paddle* refers to the combination of Sector Pulley and Handle).

Hapkit 3.0 was designed specifically for 3-D printing (Figure 2.4). All of the part designs were changed to have a more rounded design with fewer corners to reduce warping during printing. The Base was also made slimmer in order to reduce the amount of 3-D printing material and time needed to print. Suction cups were added to the Base in order to prevent it from slipping off of desks, a design addition inspired by work at Colorado School of Mines [55]. We also added features to the Base to aid in the assembly of Hapkit such as a Motor Stop and a bigger Motor Hole to prevent rubbing of the neoprene against the base while rendering virtual environments. The attachment of the Hapkit Board to the Base was changed to a snapping mechanism rather than screws, to eliminate the problem of the plastic breaking at the screw attachment point. The transmission was also changed from a friction drive to a capstan drive, which is more tolerant of inconsistencies in the 3-D printing process

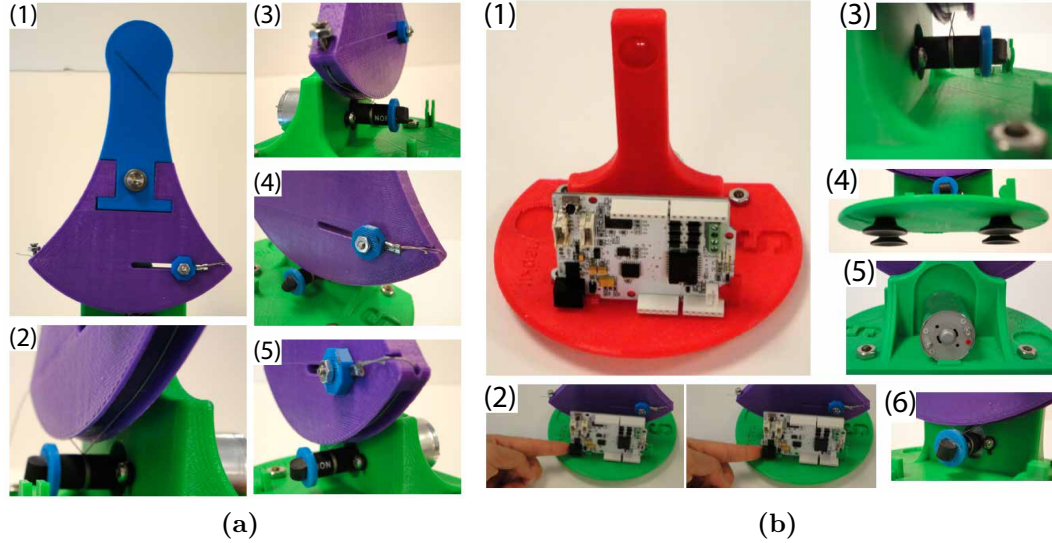


Figure 2.4: (a) Hapkit 3.0 Sector Pulley features: (1) Sector Pulley. (2) Cable-Guide. (3) Cable Attachment Screw. (4) Pre-Loading Feature. (5) Cable-Routing Features. (b) Hapkit 3.0 Base features: (1) Base. (2) Board Snap-Fit. (3) Drive-Wheel. (4) Suction Cups. (5) Motor Support. (6) Motor Shaft Hole. (© 2016 IEEE.)

(The transmission will be described more in depth in Section 2.2.4). These changes resulted in a Hapkit with structural components made entirely out of 3-D printed parts that could be printed on inexpensive 3-D printers. Hapkit 3.0 also allowed customization: it was designed in two parts such that the handle snaps into the driving mechanism of the Sector Pulley, allowing users to customize the handle while leaving the driving mechanism intact.

2.2.3 Motor

Hapkit 1.0 used a Maxon A-Max motor that was obtained as a surplus item at a very low cost, but with limited availability. This motor presents no cogging torque, which is desirable for haptic devices, but the cost of the motor when bought from the manufacturer makes it unattainable for educational applications. Hapkits 2.0 and 3.0 use a Mabuchi RF-370CA motor that presently can be purchased from Jameco for \$3.49. This motor does have significant cogging torque (analyzed in Section 2.3.4)

but was nevertheless successfully used as a driving actuator for Hapkits 2.0 and 3.0 as shown in Section 2.4.

2.2.4 Transmission

Inspired by updates to the original Haptic Paddle made at Vanderbilt University [43], Hapkit 1.0’s transmission was a friction drive. This change was made because tensioning of the cable in a capstan drive is a tedious and difficult procedure for novices.

Selecting the friction drive for ease of assembly (over the more robust capstan drive) worked for Hapkit 1.0 due to its robust structural material. However, Hapkit 2.0 was made out of 3-D printed structural components. With the high variability of part quality achieved with low-cost 3-D printers, the “feel” of Hapkit 2.0 was highly dependent on the print quality and how it was assembled. A friction drive depends on two elements maintaining contact without slipping. Both Hapkit 1.0 and Hapkit 2.0 have a height adjustment bar (Figure 2.3 item (j)) that allows the user to adjust the level of friction between the Sector Pulley and the Drive Wheel. Ideally, minimal compression of the neoprene strip is achieved to minimize rolling friction. However, putting together the transmission of Hapkit 2.0 required an expert in haptics who could adjust the height adjustment bar in order to attain a smooth transmission throughout its workspace, making Hapkit 2.0 too difficult for students to put together themselves.

For Hapkit 3.0, a robust capstan transmission was chosen. This is the approach taken by the original Haptic Paddle design as well as most high-end, multi-degree-of-freedom haptic devices. Thus, one of the main goals for the design of Hapkit 3.0 was creating a capstan drive that was easy to assemble by novices, as well as robust to avoid unwinding of the cable when the device becomes unstable due to programming error. In the original Haptic Paddle, the cable transmission is assembled by compressing the flexure on one side of the sector pulley, winding the cable around the drive wheel, tightening the screws to anchor the cable, then releasing the flexure to preload the cable. This process is difficult since it requires the user to assemble the transmission using only one hand, or use a clamp or binder clip to hold it closed.

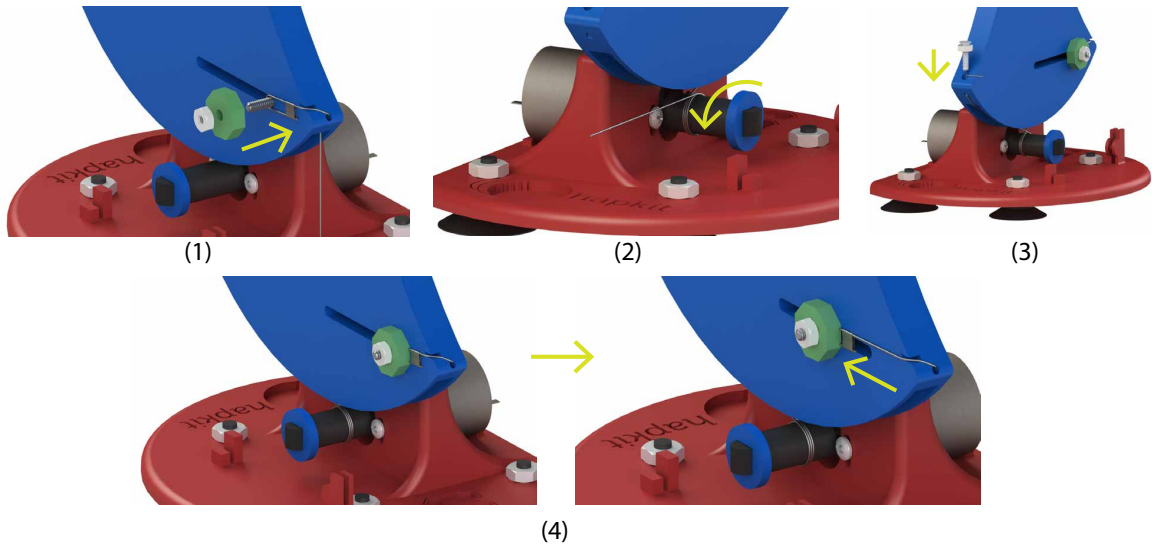


Figure 2.5: Hapkit 3.0 uses a capstan transmission with an easy-to-assemble design. This figure shows the steps to assemble Hapkit 3.0’s capstan transmission: (1) Crimp the cable and attach it to Hapkit using the Cable Routing Features. (2) Wind the capstan cable around the neoprene tube 2 times. The tube will compress preventing the cable from slipping. (3) Attach the cable to the other end using the Cable Attachment Screw. (4) Tension the cable by loosening the screw in the slot of the Pre-loading Feature and pushing it with your fingers as far as it will go. Tighten the screw again. (© 2019 IEEE.)

Hapkit 3.0 incorporates a slot with a fastener to help the user tension the wire. It also has cable routing features and a cable attachment screw, which keep the cable in place as the transmission is assembled (Figure 2.3 , item (n)). This allows the user to assemble the transmission using both hands and pre-load it as a final step using a pre-loading feature, which maintains the necessary cable tension. In order to prevent the cable from slipping off the end of the drive wheel, Hapkit 3.0 has a slot at the bottom of the Sector Pulley to guide the cable and a stop at the end of the Drive Wheel that is two times larger in diameter than the cable interface surface. The neoprene tube on the drive wheel, which has a Shore Durometer hardness of 60A, also compresses as the cable is tensioned, providing friction and preventing unwinding and slipping. Figure 2.5 shows the assembly process of the Hapkit 3.0 capstan drive transmission.

2.2.5 Control Board

One of the most significant changes between the original Haptic Paddle and Hapkit 1.0 is the development of a custom electronics board: Hapkit Board. This board was designed in partnership with Seeed Studio and is sold by Seeed on their website [56] at a cost lower than an Arduino combined with a motor driver. Hapkit Board is based on the Arduino Uno development board and uses the same ATmega328 microprocessor, but with added electronics: a motor driving circuit capable of driving two motors, a magnetoresistive sensor to be able to sense position, and an SD card reader capable of storing data. The board communicates to a computer and is programmed via USB and is compatible with the Arduino programming language, thus taking advantage of the robustness and versatility of Arduino. Hapkit board was originally designed for Hapkit 1.0 and adopted for Hapkits 2.0 and 3.0 because it allows Hapkit to be free from a computer, data acquisition boards, and motor amplifiers. With Hapkit board, Hapkit's control loop can run at a frequency greater than 1 kHz and can output a maximum force of 6N, while rendering 1-DOF virtual environments such as springs, dampers, walls, math functions, etc.

2.3 Analysis of Hapkit Mechanics

To quantify the differences between the three Hapkit versions and compare them to previous analyses of the Haptic Paddle, we use a linear model to describe the dynamics of the mechanical components (Section 2.3.1). This section describes our characterization of the three Hapkits.

In order to fit the parameters of this model, we used a frequency-based approach where we input individual torque sine waves (not chirps) and measured the output position (Section 2.3.2). We then validated our frequency-based method by analyzing Hapkit 1.0 using a first principles approach as well as a transient response approach, and compared these results to those obtained using the frequency-based method (Section 2.3.3). We also compared our results to those of [45,57] where the Stanford Haptic Paddle was analyzed using a time-based and a first principles approach.

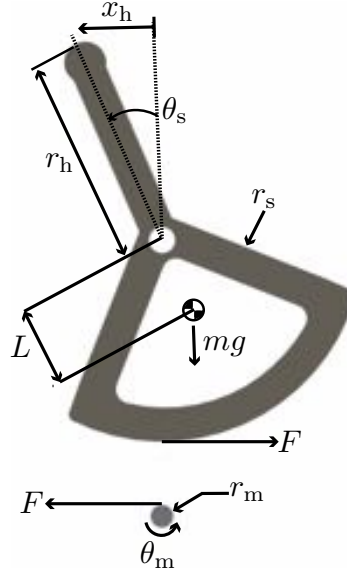


Figure 2.6: Free body diagram of Hapkit's paddle and motor pulley, with the kinematic variables used to describe them. Hapkit is modeled as two rigid bodies rolling in contact with each other without slipping, where x_h is the handle position, r_h , r_s , and r_m are the handle, sector, and motor pulley radii respectively, and θ_s and θ_m are the sector and motor pulley angles respectively. (© 2019 IEEE.)

Because Hapkits 2.0 and 3.0 used a Mabuchi motor that has cogging, we performed an analysis of this cogging torque to help us quantify its effects as a force to the user (Section 2.3.4).

2.3.1 System Model

We model Hapkit as two rigid bodies (the paddle and the motor with the drive wheel attached to it) rolling in contact with each other without slipping (Figure 2.6). The equation of motion is:

$$T_m = \left(I_m + \left(\frac{r_m}{r_s} \right)^2 I_s \right) \ddot{\theta}_m + \left(b_m + \left(\frac{r_m}{r_s} \right)^2 b_s \right) \dot{\theta}_m + \frac{r_m mg L}{r_s} \sin \left(\frac{r_m}{r_s} \theta_m \right) \quad (2.1)$$

T_m is the torque output from the motor, I_m is the motor inertia, r_m is the motor pulley radius, r_s is the sector pulley radius, I_s is the sector pulley inertia, b_m is the motor

damping coefficient, b_s is the sector pulley damping coefficient, L is the distance from the center of rotation of the sector pulley to its center of mass, m is the mass of the paddle, g is the gravitational acceleration, and θ_m is the angular position of the motor. Since r_m/r_s is a small number (~ 0.05), we make a small angle approximation to obtain the following equivalent system:

$$T_m = I_{eq}\ddot{\theta}_m + b_{eq}\dot{\theta}_m + k_{eq}\theta_m \quad (2.2)$$

where:

$$I_{eq} = I_m + \left(\frac{r_m}{r_s}\right)^2 I_s \quad (2.3)$$

$$b_{eq} = b_m + \left(\frac{r_m}{r_s}\right)^2 b_s \quad (2.4)$$

$$k_{eq} = \left(\frac{r_m}{r_s}\right)^2 mgL \quad (2.5)$$

2.3.2 Frequency Domain System Identification

In order to fit the parameters of our dynamic model, we performed a frequency domain analysis similar to the one performed in [58].

Experimental Hardware Setup

To model Hapkit's mechanical components, we substituted the Hapkit Board with a PCI Express Multifunction I/O Board Model 826 to interface with the computer, an Advanced Motion Controls AMC12A8 motor amplifier and an Avago Technologies HEDS-5540 encoder. The PCI board enables a loop rate of 300 microseconds, which allows us to sample the input and output waveforms at a frequency of 1 kHz.

The amplifier controls current; using the motor torque constant provided by the manufacturer, we effectively command a desired torque. We tested the torque output of our setup by suspending various weights from Hapkit, commanding the equivalent torque from the motor and observing that the weights stayed in the position in which

they were placed. To achieve more accurate and precise position measurements, we used an Avago Technologies HEDS-5540 encoder instead of the magnetoresistive sensor. To drive the device we used a Maxon RE DC 118743 motor which presents no cogging torque.

Experimental Procedure

For each type of Hapkit we input sinusoidal torques at frequencies between 0.1 and 100 rad/s, and measured the output position of the motor. We calculated the amplitude ratio and phase of the torque-to-position transfer function and fit the model parameters given in Equations (2.2) to (2.5) to the result. Prior work has shown that using low frequencies for system identification of haptic devices is challenging due to workspace limitations [58]. In order to overcome that challenge and be able to input a wide range of frequencies into our Hapkit at a constant amplitude, we added a virtual spring (stiffness) k_{spr} to our commanded torque:

$$T_m - k_{\text{spr}}\theta_m = I_{\text{eq}}\ddot{\theta}_m + b_{\text{eq}}\dot{\theta}_m + k_{\text{eq}}\theta_m \quad (2.6)$$

This results in:

$$T_m = I_{\text{eq}}\ddot{\theta}_m + b_{\text{eq}}\dot{\theta}_m + \tilde{k}_{\text{eq}}\theta_m \quad (2.7)$$

where:

$$\tilde{k}_{\text{eq}} = \left(\frac{r_m}{r_s}\right)^2 mgL + k_{\text{spr}} \quad (2.8)$$

The k_{spr} used in our experiments was 12×10^{-4} N m/rad. This was determined as the minimum stiffness required to maintain the sector pulley in the center of the workspace during the experiments. Once we obtained amplitude ratio and phase data from our experiment, we used a least squares fit approach to fit our second-order model.

Results of Frequency Domain System Identification

The resulting fit parameters are given in Table 2.2. Figure 2.7 shows the experimental data and model fit using the frequency domain system identification method.

Table 2.2: Frequency Analysis System Identification Results. (© 2019 IEEE.)

	Hapkit 1.0	Hapkit 2.0	Hapkit 3.0
$I_{\text{eq}} \text{ (kgm}^2\text{)}$	1.53×10^{-6}	1.82×10^{-6}	2.28×10^{-6}
$b_{\text{eq}} \left(\frac{\text{Nms}}{\text{rad}} \right)$	16.2×10^{-6}	26.6×10^{-6}	6.57×10^{-6}
$\tilde{k}_{\text{eq}} \left(\frac{\text{Nm}}{\text{rad}} \right)$	12.5×10^{-4}	15.9×10^{-4}	14.9×10^{-4}
$k_{\text{eq}} \left(\frac{\text{Nm}}{\text{rad}} \right)$	0.49×10^{-4}	3.8×10^{-4}	2.9×10^{-4}

Hapkits 1.0 and 2.0 have a much higher damping coefficient than Hapkit 3.0. This is expected due to the friction drive transmission in contrast to the capstan drive.

The inertias of the Hapkits are very similar for this model, with Hapkit 3.0's being slightly higher. The inertia of the system is a combination of the inertias of the motor and the paddle and since all the experiments used the same motor the difference in inertia can be attributed to the change in paddle design. Hapkit 3.0's paddle has slightly higher mass than the other two, and it is distributed mostly around the edges due to the 3-D printing process. This results in higher rotational inertia than a uniformly dense material like the acrylic in the Hapkit 1.0 and 2.0 paddles. Hapkit 3.0's paddle was printed by a Makerbot Replicator 3-D printer using 10% infill, which prints a thick, dense outer layer of material and a sparse honeycomb pattern in the rest of the body.

Our model assumes a linear system that does not capture any nonlinearities. Figure 2.7 shows the effects of those nonlinearities in the system as a phase-lag of the response. These effects are most likely due to Coulomb friction. The phase lag is very apparent on the Bode plots of Hapkits 1.0 and Hapkit 2.0, which use a friction drive, and much less so in Hapkit 3.0, which uses a capstan drive.

As described earlier, we used the same motor for all three Hapkits in this experiment, which is different from the motors used in practice. In practice, Hapkit 1.0 used a Maxon A-max 313735 motor with an inertia of $13.6 \times 10^{-7} \text{ kgm}^2$, which is slightly higher than the inertia of the motor used for the analysis. Hapkits 2.0 and 3.0 use a Mabuchi motor, which has an inertia of $8.3 \times 10^{-6} \text{ kgm}^2$ and also has significant

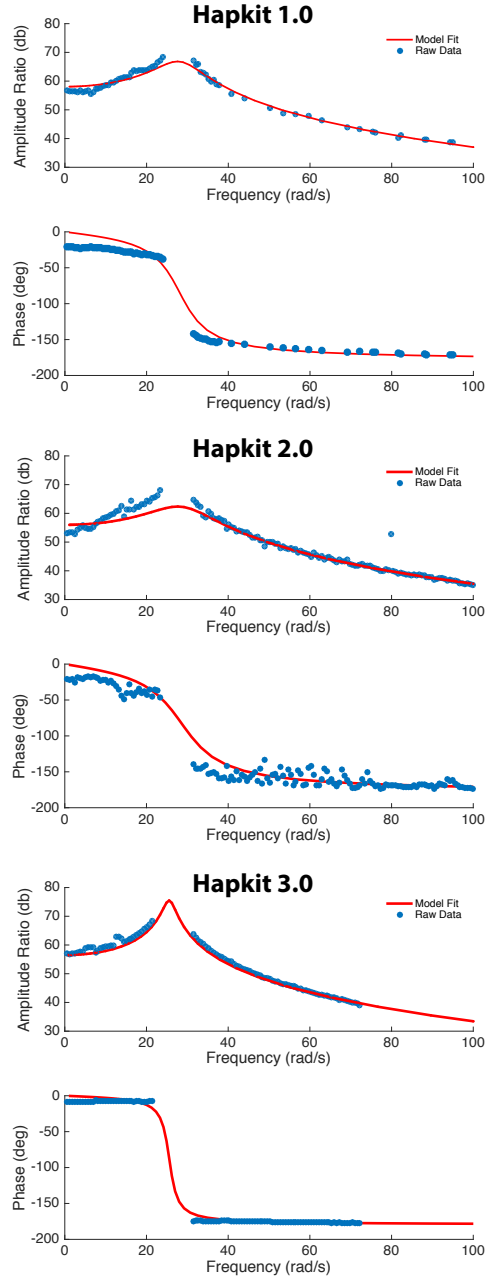


Figure 2.7: Bode plots of experimental data and fitted transfer function for Hapkits 1.0, 2.0, and 3.0 with added virtual spring. The Bode plot was obtained by exciting each Hapkit using sinusoidal waveforms of frequencies ranging from 0.1 to 100 rad/s and measuring the output position response. (© 2019 IEEE.)

cogging. Section 2.3.4 analyzes the effects of this cogging torque.

2.3.3 Other System Identification Approaches Tested with Hapkit 1.0

In order to validate our frequency domain system identification method, we performed a first principles and a transient response analysis on Hapkit 1.0 only and compared the results to those obtained for other haptic paddles in the literature.

First Principles System Identification

We calculated the inertia of Hapkit 1.0's paddle using two methods. First we performed a bifilar pendulum experiment (Figure 2.8(a)) as described in [45, 59]. This experiment obtains the inertia of the paddle about its center of mass ($8.1 \times 10^{-5} \text{ Nm}^2$). Using this value and the parallel axis theorem we obtain an inertia of $1.01 \times 10^{-4} \text{ Nm}^2$

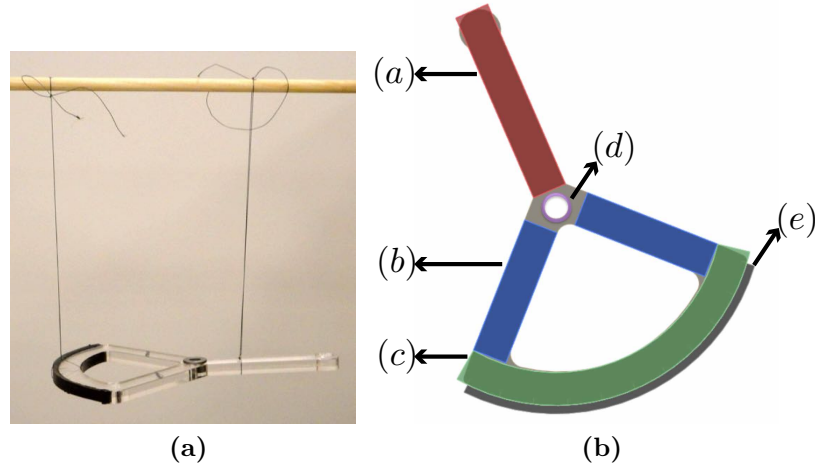


Figure 2.8: (a) We used a bifilar pendulum to measure the inertia of Hapkit 1.0's paddle. In this experiment, the paddle is suspended from two points and then its frequency of oscillation measured. This frequency of oscillation is correlated to the inertia of the paddle as described in [45, 59]. (b) We divided Hapkit 1.0's paddle into 5 main parts: (a) The Handle. (b) The Arc Supports. (c) The Arc. (d) Bearing. (e) Neoprene Strip. We calculated the moment of inertia about the center of rotation of the paddle.

of the paddle.

Second, we split the paddle into five parts (shown in Figure 2.8(b)), approximated each part to a basic geometric object (such as a cylinder for the bearing, a solid rectangular box for the handle, etc.), and directly calculated the moment of inertia about the center of rotation of the paddle. This calculation yielded an inertia of $1.49 \times 10^{-4} \text{ Nm}^2$. Both these approaches give an approximation of the inertia of the paddle of the same magnitude. The motor datasheet provides us with the inertia of the motor which, combined with the inertia of our paddle, gives us a moment of inertia about the axis of rotation of the motor (I_{eq}) of $1.53 \times 10^{-6} \text{ Nm}^2$.

In order to calculate k_{eq} , we again split the paddle into the five parts shown in Figure 2.8(b) and found the center of mass by taking the weighted average of the mass and location of each individual component. This then gives us the vector from the center of rotation to the center of mass: L . Using Equation 2.5, we obtained a k_{eq} of $2.5 \times 10^{-5} \text{ Nm/rad}$.

Transient Response Analysis

In order to analyze the transient response of Hapkit 1.0, we used the experimental setup described in Section 2.3.2. Instead of applying sinusoids as in our frequency domain experiment, we input a step waveform and measured position. Figure 2.9 shows the underdamped step response of Hapkit 1.0 with an added virtual spring of $k_{\text{spr}} = 40 \times 10^{-4} \text{ Nm/rad}$. This was determined as the minimum stiffness required to maintain the sector pulley in the workspace when a step input was applied. From this response we find the underdamped natural frequency ω_d by measuring the times at which the position achieves its maxima (t_1 and t_2). We obtain:

$$\omega_d = \frac{2\pi}{t_2 - t_1} = 51.15 \frac{\text{rad}}{\text{s}} \quad (2.9)$$

We also obtain the logarithmic decrement by measuring the ratio of the value of the response at two successive maxima (p_1 and p_2):

$$\Delta = \ln \left(\frac{p_1}{p_2} \right) = 0.3102 \quad (2.10)$$

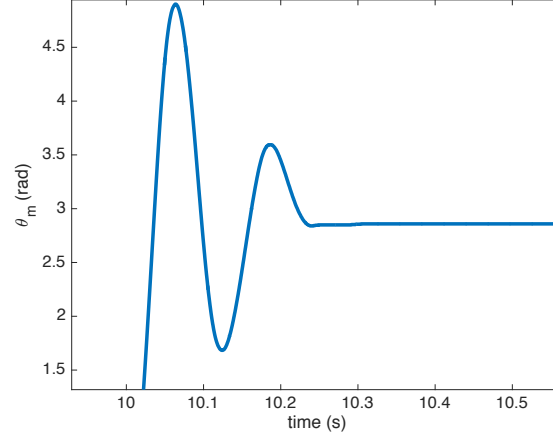


Figure 2.9: Hapkit 1.0's underdamped response to a step input with an added virtual spring of $k_{\text{spr}} = 40 \times 10^{-4}$ Nm/rad. This was measured using the hardware setup proposed in Section 2.3.2.

From these values and using the known \tilde{k}_{eq} value $(r_m/r_s)^2 mgL + k_{\text{spr}}$, we calculate the rest of the system parameters as follows:

$$\zeta = \frac{\Delta/(2\pi)}{\sqrt[2]{1 + (\Delta/(2\pi))^2}} = 0.0493 \quad (2.11)$$

$$\omega_n = \frac{\omega_d}{\sqrt[2]{1 - \zeta^2}} = 51.2 \frac{\text{rad}}{\text{s}} \quad (2.12)$$

$$I_{\text{eq}} = \frac{k}{\omega_n^2} = 1.53 \times 10^{-6} \text{ Nm}^2 \quad (2.13)$$

$$b_{\text{eq}} = 2\zeta\omega_n I_{\text{eq}} = 7.75 \times 10^{-6} \frac{\text{Nms}}{\text{rad}} \quad (2.14)$$

Results

In order to compare the results obtained from our analysis to those of other haptic paddles, we find the equivalent translational system of our rotational paddle as it would be felt by a user holding Hapkit at the top of the handle as it moves along an

arc. The equation that describes this linear motion is:

$$f_{\text{user}} = m_{\text{lin}}\ddot{x}_{\text{h}} + b_{\text{lin}}\dot{x}_{\text{h}} + k_{\text{lin}}x_{\text{h}} \quad (2.15)$$

where:

$$f_{\text{user}} = T_{\text{m}} \left(\frac{r_{\text{s}}}{r_{\text{h}}r_{\text{m}}} \right) \quad (2.16)$$

$$m_{\text{lin}} = I_{\text{eq}} \left(\frac{r_{\text{s}}}{r_{\text{h}}r_{\text{m}}} \right)^2 \quad (2.17)$$

$$b_{\text{lin}} = b_{\text{eq}} \left(\frac{r_{\text{s}}}{r_{\text{h}}r_{\text{m}}} \right)^2 \quad (2.18)$$

$$k_{\text{lin}} = k_{\text{eq}} \left(\frac{r_{\text{s}}}{r_{\text{h}}r_{\text{m}}} \right)^2 \quad (2.19)$$

$$x_{\text{h}} = \theta_{\text{m}} \left(\frac{r_{\text{h}}r_{\text{m}}}{r_{\text{s}}} \right) \quad (2.20)$$

Table 2.3 summarizes the results obtained from our system characterization of Hapkit 1.0 and comparison to [45] and [57], which performed similar analyses for the Haptic Paddle. We did not compare our values to [58] because, as stated by the authors, their values are overestimated and are not accurate representations of the physical parameters.

Both [45] and [57] perform a system identification of the Stanford capstan-driven Haptic Paddle. In [45], a first principles analysis is used; a bifilar pendulum experiment to calculate the inertia and direct measurements approximate the paddle's parameters.

Table 2.3 shows that our first principles analysis found a very similar inertia for the paddle (I_{s}) as the one obtained in [45], which is expected since they are both acrylic paddles of similar dimensions. In [57] a user applied a wide range of forces on the paddle handle, which were measured by a force sensor. The force applied, the position of the paddle, and time were recorded and a linear model was fit to the data.

Table 2.3 shows that our frequency-based analysis found similar inertia results to our first principles analysis and our transient response analysis. Our I_{eq} is also of the

Table 2.3: Comparison of System Identification Results. (© 2019 IEEE.)

Paddle	Hapkit 1.0	Hapkit 1.0	Hapkit 1.0	Johns Hopkins [57]	Stanford [45]
Method	Frequency Analysis	First Principles	Transient Response Analysis	Time-domain Analysis	First Principles
k_{eq} (Nm/rad)	4.94×10^{-5}	2.5×10^{-5}	2.5×10^{-5} *	- +	5.55×10^{-5}
I_m (kgm ²) °	10.8×10^{-7}	10.8×10^{-7}	10.8×10^{-7}	10.1×10^{-7}	10.5×10^{-7}
I_s (kgm ²)	1.486×10^{-4}	1.468×10^{-4}	1.489×10^{-4}	1.138×10^{-5}	3.156×10^{-4}
I_{eq} (kgm ²)	1.528×10^{-6}	1.534×10^{-6}	1.535×10^{-6}	1.036×10^{-6}	1.716×10^{-6}
b_{eq} (Nms/rad)	1.62×10^{-5}	- ¹	7.75×10^{-6}	1.213×10^{-5}	8.55×10^{-6}
m_{lim} (kg)	0.0823	0.0825	0.0826	0.035	0.058
b_{lim} (Ns/m)	0.872	- ¹	0.417	0.41	0.289
k_{lim} (N/m)	2.6595	1.3455	1.3455 *	- +	1.877

* For the transient response analysis, we use the value of k_{eq} calculated in the first principles analysis.

+ In [57] the value of k_{eq} is assumed to be negligible

° The values of I_m are obtained from the manufacturers' datasheets

¹ The values of b_{eq} and b_{lim} were not calculated using the first principles method for Hapkit 1.0.

same order of magnitude as that in [57].

Our frequency-based model underestimates the resonant peak of the response because it estimates a higher damping coefficient than our transient response analysis. This is due to the inability to collect frequency-response data at the resonant frequency. The transient response-based model does not have this issue and estimates a lower damping coefficient, but it still assumes a linear model.

We translate our rotational parameters into translational ones that are related to the force felt by the user. Table 2.3 shows that even though our results are very similar to [45] and [57] in rotational parameters for Hapkit 1.0 and the Haptic Paddle, the translational equivalents reveal more mass and damping for Hapkit 1.0. This is also expected because the moment arm for Hapkit is much shorter for Hapkit than for the Haptic Paddle. The point of application of the force to the center of rotation is 115 mm for the Haptic Paddle and 70 mm for Hapkit, which results in a moment arm 1.6 times longer for the Haptic Paddle.

The data in Table 2.3 demonstrate the effects of design changes on the dynamics of different versions of Hapkit in comparison with the original Haptic Paddle. These changes can have an effect on perception of virtual environments rendered with the device, as will be discussed in Section 2.4.

2.3.4 Cogging Analysis of Mabuchi Motor

The relatively inexpensive Mabuchi motor chosen for Hapkits 2.0 and 3.0 has cogging torque due to the interaction of the teeth of the iron core with the poles of the permanent magnet. As the motor rotates, the magnetization of the teeth is passed from one tooth to the next, giving it 6 preferential positions. The Mabuchi RF-370CA-15370 has 2 poles and 3 teeth, giving it six preferential positions. In contrast, the Maxon A-max 313735 motor used by Hapkit 1.0 has no iron core and as such, no soft magnetic teeth to interact with the permanent magnet. In order to understand the differences in performance due to the cogging torque present in Hapkits 2.0 and 3.0 but not in Hapkit 1.0, we measured the cogging torque in the Mabuchi motor.

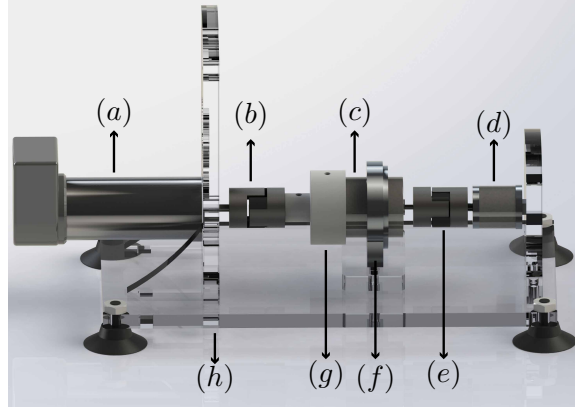


Figure 2.10: Cogging torque measurement setup for Mabuchi motor RF-370CA. A Maxon RE DC 118743 motor is powered and drives an unpowered Mabuchi RF-370CA motor which is connected to a Nano-17 force/torque sensor. The components of the setup are: (a) Maxon RE DC 118743 motor with HEDS-5540 encoder, (b) flexible shaft coupler, (c) Mabuchi RF-370CA motor, (d) Nano-17 force/torque sensor, (e) flexible shaft coupler, (f) supporting bearing, (g) coupler, and (h) base. (© 2019 IEEE.)

Experimental Setup

In order to analyze the cogging torque of the Mabuchi motor, we created a setup based on that proposed in [60], in which we spin the housing of an unpowered Mabuchi motor using a Maxon motor and measure the torque output on the rotor of the Mabuchi motor using a force/torque sensor.

Our setup (shown in Figure 2.10) consists of a Maxon RE DC 118743 motor (which has no cogging torque) with an HEDS-5540 Avago Technologies encoder. The Maxon motor is driven using an AMC12A8 motor amplifier whose output is controlled using a PC that is connected through a PCI Express Multifunction I/O board to the encoder and the motor amplifier (like our setup described in Section 2.3.2).

The shaft of the Maxon motor is connected to the housing of the Mabuchi motor using a 3-D printed coupler. The weight of the Mabuchi motor is supported by a bearing that allows it to spin freely without wobbling. Then we connect the shaft of the Mabuchi motor to a Nano-17 force/torque sensor and measure the output of the force/torque sensor using a PCI Express Multifunction I/O Board Model 826. This setup allows us to control the velocity of the Maxon motor and measure the torque

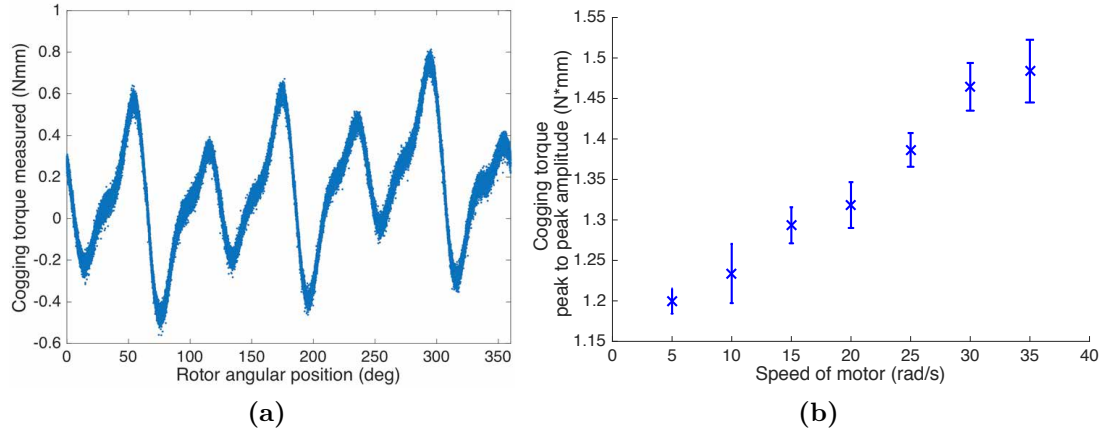


Figure 2.11: (a) Cogging torque measured for Mabuchi motor RF-370CA. The motor was rotated at a constant speed of 6 rad/s and data was collected over 2 min. (b) Mean and standard deviation of peak to peak cogging torque measured for Mabuchi motor RF-370CA measured at constant speeds of 5, 10, 15, 20, 25, 30, and 35 rad/s. (© 2019 IEEE.)

output on the rotor of Mabuchi motor.

Results

Figure 2.11(a) shows data collected during one trial of the cogging torque experiment in which we controlled the Maxon motor to rotate at a constant speed of 6 rad/s and measured the torque output of the Mabuchi rotor for 2 minutes. Figure 2.11(a) shows the expected 6 peaks of cogging torque output.

Figure 2.11(b) shows the mean and the standard deviation of the peak-to-peak amplitude of the measured torque on the Mabuchi at constant velocities of 5, 10, 15, 20, 25, 30 and 35 rad/s. Each experiment at a single speed lasted 2 minutes and was repeated 10 times to obtain the mean and standard deviation values. The relationship between speed and torque is linear rather than constant, likely due to effects of friction and eddy currents. When spinning the unpowered motor, the measured torque will be given by the relationship:

$$T_z = T_c + f(\omega) + T(\omega). \quad (2.21)$$

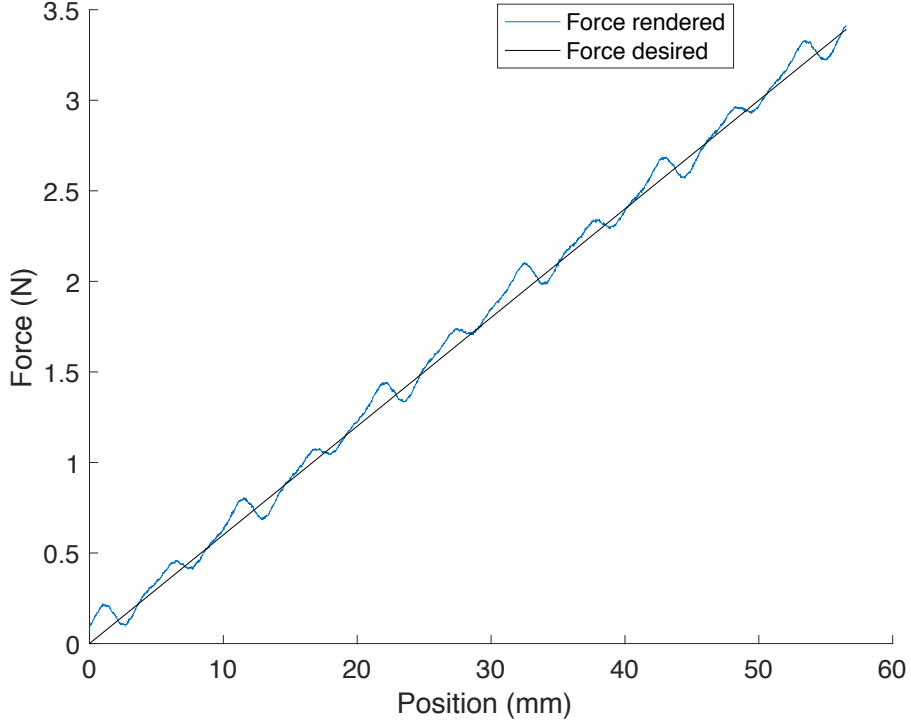


Figure 2.12: Desired and rendered forces differ due to cogging torque while rendering a spring of stiffness $k = 0.06$ N/mm. (© 2019 IEEE.)

T_z is the torque measured by the Torque sensor at the shaft of the motor. T_c is the pure cogging torque due to the interactions between the rotor and the poles of the magnet. $f(\omega)$ is the torque due to friction, which is a function of the rotational speed of the motor (ω) and $T(\omega)$ is the torque due to the eddy currents produced in the rotor of the motor as a result of the changing magnetic field due to the rotation of the motor – also a function of ω . Dissecting the sources of cogging torque is beyond the scope of this work.

Users typically move the handle of Hapkit with a hand speed between 2 and 5 cm/s, which results in motor rotation speeds between 10 and 30 rad/s and peak-to-peak measured torque between 1.2 and 1.5 Nmm. This torque is equivalent to a force of 0.3 N at the handle, which is noticeable by a user. Figure 2.12 shows the force felt at the handle by a user when Hapkit is rendering a virtual spring of stiffness $k = 0.06$ N/mm. This leads us to the question: How does the perceivable

cogging torque affect human perception of haptically rendered physical properties on the capstan-driven Hapkit 3.0, compared to the other Hapkit designs, which use a friction drive? In the next section (Section 2.4), we describe a stiffness discrimination test we performed using the three versions of Hapkit, in order to further understand the differences between them as well as determine whether this cogging torque had negative effects on user perception of virtual environments.

2.4 Experiment: Stiffness Discrimination Sensitivity using Hapkit

The purpose of this experiment was to determine participants' ability to discriminate between virtual springs of different stiffness using the three versions of Hapkit (1.0, 2.0, and 3.0), where the differences in mechanical properties described in Section 2.3 are hypothesized to affect user perception of rendered spring stiffness. We elected to test spring/stiffness discrimination due to its ubiquity as a method for evaluating haptic device and human perception performance. A total of 14 people participated in the experiment. Eight of the participants were male and six were female. The age of the participants ranged from 24 to 31 years of age. Six participants reported having no experience using Hapkits 1.0, 2.0 or 3.0. Three participants reported having experience using Hapkit 1.0, one participant reported having experience using Hapkit 2.0, and one participant reported having experience using Hapkit 3.0. One participant reported having experience using Hapkits 1.0 and 2.0 and two participants reported having experience using all Hapkits. All participants gave informed consent, and the protocol was approved by the Stanford University Institutional Review Board.

2.4.1 Methods

We performed a stiffness discrimination study using each of the three Hapkits. The study was divided into three phases. Participants sat in front of a computer and the three Hapkits: Hapkit 1.0, Hapkit 2.0 and Hapkit 3.0. A computer program then pseudorandomly chose the order of the test and instructed the user which Hapkit

should be used for each of the three phases. The participant used only one Hapkit for the entirety of that phase.

We implemented a two-alternative forced-choice paradigm using the method of constant stimuli [61]. In each phase of the experiment, participants attempted to distinguish between virtual springs of varying stiffness using force stimuli provided by the Hapkits. The reference stiffness was 60 N/m and the comparison stiffnesses were 35, 40, 45, 50, 55, 65, 70, 75, 80, and 85 N/m. Each spring comparison was repeated 10 times for each Hapkit, in a pseudo-random order. Participants were allowed to interact with each spring pair for as long as they desired. In order to progress to the next spring pair, they had to decide which spring was stiffer. After a participant made 100 comparisons, they took a break and then proceeded to the next phase of the experiment in which they would use another Hapkit, again chosen pseudorandomly. This was repeated until the participant completed the experiment with all three Hapkits.

After the experiment was completed, participants were asked to rank the level of difficulty completing the task with each Hapkit from 1 to 5, where 1 was considered very easy and 5 was very hard.

2.4.2 Data Analysis

We fit a psychometric function to the data of each participant for each Hapkit using MATLAB's `sigm_fit` function (Figure 2.13). The 0.5 threshold value determined the point of subjective equality (PSE) for each participant, and the 0.75 and 0.25 threshold values were used to calculate the Just Noticeable Difference (JND). The Weber fraction (WF), which determines participants' ability to discriminate between different stimuli in proportion to the reference stimulus, was calculated for each participant as:

$$\text{WF} = \frac{\text{JND}}{k_{\text{ref}}} * 100\% \quad (2.22)$$

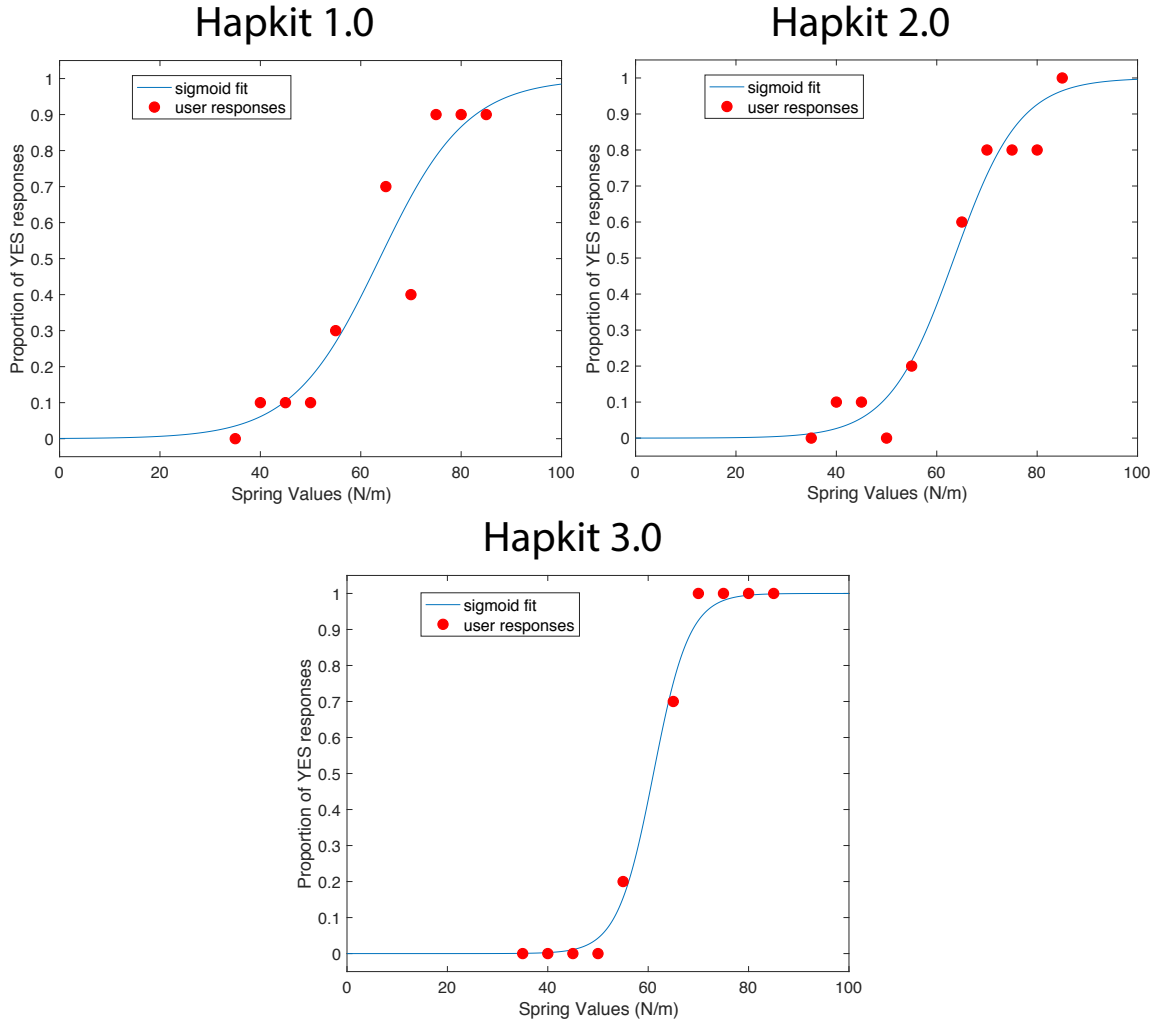


Figure 2.13: Sigmoid function fits based on one user’s responses during the stiffness discrimination test, for all three versions of Hapkit. (Adapted from [1] © 2019 IEEE.)

2.4.3 Results

Figure 2.14 shows the truncated (without outlier data) mean WF with 95% confidence intervals for our participants. The truncated mean Weber Fractions for Hapkits 1.0, 2.0, and 3.0 were 10.5%, 11.6% and 6.1% respectively. A Weber Fraction data point was considered an outlier if it was more than two standard deviations away from the mean. Figure 2.14 shows the three outlier data points.

A 2-way ANOVA test revealed a statistically significant effect of Hapkit version

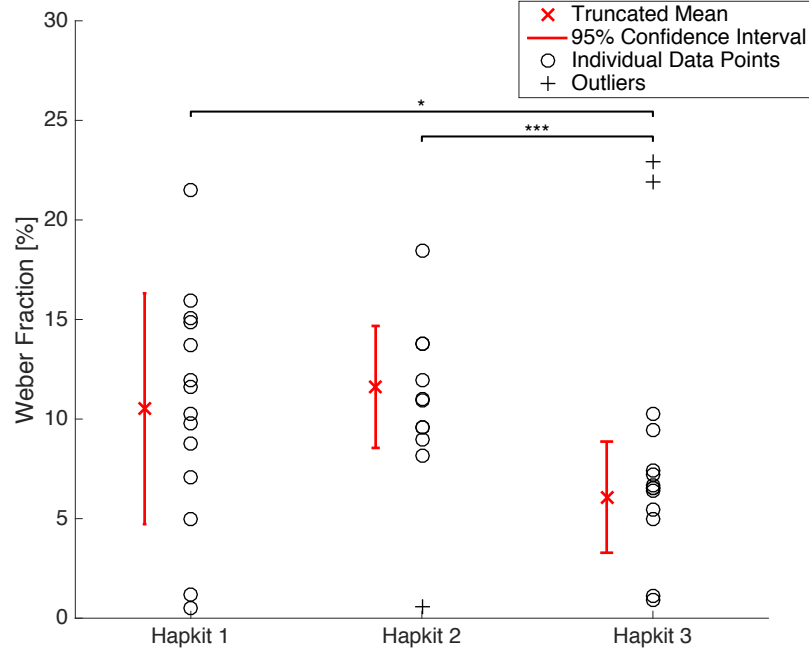


Figure 2.14: Weber fraction calculations along with the truncated mean and 95% confidence interval for Hapkits 1.0, 2.0, and 3.0. The plot highlights the statistically significant difference between Hapkits 1.0 and 2.0 versus Hapkit 3.0 obtained using an anovan and multcompare test. The outlier Weber Fractions (greater than two standard deviations away from the mean) were excluded from the calculation of the truncated mean. ((© 2019 IEEE.)

on the Weber Fraction, with $p = 0.0009$. We performed a multiple comparison test to determine which Hapkits were statistically different from each other. This revealed that Hapkit 3.0 was statistically different from Hapkits 2.0 and 1.0, but Hapkits 1.0 and 2.0 were not statistically different from each other. The 2-way ANOVA test also revealed a statistically significant effect of the user, with $p = 0.0139$. A multiple comparison test on the users found that only one user was statistically different from other users. An analysis of the responses over time revealed no statistically significant differences in correct responses that could be attributed to learning.

In the post-experiment survey, two participants reported that the task was easiest using Hapkit 1.0, no participants reported the task to be easiest using Hapkit 2.0, and four participants reported the task to be easiest using Hapkit 3.0. Five participants reported that the task was easiest using both Hapkits 1.0 and 3.0, and two participants

reported that the task was easiest using both Hapkits 2.0 and 3.0. One participant reported that the task was equally easy to accomplish using all Hapkits.

From this study's analysis and survey we conclude that Hapkit 3.0 is a better tool to discriminate springs of different stiffnesses. While cogging torque is present due to the Mabuchi motor, the capstan drive (different between Hapkits 2.0 and 3.0) enables significantly better perception of stiffness. This could translate to more accurate display of physical properties for educational applications.

2.5 Use in the Classroom

Hapkit has been used to teach a variety of concepts, such as haptics, controls, and physics, in a variety of educational environments, including undergraduate and graduate classes at Stanford, a middle school workshop, an online class, and education research. Each of these experiences informed the design of Hapkit and provided valuable insights on haptic devices as educational tools. In the following sections we discuss some of these experiences in the classroom and describe the lessons we learned, with a focus on the insights most impactful to the development and design of Hapkit.

2.5.1 Haptics Online Class

The online course had two types of offerings. The first was a real-time (instructor-paced) course where modules were posted weekly and students had to complete the viewings, online quizzes, and laboratory assignments on a weekly basis. The first version of the course was offered for 100 students who used Hapkit 1.0 for the laboratories. All of the Hapkit 1.0 components were mailed to the students. The second (and ongoing) course is self-paced, so students complete the videos, quizzes, and lab assignments on their own schedule. In this offering of the course, students have used Hapkit 2.0 or Hapkit 3.0 and students are responsible for obtaining the materials to build their own Hapkits. Grading was automated in both cases.

In this Section we will focus on the second offering of the course. We will start by

stating the course’s learning objectives and content. We will then provide an analysis of the course developed through instructor observations, forum discussions, and Stanford’s analytic service. Finally, we will present a discussion and conclusion section.

Learning Objectives

Students in the class learn how to build, program, and control haptic devices. The course focuses on the design and implementation of haptic technology, while addressing other topics as needed to motivate the course content and place it in context.

By the end of this course, students should be able to:

- Assemble, program, and simulate haptic virtual environments with their own haptic device.
- Identify the primary mechanisms of human haptic sensing.
- Understand methods for sensing the position of and actuating haptic interfaces.
- Identify salient features of a haptic device design.
- List a variety of different types of haptic interfaces.
- Implement virtual environments to render various dynamics (e.g., stiffness, damping).
- Describe applications of haptic devices.
- Develop a new haptic device or application of a haptic device.

The required background for this course is high-school physics (non-calculus) and pre-calculus. Beginner-level programming experience is helpful. Haptic device design, robotics, and mechatronics experience is not required – this was designed as a gateway course for these topics.

Course Content

The course is divided into 5 modules, and it is recommended that students complete one module per week. In each module, participants view online lectures, take online quizzes (interspersed with the lectures), and complete a laboratory assignment. Data

for each lab assignment is submitted online. The modules are shown in Table 2.4. Each module was expected to take about 10 hours of student time, although this could vary widely depending on the student’s background/experience.

The pass/fail grade is based on quiz responses (50% of the grade) and submitted laboratory data (50% of the grade). To receive a “Statement of Accomplishment” for this course (i.e., a passing grade), students must receive a score of at least 50%. This means that students likely must do at least some parts of the laboratory component in order to “complete” the course. (The first lab does not use the Hapkit, so it is possible to pass the course without building a Hapkit. This was meant to make the course more accessible to people without the necessary resources to make a Hapkit.) Of course, a student is welcome to do only the components of the course they are interested in, if receiving a passing grade is not the student’s objective.

The current instantiation of the MOOC can be accessed free of charge at <http://hapticsonline.class.stanford.edu>.

Course Analysis

The course was launched in 2014 as a self-paced Introduction to Haptics MOOC. This course has had over 7,000 students enrolled, ranging from elementary school students to graduate students to professionals. As mentioned previously, in contrast to the 2013 course, we did not distribute Hapkits to the students. Students had to obtain the parts on their own, following instructions provided online. For the initial release of the MOOC, students were given the parts list for Hapkit 2.0. Students had significant challenges implementing Hapkit 2.0. Different 3-D printer software packages interpret STL files differently, and as a result, some students found that their printers created the device in the wrong orientation. Especially on lower quality 3-D printers, this caused much of the detail of the part to merge into the support material and make part removal impossible without damaging the part. Another problem we encountered was that students did not know the target “feel” that should result from moving the height adjustment bar. It was very difficult to explain remotely to students what the right amount of friction should be. Also, due to the fact that PLA plastic is more compliant than acrylic, the correct height adjustment was harder to attain in Hapkit

Table 2.4: Modules of the Online Haptics Course

Module	Syllabus	Laboratory
Module 1	Introduction to haptic technology and human haptics - Introduction to haptics - Applications of haptic technology - Human haptics	<i>Two point discrimination test:</i> Students determine, for a few locations on the body, the distance between two contact points at the threshold of when they are perceived as a single contact point versus two separate contact points.
Module 2	Hapkit mechanical design and assembly - Design of kinesthetic/force-feedback haptic devices - Example kinesthetic haptic devices - Kinematics - Force/torque relationships	<i>Hapkit assembly:</i> Students download the parts list, obtain the components and assemble their Hapkit.
Module 3	Hapkit mechatronics - Introduction to the Hapkit Board - Position sensors for haptic devices - Actuators of haptic devices - Force-sensitive resistors (optional) - Hapkit Board analog inputs/outputs - Arduino programming language (optional)	<i>Reading the position sensor and outputting motor torque:</i> Students use an Arduino program to print the position of the handle to the screen. They then experiment with writing motor commands. Students measure the minimum and maximum distance moved by the Hapkit handle as output by their code and the minimum and maximum force that could be felt/output at the Hapkit handle.
Module 4	Programming virtual environments - Haptic rendering - Rendering haptic effects	<i>Virtual environment rendering:</i> students render a virtual spring, a virtual damper, a virtual texture, and a virtual wall.
Module 5	Mechanical characterization and simulation - Recording and modeling mechanical interactions - Validating a simulation	<i>Rendering complex virtual environments:</i> Students first render the sensation of a “clicking” a click-top pen, then they render a virtual environment of their choice.

2.0 than in Hapkit 1.0.

The attachment of the Hapkit Board was also problematic. In Hapkit 1.0, laser-cut pieces allowed students to screw the Hapkit Board to the base in three locations that were easily accessible. In Hapkit 2.0, the locations of these holes were inconsistent due to the poor tolerances of the 3-D printing process. This inconsistency, compounded by the fact that there is little room to work with when screwing the Hapkit board to the base, made it very hard to secure the Hapkit Board to the Base.

The need for a laser-cut Sector Pulley introduced an impractical manufacturing requirement for most students. Students who had no access to a laser cutter attempted to 3-D print the Sector Pulley, which was typically unsuccessful. Some students opted for altering the design since the friction drive was not working for them and posted on the course discussion board designs with other transmissions including gears. The main lesson learned from Hapkit 2.0 was the need for a thorough redesign of the Hapkit for assembly and 3-D printing, taking into account warping and inconsistency between prints.

In 2015, we released the designs for Hapkit 3.0; we noticed that students in the MOOC had started using this design because we received questions about it in the discussion boards. However, there was no specific announcement that students should use this design, and the laboratory assignments still pointed to the Hapkit 2.0 design. Nonetheless, since the introduction of Hapkit 3.0 on our website, the only problem reported with the 3D-printed Hapkit has been one student who could not find an online service in India that would 3D print the Hapkit 3.0 base at a reasonable price. We do not know exactly how many students have built a Hapkit 3.0, but we can see that 300 students total (as of 2019) have completed the programming virtual environments laboratory which requires a Hapkit to be used.

In 2018, Stanford launched a new analytic service for its online courses which gives us engagement information about the course. This service allows us to estimate that on average there have been 50 active learners every week in the course since 2018. Interesting demographic statistics include: 24.5% of these students have a high school diploma or less, 43.4% have a college degree, and 30.5% have an advanced degree; 40% of the students are 25 years old or under, 45.5% are between the ages of 26 and

40 and 14.5% are over 41 years of age; 77.9% of students are male and 21.1% are female. Students have enrolled from 121 different countries. The countries with most students have been the United States (26.9%) and in India (12.5%).

Discussion and Conclusions

The challenges faced by the students in the online course using Hapkit 2.0, inspired the design of Hapkit 3.0 as discussed in Section 2.2. Currently, Hapkit 3.0 components can be easily procured in the United States. However, common feedback from the class is that students outside of the United States have trouble obtaining the magnetoresistive sensor that Hapkit uses in order to calculate the position of its handle. To solve this problem, the device design needs to be either modified so that the components are equally accessible in all countries, or alternative designs need to be made available to account for shortcomings in the availability of parts. One possible solution is a Hapkit version that supports the use of an encoder instead of the magnetoresistive sensor to help students in markets where the magnetoresistive sensor is not available. We are motivated to continue learning from the course participants and updating the design of Hapkit around the MOOC as the community is consistently active (the course analytics show an average of 50 active students per week) and the students are representative of the population we would like to get involved in open-source haptics: they are from a varied age group and span different cultural and educational backgrounds.

2.5.2 Haptics Workshop in a Middle School Classroom

Over two weeks in May 2015, we ran a pilot workshop using Hapkit 2.0 in a middle school classroom with 32 students. The course was a modified version of our MOOC in which students were given a brief introduction to haptics and how the Hapkit works, and were aided in rendering a virtual spring and a virtual wall. Facilitators brought pre-assembled Hapkit 2.0s to the classroom for several hours each day, and students worked together in groups of two with a single Hapkit.

In this section we will describe the workshop as well as provide an analysis of the

insights gained from using Hapkit 2.0 in a middle school environment. We will begin by stating the learning objectives and course content. We will then describe how the workshop was conducted and finally discuss our findings.

Learning Objectives

Students in this workshop engage in a hands-on physics exploration using haptic technology. By the end of this workshop, students should be able to:

- Program simple haptic algorithms using the Arduino programming language.
- Understand the concept of stiffness and force.
- Analyze data from simple haptic experiments and present their results to their peers.

Course Content

The workshop was divided into 4 modules which were carried out over two weeks during the students' science class. The modules are shown in Table 2.5. The first two modules do not involve the Hapkit and were given to introduce the students to the concepts of haptics and forces which they will use in the later modules. Module 3 introduces programming with Arduino to the students and it also teaches them how to use their Hapkit Board. Module 4 then teaches students about virtual environments and to render a virtual spring and a virtual wall.

Analysis and Discussion

The middle school environment introduced many challenges beyond the university classes and MOOC described above. One difficulty was the use of a middle school classroom. The classroom desks were slanted and slippery, and a large number of extension cords and power strips were required to route power to all the desks in the room. In addition, the younger student population stressed the mechanics of the device in ways we had not seen from older students, including: knocking the Hapkit onto the floor, breaking thin pieces of 3D-printed material attaching the Hapkit Board to the base, and applying too much pressure to the motor shaft by pushing down on

Table 2.5: Modules of the Middle-School Haptics Workshop

Module	Duration	Activity
Module 1	1 class	<i>Introduction to haptic technology and human haptics</i> Students received an introductory lecture to haptics and they performed a two point discrimination test laboratory. During the laboratory, students determine, for a few locations on the body, the distance between two contact points at the threshold of when they are perceived as a single contact point versus two separate contact points. Students work in pairs and record and compare their answers.
Module 2	1 class	<i>Hooke's Law</i> Students received a short lecture on Hooke's law and then performed experiments using springs of different stiffnesses where they measured the force required to deform the springs different distances. Students learned to calculate the stiffness constant k based on their measurements.
Module 3	2 classes	<i>Introduction to programming with Arduino and the Hapkit Board</i> Students received a short lecture on programming using Arduino and then they learned how to write and download a program that turns an LED on and off using their Hapkit Board.
Module 4	3 classes	<i>Programming virtual environments</i> Students worked in pairs with one Hapkit to program and render virtual springs and virtual walls of various stiffnesses.



Figure 2.15: A student in middle school workshop moved the Sector Pulley back and forth to feel a virtual spring with Hapkit 2.0. (© 2016 IEEE.)

the handle. The latter sometimes caused the 3D-printed motor's drive wheel to shear off the motor shaft, which is very difficult to notice and debug; even if the Drive Wheel is not consistently attached to the motor shaft, as the motor spins, it can still spin the Drive Wheel some amount. We also noticed that the height adjustment bars had to be re-set often. This would be an impossible task for novices at the middle school level, indicating that such a workshop could not be run without expert facilitators present.

During the middle school workshop we recorded video and audio of three pairs of students as they learned to program the Hapkit to render the virtual spring (Figure 2.15). The students were tasked with modifying the value of the spring constant in the code, feeling the change in the haptic feedback, and commenting on what they were feeling. In our video analysis, we focused primarily on the students' interactions with the Hapkit in order to gain a deeper understanding of how the Hapkit's design interacted with the realities of the classroom. We learned that many aspects of the Hapkit design fit naturally into the classroom context, feeding the students' curiosity and guiding them towards the completion of their task, but some aspects caused confusion.

All three groups analyzed initially followed the same path and encountered the same obstacles, but all three ended the workshop in different places. They were all initially curious about the Hapkit, spending minutes examining the different components and their relationships to one another. We see these interactions as evidence of the advantages of the open design of the Hapkit. If we had chosen to hide the circuit board and motor inside a black box, the students would not have had the opportunity to explore the device in the ways that we observed. The students were highly engaged when the Hapkit correctly rendered the virtual spring, spending a significant amount of time moving the Sector Pulley back and forth. And even when the code was incorrect, the friction drive allowed the Hapkit to fail gracefully. In this case the vibrating Sector Pulley acted as an alarm that quickly alerted the facilitators. However, in one case this behavior led to confusion given the nature of the task (to render a spring). We also recorded one instance on video (and observed multiple instances not on video) of the Sector Pulley and Drive Wheel not interacting correctly, leading to lengthy repairs that spoiled students' chances of experiencing the virtual spring.

All three pairs of students uploaded malformed code that caused their Hapkits to misbehave and only one pair of students successfully rendered a virtual spring. Due to this high failure rate, for the other environment (virtual wall), we made sure to implement a more robust uploading process so that malformed code would not be used and students were able to render the environment successfully with help from the facilitators. However, the problem remained that a software failure would lead to a mechanical one, and students were not able to easily distinguish a working Hapkit 2.0 from a broken one. These observations directly informed many of the changes that were made in the design evolution from Hapkit 2.0 to Hapkit 3.0.

2.5.3 Undergraduate Freshman Haptics Seminar: ME 20N

In the Fall quarter of 2013, Professor Allison Okamura introduced a new Freshman Introductory Seminar course titled “Haptics: Engineering Touch.” The purpose of the course is to introduce students to the design and control of haptics systems, establish creative confidence, and give students the technical and mental tools to realize novel

devices. Since its creation, ME 20N has had three course offerings to date. One in 2013 where Hapkit 1.0 was used, one in 2014 where Hapkit 2.0 was used, and one in 2017 where Haplink (to be described in Chapter 3) was used. In this section we will focus on the 2014 offering. We will start by stating the course’s learning objectives and content, followed by a discussion section in which we will comment on some of the difficulties we had with Hapkit 2.0 that informed the design of Hapkit 3.0.

Learning Objectives

Students in this course learn how to build, program and control haptic devices. In the process, students gain an appreciation for the capabilities and limitations of human touch, develop an intuitive connection between equations that describe physical interactions and how they feel, and gain practical interdisciplinary engineering skills related to robotics, mechanical engineering, electrical engineer, bioengineering, and computer science.

By the end of the course, students should be able to:

- Identify the primary mechanisms of human haptic sensing, as well as the capabilities and limitations of human touch.
- Describe the salient features of a haptic device design and the physics of a haptic mechanism.
- Understand methods for sensing the position of and actuating haptic interfaces.
- Assemble and program a haptic device to create compelling touchable virtual environments.
- Explain current and invent potential future applications of haptic devices.
- Design and build a new haptic device.

The required background for this course is high-school physics (non-calculus), pre-calculus, and beginner programming.

Course Content

The course is taught using a flipped classroom model where students watch online

lectures and answer quizzes outside of the class. Class time is spent performing laboratories. The class is taught in a laboratory environment with large lab benches where students build, program, experiment and discuss the material together with the instructors. The last three weeks of the quarter are spent working on a project where students build their own haptic applications. Table 2.6 outlines the course’s syllabus.

Discussion

During the course, students had to assemble Hapkits themselves, but were given all components except for the 3-D printed and laser-cut structural components. They were asked to use existing STL files/CAD drawings to 3-D print and laser-cut the necessary parts using Stanford-owned machines. With the active assistance of the teaching staff, all 16 students were able to create functional Hapkits. However, throughout the use of the Hapkit 2.0 in this class, the instructors had to re-set the height adjustment bars when the transmission resulted in either too much or too little friction. It was clear that the novice students could not determine this adjustment on their own without the instructors demonstrating physically what it *should* feel like.

2.5.4 Graduate Haptics Class: ME 327

The graduate haptics class at Stanford University is taught by Professor Allison Okamura in the Department of Mechanical Engineering at Stanford University, and it focuses on the design and control of haptic systems. It is intended for graduate and advanced undergraduate students who have experience with dynamic systems, programming, and feedback control. Since its creation, Hapkit has been used in four offerings of ME 327. Hapkit 1.0 was used in 2014. Hapkit 3.0 was initially tested in the 2015 offering of this course for 30 students. Since that initial release, Hapkit 3.0 has been used in two other offerings of ME 327: in January 2018 (30 students) and April 2019 (40 students). In this section we will focus on the latest offering of the course (Spring 2019). We will describe the learning objectives and course content, followed by a discussion of the use of Hapkit 3.0 during the class.

Table 2.6: Modules of the Freshman Haptics Seminar Course

Week	Syllabus	Laboratories
Week 1	<ul style="list-style-type: none"> - Introduction to haptics and human touch - Experiments with human touch 	<p><i>Two-point discrimination:</i> Students determine, for a few locations on the body, the distance between two contact points at the threshold of when they are perceived as a single contact point versus two separate contact points.</p> <p><i>Create Hapkit components:</i> Students use Solidworks and customize, 3-D print, and laser-cut the structural components of Hapkit 2.0.</p>
Week 2	<ul style="list-style-type: none"> - Haptic device design, kinematics, and forces. - Product Realization Laboratory introduction 	
Week 3	<ul style="list-style-type: none"> - Hapkit assembly and kinematics - Introduction to programming 	<p><i>Hapkit Assembly:</i> Students assemble their Hapkits using the structural components they created.</p>
Week 4	<ul style="list-style-type: none"> - Introduction to electronics and the Hapkit Board - Sensors 	<p><i>Hapkit sensor reading and calibration:</i> Students develop an Arduino program to read the output from the magnetoresistive sensor on the Hapkit Board and calculate the Handle's position.</p>
Week 5	<ul style="list-style-type: none"> - Actuators - Hapkit Calibration 	<p><i>Motor Output and Graphics:</i> Students calculate the torque outputting capabilities of Hapkit's motor and develop an Arduino program which takes the position they calculated in the previous laboratory and outputs a force at the Handle.</p>
Week 6	-Programming virtual environments	<p><i>Haptic Rendering:</i> Students write code to render a virtual spring, a virtual damper, a virtual texture, and a virtual wall.</p>
Week 7	-Recording and rendering realistic environments	<p><i>Something new:</i> Students work in pairs to complete two of the following activities: create a new virtual environment, combine two Hapkits for teleoperation, vibration feedback, creating a graphical display to accompany a haptic rendering, or use a force sensitive resistor for input.</p>
Weeks 8-10	-Work on final project	<p><i>Work on final project:</i> Students work in teams to create novel haptic applications.</p>

Learning Objectives

Students in the class study the design and control of haptic systems. The focus is on device modeling (kinematics and dynamics), synthesis and analysis of control systems, and design and implementation and human interaction with haptic systems. Coursework includes homework/laboratory assignments and a hands-on project.

By the end of this course, students should be able to:

- Identify the primary mechanisms of human haptic sensing.
- Understand a number of methods for sensing the position of and actuating haptic interfaces.
- Describe the differences between grounded and ungrounded force feedback.
- Identify salient features of a haptic device design.
- List a variety of different types of haptic interfaces.
- Implement controllers to render various dynamics (stiffness, damping, inertia).
- Describe and implement basic telemanipulation controllers.
- Understand the causes of instability and virtual reality and teleoperation systems.
- Design psychophysical and perceptual tests.
- Describe applications of haptic devices.
- Develop a new haptic device or application of a haptic device.
- Read, evaluate, and critique research papers.
- Design and deliver a research presentation.

Course Content

The course is comprised of 12 lectures that are given two times per week and five assignments that should be completed every week. In the last four weeks of the course, students investigate and present relevant haptics papers and work on their

final projects in teams of 3 or 4. In the last week of the course, students present their projects to the Stanford community who are invited to a demonstration session. Table 2.7 outlines the class syllabus.

Discussion

As mentioned above, our first test of Hapkit 3.0 began in October 2015 in a 30-person graduate haptics class (ME 327) at Stanford University. Students in the course were able to assemble the capstan drive transmissions on Hapkit 3.0 in less than five minutes. This transmission also proved to be robust to unstable virtual environment rendering. During an intensive first haptic rendering assignment, we observed many unstable renderings during which no capstan drives became unwound. However, the robust transmission combined with stops at the end of the Hapkit 3.0 workspace (not present in Hapkit 2.0) caused about a third of the 3D printed drive wheels to shear off of the motor shafts. As a result, we improved the design of the drive wheel by increasing the diameter of the 3D-printed piece, reducing the thickness of the rubber tube to make sure the total drive wheel diameter stayed constant, and 3D-printing the drive wheel with 100% infill.

During the most recent completed course offering (April 2019), all Hapkit 3.0 devices were robust enough to withstand various unstable renderings and interactions. As mentioned in the course content section, the second half of the course tasked students with developing the design and control of their own haptic device with a well thought-out purpose or application. During this part of the course, we were very pleased to observe how students took advantage of the customization capabilities of Hapkit 3.0 enabled by its 3-D printed design. Several students built on the design of Hapkit for their novel application. For example, Figure 2.16 shows one team's design and implementation of their project to provide haptic feedback to an electronic keyboard that is similar to the kinesthetic response from a piano or harpsichord.

As of this writing in Spring 2020, Hapkit 3.0 is currently being used in a 65-person online version of ME 327, necessitated by the ongoing COVID-19 pandemic. This course significantly increased in enrollment compared to previous years because of the ability of the course to offer a hands-on laboratory experience despite the need

Table 2.7: Modules of the Graduate Haptics Course

Week	Syllabus	Assignment
Week 1	<ul style="list-style-type: none"> -Introduction to haptics and course overview -Kinesthetic haptic devices: design and kinematics 	<i>Introduction to Hapkit and Kinesthetic Haptic Devices:</i> <ul style="list-style-type: none"> -Customize a Hapkit Handle and 3D print it. -Brainstorm and develop ideas for new haptic applications. -Understand the 1-DOF Kinematics of Hapkit.
Week 2	<ul style="list-style-type: none"> -Kinesthetic haptic devices: dynamics and control -Kinesthetic haptic devices: sensors and actuators 	<i>Dynamics and Control of Kinesthetic Haptic Devices:</i> <ul style="list-style-type: none"> -Derive a block diagram for a 1-DOF haptic device in the time and Laplace domains. -Simulate using matlab the behavior of that 1-DOF haptic device rendering virtual walls of different stiffnesses.
Week 3	<ul style="list-style-type: none"> -Hapkit Distribution and Assembly -Kinesthetic haptic devices: 1-DOF rendering 	<i>Haptic Rendering on a 1-DOF Kinesthetic Haptic Device:</i> <ul style="list-style-type: none"> -Assemble Hapkit. -Calibrate position sensor and force output of Hapkit. -Render basic virtual environments.
Week 4	<ul style="list-style-type: none"> -Kinesthetic haptic devices: multi-DOF design and kinematics. -Kinesthetic haptic devices: multi-DOF rendering 	<i>3-DOF Rendering:</i> <ul style="list-style-type: none"> -Use a Phantom Omni to render 3-DOF virtual environments.
Week 5	<ul style="list-style-type: none"> -Teleoperation: Implementation and Transparency. -Teleoperation: Stability and Setup 	<i>Teleoperation and Graphics:</i> <ul style="list-style-type: none"> -Create graphics for various virtual environments using Processing. -Implement and analyze different teleoperation environments using two Hapkits.
Week 6	<ul style="list-style-type: none"> -Human haptics: Mechanoreceptors and Kinesthesia 	<i>Final Project ideas:</i> <ul style="list-style-type: none"> -Finalize project proposal.
Weeks 7-10	<ul style="list-style-type: none"> -Human haptics: user studies -Student paper presentations and project demonstrations. 	<i>Work on paper presentations and final project:</i> <ul style="list-style-type: none"> -Present one paper which is relevant to the final project. -Build a novel haptic application.

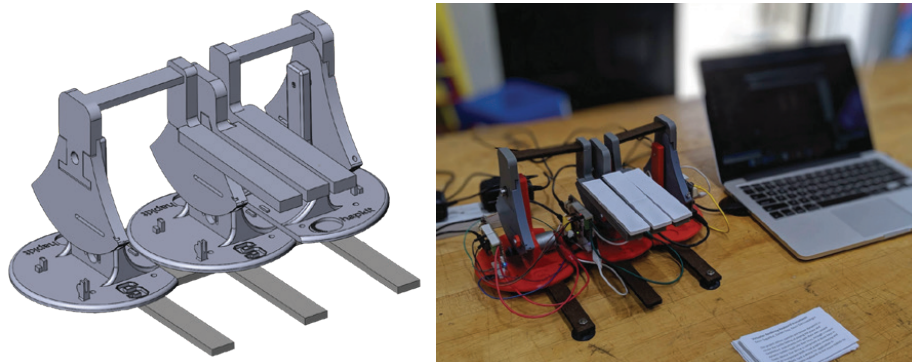


Figure 2.16: Design of a haptic keyboard that uses three modified Hapkit 3.0 devices to provide feedback to three different keys. (© 2019 IEEE)

to offer a normally on-campus course in an online format, The prior experience making and distributing Hapkits to students in previous on-campus and online formats greatly facilitated the transition to online learning and teaching in Spring 2020. However, one consideration is that additional tools (normally accessed in a physical laboratory space) had to be mailed to students, increasing the cost of each kit by approximately \$10.

2.5.5 Teaching Trigonometry using Haptics

In the Spring of 2016 we set out to test a new concept in teaching with haptics – whether it could be used for rendering abstract concepts in mathematics and support learning of these concepts. This work was done in collaboration with Richard Davis and Paulo Bilkstein of the Stanford School of Education. We conducted a small pilot study using Hapkit 3.0 with four high school students, and recorded video and audio of them learning trigonometry using a system developed by us meant to use haptics to bridge the graph of a trigonometric function with its equation and the unit circle. The results and detailed explanation of the analysis from this study are published in [40]. In the following sections we provide a summary of the experiment and findings.

Methods

We recruited four high-school seniors (two girls, two boys, aged 17 to 19) from a

charter school in the San Francisco Bay Area serving one of the lowest-income communities in California, USA. These students had previously studied trigonometry, but lacked an understanding of the relationship between the unit circle and the graph of sine/cosine on the Cartesian plane. None of them had seen or used a haptic device before the study. Students worked in pairs to promote discussion and collaboration. We collected three primary sources of data. We captured students' interactions with the software by recording screen and webcam video from the interaction computer, audio and video of student interactions with the haptic device and one another with a tripod-mounted camera; and student drawings of the graphs of sine and cosine.

Materials

For this study we used Hapkit 3.0 as described in Section 2.2. We also developed a software interface: Trigonometry Explorer, (Figure 2.17) which illustrates the relationship between the graph of the trigonometric functions of $\sin(x)$ and $\cos(x)$, and the unit circle. The program has two parts: the graphical user interface (GUI), programmed in Processing to run on any personal computer, and the Arduino-based software on the Hapkit Board, which renders haptic feedback and sends position data to the computer. Figure 2.17(a) shows Trigonometry Explorer's graphical user interface (GUI)'s eight components:

- *Graph Box*: Shows graph of the function chosen by the user, and the point in the graph the user is currently on using the User Marker.
- *Function Drop Box*: The user “drops” the function he or she wants displayed in the Graph Box into this box.
- *Amplitude and Frequency Sliders*: Allow the user to change the amplitude and frequency of the graph chosen via the Function Drop Box.
- *Functions*: The functions the user can choose to display and feel.
- *Start Quiz Button*: The software supports a “quiz” modality that was not used for this work.

- *Hide Graph Button*: Hide the graph from the user. The user can still explore the chosen graph using Hapkit, but will not see it in the graph box.
- *Unit Circle Box*: Always displays the unit circle and the value of the graph the user is currently on.
- *User Marker*: Represents the value of the current graph on the Graph Box and the Unit Circle Box.

A student uses Hapkit to move across a graph of a trigonometric function, and “feel” it. Hapkit translates the position of the user in the graph to a force at the handle using the equations $F = A \sin(fx)$ or $F = A \cos(fx)$, depending on the function chosen. x is the horizontal position of the user in the graph, f is the frequency of the function, and A is the amplitude of the function. The student can change the function’s amplitude and frequency using the GUI and see and feel the effects of those changes, while observing the relationship between the graph and the unit circle (Figures 2.17(b) to 2.17(e)). When moving the handle, the student feels the forces computed from the equations above as either resisting or assisting the motion. The larger the amplitude above the x -axis, the more force pushes the handle to the right. The larger the amplitude below the x -axis, the more force pushes to the left (Figure 2.17(f)).

Procedure

The students received an hour-long lesson about the relationship between the unit circle and the trigonometric functions sine and cosine. The lesson was designed for students who had studied some trigonometry and were familiar with sine and cosine, but did not fully grasp the relationship between the unit circle and these trigonometric functions. The lesson’s learning goal was to help students understand the relationship between the unit circle and the graph of sine and cosine on the Cartesian plane.

The lesson consisted of four 15-minute phases:

- *Phase 1: The Initial Lecture*. During the first phase, we (re)introduced the students to trigonometric concepts. This phase did not directly address any research questions, but ensured that the students were familiar with the concepts

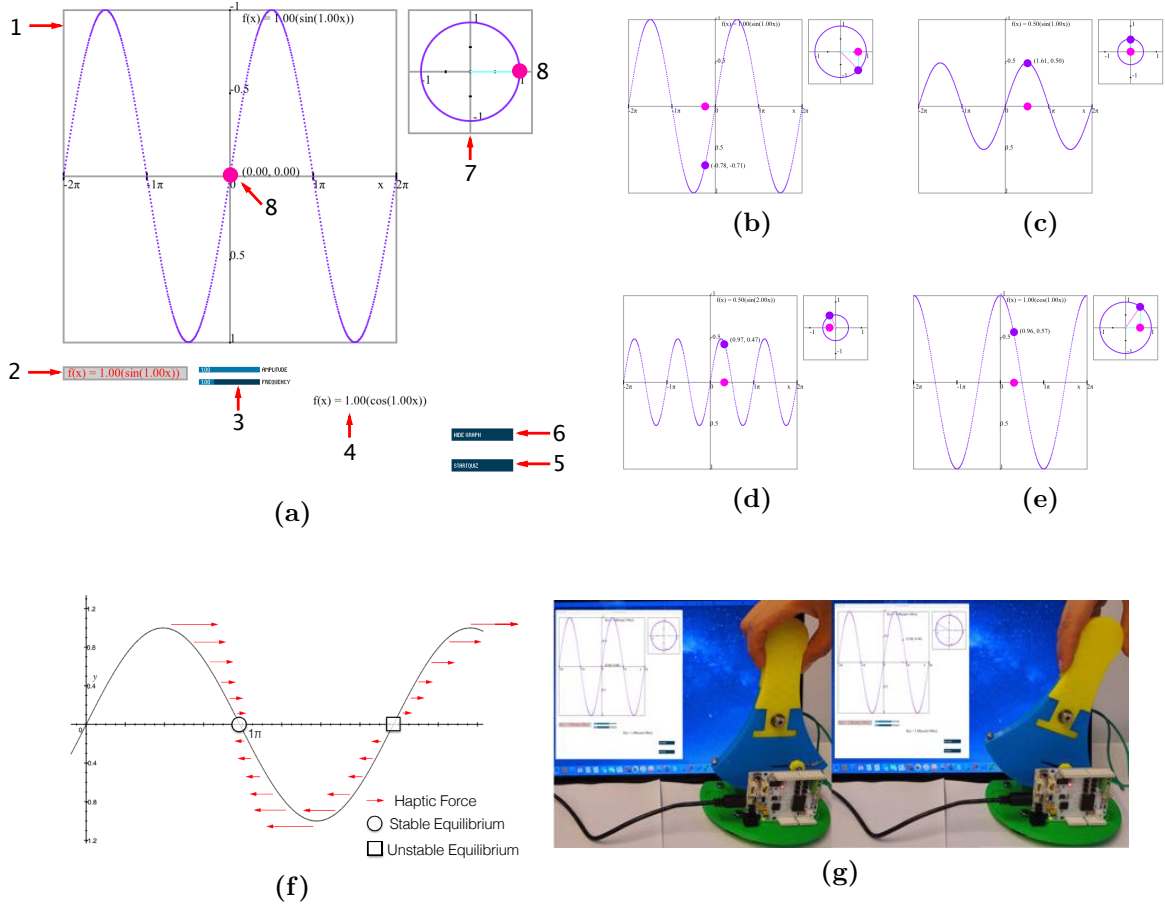


Figure 2.17: Trigonometry Explorer Software: (a) Initial screen of Trigonometry Explorer’s graphical user interface with its components: (1) Graph Box. (2) Function Drop Target. (3) Amplitude and Frequency Sliders. (4) Draggable Functions. (5) Quiz Button. (6) Hide Graph Button. (7) Unit Circle Box. (8) User Marker. (b–e) Four different use cases for Trigonometry Explorer: (b) Sine function selected with amplitude 1 and Hapkit at -0.78 rad. (c) Sine function selected with amplitude 0.5 and Hapkit at 1.61 rad. (d) Sine function selected with amplitude 0.5, frequency 2 and Hapkit at 0.47 rad. (e) Cosine function selected with amplitude 1 and Hapkit at 0.96 rad. (f) The displacement of the Hapkit Handle represents the input to the sine function, and the forces rendered by Hapkit represent the output of the sine function. (g) Moving Hapkit side to side moves the User Marker along the graph and the unit circle in Trigonometry Explorer. (Adapted from [40])

in the study.

The instructor led the student pair through a lesson on the unit circle, sine, and cosine. The instructor lectured with PowerPoint slides, asked the students questions, and occasionally moved to the whiteboard to elaborate on specific concepts. Hapkit was not used.

- *Phase 2: Derive the Graph of Sine.* The second and third phases of the study were designed to expose the ways in which the haptic feedback impacted the students' understanding of the connections between the two different trigonometric representations. As the students worked through the tasks in these phases, we were able to observe the impact of the haptic feedback through the students' actions, their discussions with each other, and their questions to the experimenters.

In Phase 2, students were asked to derive and draw the graph of $\sin(x)$ using the Hapkit and the unit-circle software. At this time, the actual graph of $\sin(x)$ was not displayed in the central window. The students were given a piece of paper with the central window's coordinates printed out and asked to derive and draw the graph.

- *Phase 3: Derive the Graph of Cosine.* The third phase was identical to the second, except that students were asked to derive and draw the graph of $\cos(x)$. This was more difficult than deriving and drawing $\sin(x)$, because it required students to translate the horizontal position of the point rotating around the unit circle to the vertical position on the graph.
- *Phase 4: Determine the Unknown Frequency and Amplitude.* The fourth phase of the study served to check robustness of the students' interpretation of the haptic feedback. Since the students had not been taught about frequency and amplitude in the initial lecture, their ability to correctly guess the unknown frequency and amplitude would depend on their ability to correctly interpret the haptic feedback and unit-circle animation. The instructor increased task difficulty by changing frequency and amplitude. Before this, frequency and amplitude of sine and cosine were both fixed at 1. Now, the unit circle radius

changed with amplitude, while the rate at which the point moved around the circumference changed with frequency. Students could not see the adjusted values of frequency and amplitude. Then students were asked to determine the frequency and amplitude by viewing the “unit circle” representation and feeling the forces rendered by Hapkit.

Results and Discussion

Davis et al. [40] presents a detailed analysis of the students’ utterances and actions from their sessions, as well as findings in chronological order grouped into sections that reflect themes that emerged during the analysis. Here we present a summary of the results of that analysis.

Before working with Hapkit and Trigonometry Explorer, the students were unable to make the connection between the unit-circle representation of sine and cosine and the function’s graphs on the Cartesian plane. With the support of the haptic representation rendered by Hapkit and Trigonometry Explorer, the students learned to make this connection. The haptic representation was initially confusing for the students; however, over the course of the study they learned to correctly interpret the haptic representation. The student’s understanding of the connection between the two different trigonometric representations changed in tandem with their ability to correctly interpret the haptic representation.

One of the students learned to correctly interpret the haptic feedback early in the study, and encountered little resistance while working through the remaining problems. In contrast, that student’s partner failed to interpret the haptic feedback in Phase 2, struggled with the interpretation in Phase 3, and then with the help of the first student, learned to correctly interpret it in Phase 4. Once he learned to correctly interpret the haptic representation, he was able to complete the activity in the final phase (determining the unknown frequency of the equation from the haptic feedback).

Due to the small sample size, the results of this study are preliminary. We decided to include it in this thesis as a novel potential application of haptics to be used to explore rendering of abstract mathematical concepts. The haptic feedback incorporated into Trigonometry Explorer is different: the forces rendered by Hapkit

have no real-world analogue. That is, it is impossible to go into the physical world, find a sine wave, and learn how it feels by touching it. In cases like this, the haptic feedback becomes symbolic, which means (in this case) that the students' ability to use Hapkit to help them imagine the hidden graphs of sine and cosine is dependent on their ability to make meaning of, or interpret, the haptic feedback.

Although our results are preliminary, there is evidence that the students working with Trigonometry Explorer were aided in their ability to make sense of the multiple representations by the haptic feedback from Hapkit 3.0.

2.5.6 Discussion

The use of Hapkit in the different educational environments described previously drove its design evolution, and showed its capabilities as an educational and research tool. First, we will address the design evolution. As mentioned in previous sections, the starting point for designing 3D-printed haptic devices was Hapkit 1.0 [48]. We designed Hapkit 2.0, as an easier-to-manufacture version of Hapkit 1.0 with 3D-printed components and an inexpensive motor. Hapkit 2.0's use in the haptics online class (Section 2.5.1), middle school workshop (Section 2.5.2), and freshman haptics seminar (Section 2.5.3) highlighted several challenges which drove the design of Hapkit 3.0. First, through the haptics online course and the freshman haptics seminar, we learned that it was difficult for students to adjust the height adjustment bar properly and that instructors needed to aid students and show them what their devices should feel like. From this experience, we learned that Hapkit should have a transmission that does not need an expert to adjust it and decided to use a capstan drive for Hapkit 3.0 (Section 2.2.4). The middle school environment tested the robustness of Hapkit 2.0; we learned that Hapkit 2.0's components could warp due to the 3D printing process resulting in brittle structural components that would break off easily. From this experience we designed Hapkit 3.0's structural components (Section 2.2.2) specifically for the 3D-printing process; with a rounded design to prevent warping, a snapping mechanism to attach Hapkit Board, and we added suction cups to better ground it to the surfaces it is being used on.

Second, we will address Hapkit’s capabilities as an educational and research tool. During the middle school workshop we observed students taking advantage of Hapkit’s open design to explore, examine, and ask questions about the different components and how Hapkit works. This taught us that even when the main topic of the course is not the device itself, it is valuable to explore open designs which will inspire students’ curiosity and engagement with the material. In the graduate course, ME 327, (Section 2.5.4) we observed how students took advantage of the customization capabilities of Hapkit 3.0 to design novel haptic applications. In the online haptics class, we demonstrated that Hapkit is a tool that can be used for distance laboratories. Students in the online haptics class have been able to obtain the parts, manufacture their own Hapkits, and complete several laboratories where they render virtual environments. At the time of the writing of this dissertation, Hapkit 3.0 is being used in the Spring of 2020 to teach an offering of ME 327 remotely due to the COVID-19 pandemic. Finally, Hapkit 3.0 was used in a pilot study to render the waveforms of $\sin(x)$ and $\cos(x)$ using force-feedback and help teach trigonometry to high school students. This last example demonstrates the use of Hapkit for teaching an abstract math concept. In this example, the haptic force-feedback is used as an interactive media to display a math concept as opposed to using the haptic device to teach haptics or robotics as was done in ME 327 and ME 20N.

2.6 Conclusions

Haptic Paddles have been used successfully in a number of undergraduate and graduate engineering courses, such as haptics and controls [42, 43, 44, 45, 46, 47, 62, 63], in university laboratory settings. In order to enable broader research on the role of haptics in education, such as embodied cognition for K-12, haptic devices need to be designed not as laboratory equipment, but as personal devices. In this chapter we presented the design evolution of the Hapkit family of devices which were specifically designed to not be laboratory equipment and to support a wide variety of educational applications. We describe specific design changes and the reasoning behind them as Hapkit evolved from version 1.0 to 3.0 in order to pass on our learnings to other

open-source haptic device designers.

Our goal in designing Hapkit was to create an accessible, open-source device that facilitates both education and haptics science. Hapkit has been used at Stanford University in undergraduate and graduate courses on haptics and controls, and it is being used in a self-paced massive open online class (MOOC) on haptics that has had over 5,000 students enrolled. Hapkit was also used in a collaboration between Stanford University and the University of British Columbia [40,52] to understand the role of haptics in learning as well as how to best design course material that takes advantage of haptics. In this chapter, we discussed specific educational uses of Hapkit and highlighted the insights which drove the design evolution of the Hapkit family of devices.

This chapter also presented a comparison of the mechanical properties of the Hapkit family of devices and their effects on stiffness discrimination by users. The analysis employed a frequency domain system identification method to compare the mechanical properties of different Hapkits as well as previous haptic paddles from the literature. The purpose of this analysis was to report the capabilities and characteristics of the Hapkit family of devices in order to inform future researchers using Hapkit in their education applications as well as provide designers of inexpensive haptic devices a baseline to compare to. With these tools, guidelines into the rendering requirements for educational applications could start to be established. Through the user study we found that users performed better in a stiffness discrimination task using Hapkit 3.0, in spite of it having a motor with cogging torque. This was a surprising result since the magnitude of the cogging torque is significant and can be felt at the handle typically as a force of 0.3 N as reported in Section 2.3.4. From this result, when performing stiffness discrimination tasks, damping in the transmission is a more significant effect than cogging torque at the motor for Hapkit. Because users perform better in stiffness discrimination tasks using Hapkit 3.0, Hapkit 3.0 has the potential to render a wider variety of distinct stiffness virtual environments. We propose that this makes it a better educational and research tool as it offers more possibilities and freedom for development. In future work, it would be interesting to test the effects of the cogging torque in Hapkit 3.0 in the rendering of different virtual

environments such as rendering a very smooth surface with occasional virtual bumps, which have very different perceptual requirements than those of rendering a spring.

By using Hapkit 3.0 in different educational environments, we have learned anecdotally that one of its best features is its capability to be customized and personalized by students, giving them a sense of ownership over their individual devices. In the next chapter of this dissertation we discuss the design of a novel 2-DOF educational haptic device inspired by Hapkit's customization capabilities.

Chapter 3

Haplink: A Serial 2-DOF Kinesthetic Device

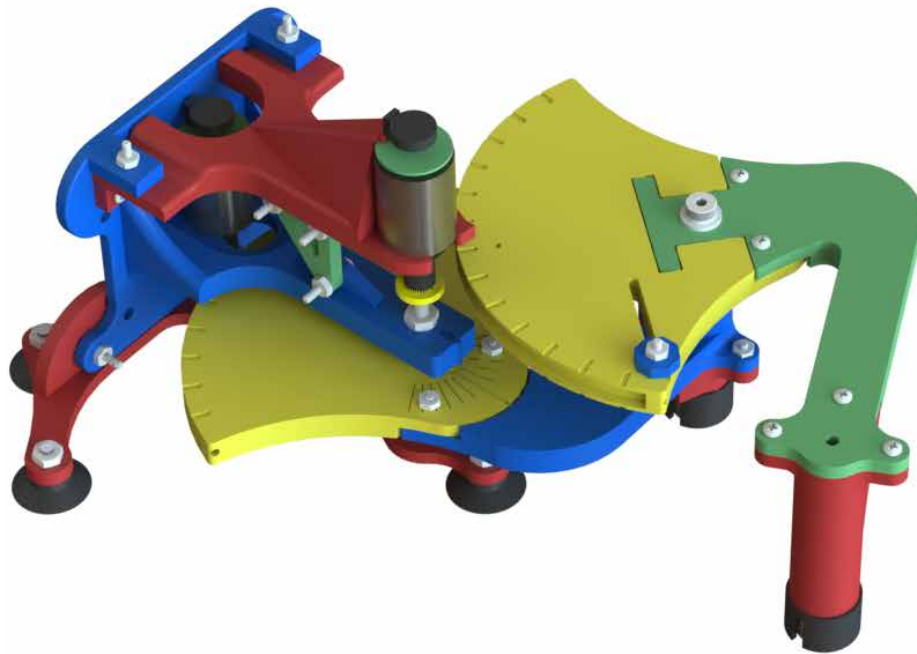


Figure 3.1: Haplink: An open-source, 2-DOF, 3D-printed, kinesthetic haptic device

Portions of this chapter are reprinted from [64] © 2017 IEEE.

3.1 Introduction

Chapter 2 discussed 1-DOF open-source kinesthetic haptic devices for educational applications, including previous devices and a new contribution, Hapkit 3.0. Higher-degree-of-freedom kinesthetic haptic devices have also been made open source. One example is the Snaptic Paddle, created by Okamura et al. [63] in 2005. This device allows two modular 1-DOF devices to be connected to form a single 2-DOF device through modifications to the original haptic paddle design. The Snaptic Paddle's spherical kinematics are similar to that of a traditional joystick in its 2-DOF configuration. The device allows for 2-DOF simulations at a low cost, but the authors report noticeable backlash and excessive flexibility in the structural components of the device. Another example of an open-source, kinesthetic haptic device is the Wooden Haptics Project [65]. This is a 3-DOF haptic device designed for educational applications and presented as an open-source kit made out of laser-cut plywood structural components that students can assemble themselves. Wooden Haptics was reported by the authors to render virtual environments with performance comparable to commercially available 3-DOF haptic devices. However, it has a high cost (\$3000 reported by the authors) and a long assembly time required (11 hours over 4 days). Outside of the research groups that originally developed the devices, there have been no reported contributions to the design of either the Snaptic Paddle or the Wooden Haptics Project.

An open-source project that has been iterated on by multiple groups in the haptics scientific community is the Pantograph mechanism. In 1994 Ramstein and Hayward [35] created the Pantograph haptic device using a five-bar linkage mechanism, and released their mechanical designs to the scientific community. In the early 2000's Campion et al. [66] redesigned the Pantograph and made both the hardware and control software open source. This device uses two Maxon Re-25 motors with 100,000 counts per revolution encoders from MicroE Systems Inc, structural components that are machined from aluminum, and a PCI card from Quanser Inc. The Pantograph

has the capability of reading the normal force applied by the user on the device through a load cell and has the option to add an accelerometer in order to gather detailed information about the high-frequency movements of the end-effector. Due to its high-fidelity rendering capabilities, this device has been built and used by numerous research groups to perform research into rendering and perception, such as using stochastic algorithms to render Gaussian textures [67] and using lateral force fields to study human perception [68]. Several research groups have iterated on the design of the Pantograph and built their own versions such as Sketch et al. [69] who built a skin-stretch feedback device using a Pantograph mechanism to evaluate the effects of skin-stretch feedback for control of brain-computer interfaces, and Gillespie et al.'s cTouch: A device that uses the Pantograph mechanism with compliant joints and voice coil actuators in order to achieve friction-free motion. This device was used to teach controls and dynamics classes at the University of Michigan [70].

In all of the above examples, the control of the Pantograph is done on a PC computer connected to the device through a PCI card, or similar, and the devices are made out of either laser-cut or machined structural components, which are difficult for schools to obtain- limiting accessibility for educational applications. In 2016 two Pantograph-like devices that addressed the accessibility concerns above were designed: Haplet [71], built by Gallacher et al., which has an open architecture electronics board based on Stanford's Hapkit Board [48] and open-source designs using different structural materials such as laser cut acrylic and 3D-printed PLA, and Graphkit [64], a 3D-printed device that combines two Hapkit 3.0's into a Pantograph mechanism. Both Haplet and Graphkit have been used in educational applications. Haplet was used in the 2017 Student Innovation Challenge at the World Haptics Conference in Fürstenfellbruck in Germany, in which students were tasked with creating innovative education solutions. Graphkit was used to teach an undergraduate robotics class in Kyoto, Japan. Although the Pantograph mechanism has been used for educational applications, the derivation of the kinematics associated with the device is very complicated, making the device unsuitable for certain educational applications such as entry-level courses where an understanding of the kinematics would be necessary in order to program the device.

In this chapter we investigate the design of a new haptic device, Haplink, that was specifically created to function as a 1- and 2-DOF device and where the kinematics of the 2-DOF device build on the kinematics of the 1-DOF device. Haplink was designed with the idea that students will benefit from learning concepts incrementally, and that a device that itself increments from one to two degrees of freedom using a simple mechanism will aid in this process. Haplink can be used as a 1-DOF device similar to Hapkit, as well as a 2-DOF device by adding a second Hapkit capstan mechanism in series with the first. In this chapter we also describe how it was used in the classroom, specifically a Freshman Introductory Seminar at Stanford University in Autumn 2017.

Section 3.2 discusses the design implementation as well as the challenges faced while creating a 2-DOF device meant to be customizable, open source, and built by students. Section 3.3 describes the force output and resolution characterization of the device, and Section 3.4 describes how the characterization of the device informs the rendering of virtual environments with Haplink. Finally, Section 3.5 introduces the use of Haplink in the Freshman Introductory Seminar on haptics. Portions of this chapter were published in [64].

3.2 Haplink Design

Haplink is a kinesthetic haptic device that derives from the 1-DOF Hapkit, and has been designed to function as either a 1- or 2-DOF device (Figure 3.2). A number of changes to the Hapkit 3.0 design were made to achieve this. These modifications will be discussed in the following subsections.

3.2.1 Design Goals

As with Hapkit, our design goals for Haplink can be categorized into manufacturing and assembly requirements, and dynamic and rendering requirements. Here we describe these goals.

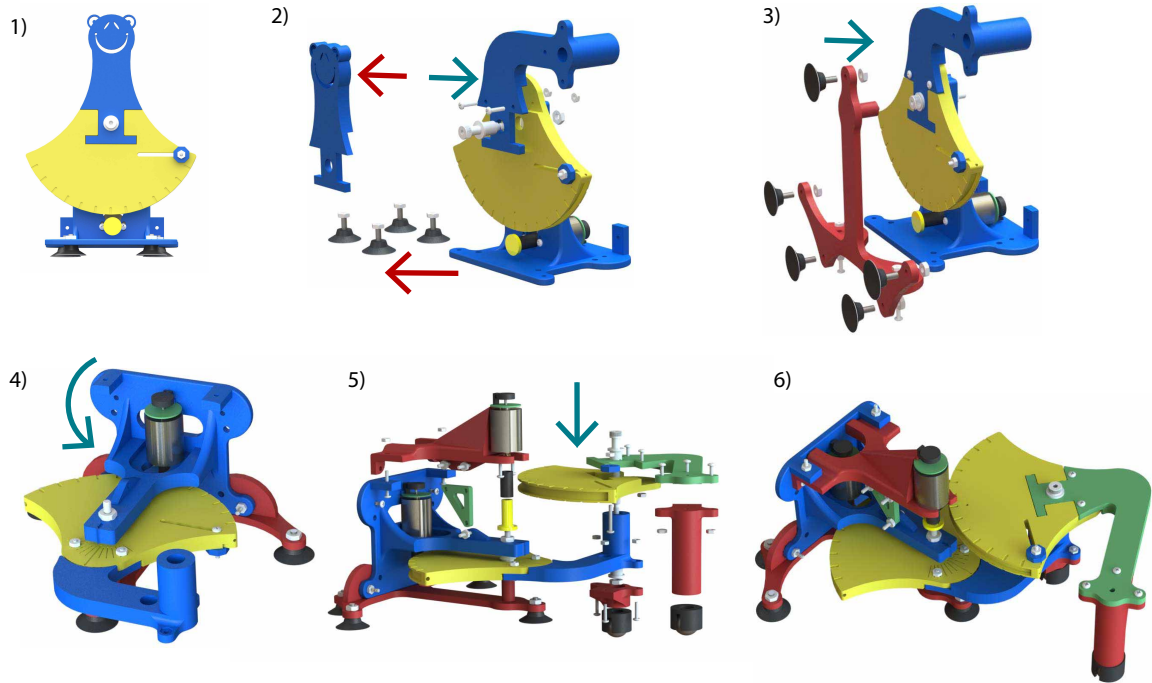


Figure 3.2: Haplink assembly process: 1) Start with your assembled and customized 1-DOF Haplink 2) Remove the handle and suction cups and insert Haplink Handle A in its place 3) Attach it to the Haplink Base and attach the suction cups to the new base. 4) Flip the assembly on its front side 5) Attach Base 2, Sector B and Handle B 6) Finished 2-DOF Haplink

Manufacturing and Assembly

Similarly to Hapkit, the design priority for Haplink is to be an “open implementation” kit meant to be assembled by students. Haplink allows for a similar level of customization as Hapkit by allowing users to create their own handles, and all of the standard files (parts lists, solid models, instructions) are freely available at <https://sites.google.com/view/haplink>. Haplink is not meant to replace Hapkit, but rather it is meant to be used as a platform to explore more advanced concepts. Hapkit’s users range from middle school to graduate students, while Haplink is intended for use by high-school to graduate students and beyond. With this in mind, we allowed for the assembly process of Haplink to be more complex than Hapkit’s, yet still unambiguous and robust. We also designed Haplink to be an uncovered device, with its own electronics board.

Kinematics and Rendering

Haplink was built on the same mechanical principles as Hapkit. Since it is an impedance-type device, it follows that the device be backdrivable, have a transmission with minimal friction, have moving parts with low inertia, and use a motor with low cogging torque. In order to aid students in understanding the differences and similarities between 1- and 2-DOF systems, another requirement was to use a mechanism that derived from the 1-DOF device to transform it to 2-DOF. Hence, in its 1-DOF configuration, Haplink uses one rotational joint that is identical to that of Hapkit's, and combines two of these rotational joints using a serial linkage to become a 2-DOF device.

3.2.2 Haplink Design Implementation

The following section describes how Haplink was designed to meet the requirements discussed in Section 3.2.1.

Electronic Hardware

As described in Chapter 2, Hapkit uses an Arduino Uno-based [72] board (Hapkit Board) in order for its electronics to be accessible to a wide user base. Using the Hapkit Board, virtual environments in Hapkit can be rendered with a frequency of 1 kHz [39]. Due to the increased computation load required by the 2-DOF Haplink (e.g., two sensors, calculating 2-DOF kinematics, etc.), the maximum haptic loop rate dropped to 500 Hz using the Hapkit Board. While this allowed the rendering of simple virtual environments, such as the inside of a box [64], it was not sufficient for modeling more complex environments or to communicate with a computer over its serial bus. To allow more complex virtual environments to be modeled, the electronics of Hapkit were re-designed for Haplink as follows:

- **Processor:** Haplink uses a Nucleo-F446ZE [73] board (Figure 3.3), which is an STM32 Nucleo-144 development board with an STM32F446ZE MCU microcontroller from ST Microelectronics (www.st.com). This board can run up to

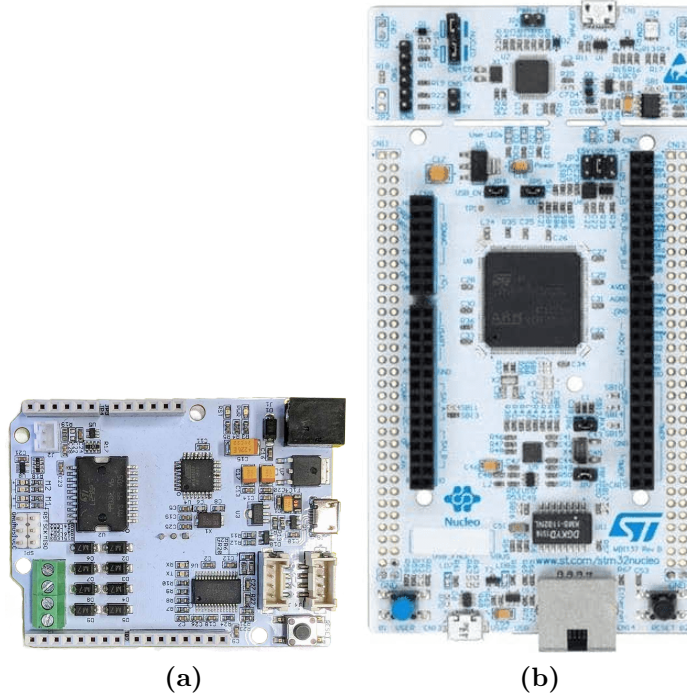


Figure 3.3: Processors used by Hapkit and Haplink. (a) Hapkit uses the Hapkit Board. It is a custom board created based on the Arduino Uno platform. It includes a motor driver and an sd card reader. It can be purchased from Seeed studio for \$35 dollars. (b) Haplink uses the Nucleo-F446ZE processor. This board can run up to speeds of 180 MHz and has native USB, direct memory access, 14 timers and 144 ports. It can be purchased from Mouser, Digikey and other electronic vendors for \$19 dollars.

speeds of 180 MHz and has native USB communication, which allows communication with a computer at USB speeds. Using the Nucleo board we are able to achieve a haptic loop rate frequency of 4.4 kHz while rendering a virtual box and printing to the screen. Haplink uses the USB protocol on the Nucleo board to send the end effector position to a computer. This position can be received by any PC software that allows serial communication such as Processing [74], the Arduino Terminal [75], etc. Using Processing to receive the position of the end effector enables graphic rendering. The Nucleo-F446ZE can be purchased from Mouser Electronics (www.mouser.com) for \$19, its design is open source,

and it can be programmed using the mbed online compiler [76], which is a free and user-friendly online IDE that allows users to program Nucleo boards using C and C++.

- **Position Sensing:** Hapkit uses a magnet placed on the shaft of the motor, a magnetoresistive angle sensor (the KMA 210 from NXP), and the Hapkit Board’s ADC to sense rotations of the motor. This combination suffers from inaccuracies ($< 1^\circ$) in the angular rotation measurement due to temperature drift and hysteresis in the magnetoresistive sensor, and is sensitive to noise and capacitive coupling in the ADC channels. The inaccuracies in angular rotation sensing due to these errors translate to sub-millimeter errors in the position of the handle. These errors are not a significant issue for a 1-DOF device because a user will not notice the effects on rendered forces due to sub-millimeter changes in the position of a 1-DOF rendered virtual environment. We originally attempted to use this configuration with an initial prototype of Haplink that was built with Hapkit’s sensing solution [64]. However, we encountered several issues trying to adapt this method to the higher-DOF system. Capacitive coupling in the ADC channels made the position sensing untrustworthy; a change in measured position from one sensor affected the sensed position of the other. In addition, the sensor hysteresis and temperature drift, which manifested as inaccuracies in the sensed position, were unpredictable and noticeable in the rendered output. Even though these issues could have been mitigated through filtering, a filtering solution would result in a slower loop rate.

Instead, we replaced Hapkit’s motor and position sensing solution with a Pololu (www.pololu.com) 12 V motor with a 5 V 48 CPR encoder (Figure 3.4). We chose an integrated solution instead of obtaining the motor and encoder separately in order to minimize the number of parts that students have to procure. This Pololu motor-with-encoder uses the same Mabuchi motor used in Hapkit and integrates it with a 48 CPR encoder, and it can be obtained for \$21.95. (For comparison, the combination of the Hapkit motor and the MR sensing solution, which includes the MR sensor and a magnet, is \$11.) Although this solution

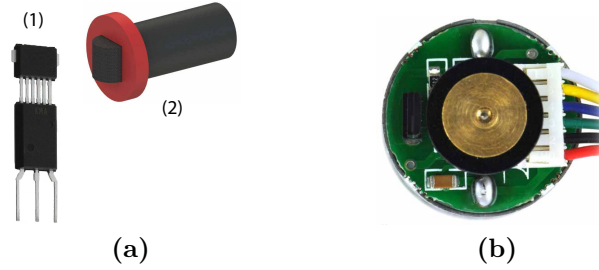


Figure 3.4: Hapkit and Haplink position sensing solutions: **(a)** Hapkit uses the KMA210 magnetic angle sensor from NXP (1) combined with a cylindrical magnet attached to the shaft of the motor (2). This combination is then capable of sensing the rotations of the motor from 0° to 180° . **(b)** Haplink uses a 48 counts-per-revolution encoder attached to a Mabuchi 12 V motor. This motor with encoder is sold by Pololu.

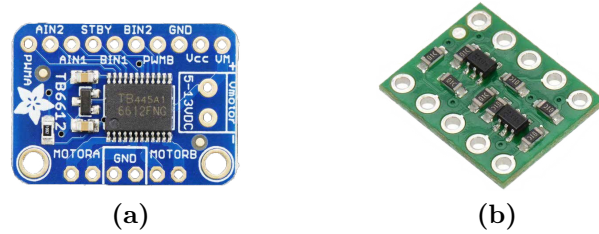


Figure 3.5: Haplink's additional electronics: **(a)** A TB6612 Adafruit Motor Driver is used to drive both motors. **(b)** A 4-channel bi-directional logic level shifter from Pololu is used to connect the 5 V encoder signals to the Nucleo's 3 V channels.

also has poor resolution of position sensing, it does not have the unpredictable effects caused by drift, hysteresis, and noise. The capabilities of Haplink's position sensing solution will be further discussed in Section 3.3.

- **Additional Electronics:** A Pololu 4-channel logic level shifter is used to allow the microcontroller (STM32F446ZE, which works with 3.3 V logic) to read the signal coming from the position sensor (5 V encoder). The Pololu 4-channel logic level shifter costs \$2.50. Additionally, Haplink uses a TB6612 dual DC motor driver from Adafruit (which costs \$4.95) to drive the motors. Figure 3.5 shows Haplink's additional electronic components.

Hapkit's electronics solution (Hapkit Board, magnet, magnetoresistive sensor, Mabuchi motor) costs about \$45 and requires minimal integration effort. (A user only needs

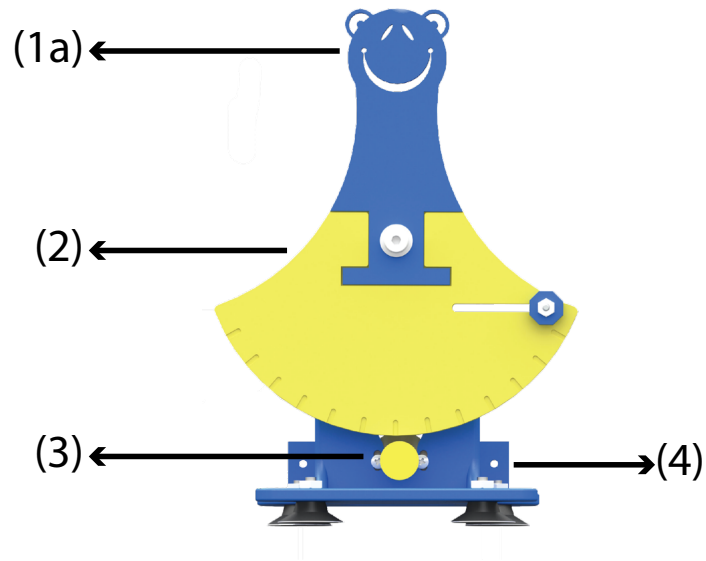
to solder the MR sensor to Hapkit Board.) The total cost of Haplink’s electronics, including the additional parts that must be purchased and integrated ((Nucleo board, 2 Pololu motors with encoders, logic level shifter, motor driver)), is \$70. Haplink requires a user to solder headers and connect the motor driver and level shifter to the microcontroller by using jumper wires or a protoboard. However, because all the electronics (microcontroller, motor drivers, logic level shifter, and encoders) are sold as integrated packages with resistors and protective circuitry against electrostatic discharges, the connection process is straightforward for an inexperienced user.

Structural Components

One of the main considerations in Haplink’s design is the transformation from a 1-DOF device to a 2-DOF device. Figure 3.6 identifies the main components of Haplink. In its 1-DOF form, Haplink is composed of Base 1, Paddle A, and Motor A. In its 2-DOF form, Base 3 supports 1-DOF Haplink on its side. The addition of Base 2, Supports 1 and 2, Motor B, Paddle B, and Haplink Handle creates the second degree of freedom. The 1-DOF version of Haplink is based on Hapkit with the following modifications (Figure 3.7):

- **Sector range of motion:** In order to increase Haplink’s workspace area, Haplink’s Sectors range of motion changed from 90° to 120° compared to previous Hapkit designs, by increasing the arc length of the Sector at the cable interface.
- **Base 1:** In order to transform Haplink from a 1-DOF to 2-DOF device, the base of the 1-DOF device (Base 1) required several changes. First, the attachment of the Sector to the base changed from a screw that taps into the plastic to a shoulder screw with a pressed-in flange bearing and a nut to retain the shoulder screw. This change was necessary because after several cycles of removing and installing the shoulder screw, the tapped material at the base would degrade, causing a loose pivot point in Haplink’s Sectors which allowed for the Sectors to pitch and roll. Second, the hole in Base 1 increased in size so the motor could be replaced without having to remove the motor pulley from the motor shaft (thereby breaking a glued connection). Third, the footprint of Base 1 changed in

Haplink 1-DOF



Haplink 2-DOF

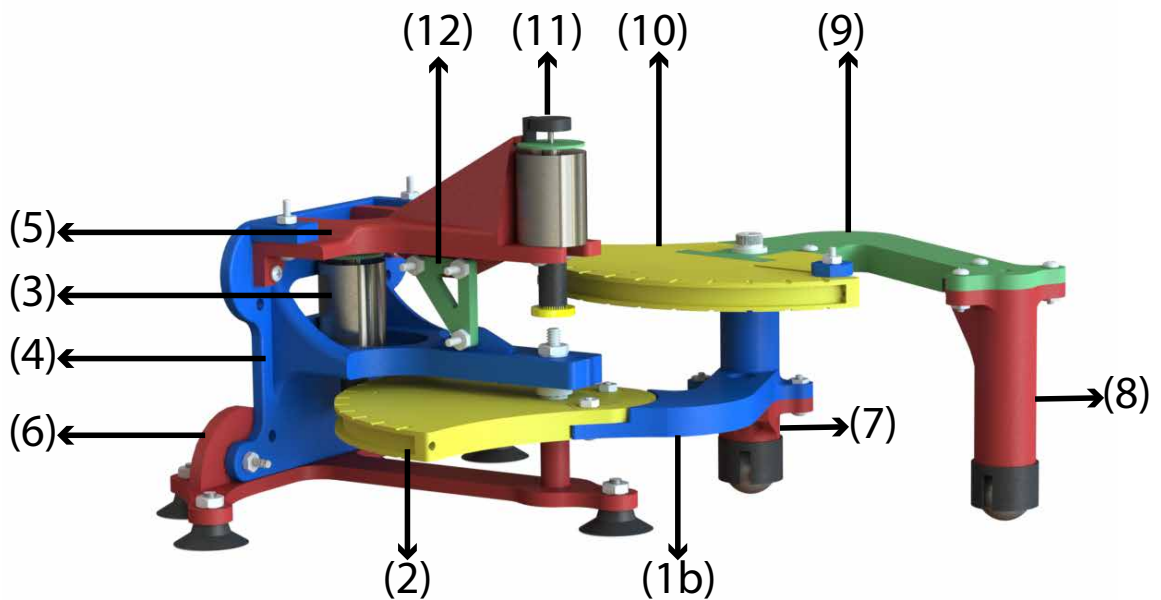


Figure 3.6: Haplink's structural components for both 1-DOF and 2-DOF assemblies: (1) Handle A: (1a) Customized Handle A for Haplink's 1-DOF configuration. (1b) Handle A for Haplink's 2-DOF configuration. (2) Sector A. (1-2) Paddle A. (3) Motor A. (4) Base 1. (5) Base 2. (6) Base 3. (7) Support 1. (8) Haplink Handle. (9) Handle B. (10) Sector B. (9-10) Paddle B. (11) Motor B. (12) Support 2.

order to reduce the size of the final device, while maintaining rounded features to prevent warping during 3D printing. Finally, Base 1 has new holes and features that enable the attachment of the rest of Haplink's parts.

- **Drive wheel:** Whereas Hapkit's motor has a smooth shaft, Haplink's motor comes from the supplier with a small gear at the end of the shaft. Thus, the end of Haplink's drive wheel that interfaces with the motor fits this gear pattern, and has a diameter of 11 mm. Hapkit's drive wheel has a diameter of 9.4 mm.

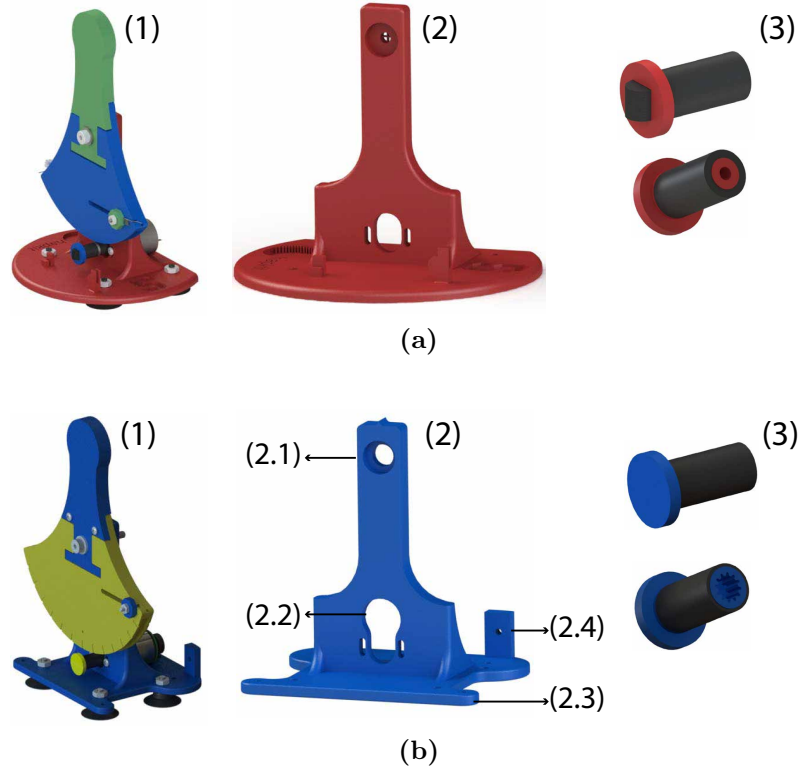


Figure 3.7: Hapkit's structural components compared to Haplink's in its 1-DOF version: **(a)** Hapkit: (1) Assembled Hapkit with a sector that has 90° of rotation. (2) Hapkit's base. (3) Hapkit's drive wheel. **(b)** Haplink: (1) Assembled Haplink in its 1-DOF version with a sector that has 120° of rotation. (2) Haplink base 1 with: (2.1) a place for a bearing and screw to attach Paddle A, (2.2) a hole to facilitate changing the motor, (2.3) a smaller footprint with rounded features, and (2.4) additional features to attach parts in order to transform into a 2-DOF device. (3) Haplink's drive wheel with a star pattern that fits Pololu's 12V MP motor with gearhead.

To enable the transition to the 2-DOF device, we added the following features to the 1-DOF device:

1. **Additional parts:** Haplink is built by combining two paddles in a serial mechanism. To accomplish this several parts (Base 2, Base 3, Triangle Support, Base Support, Haplink Handle) are attached to the 1-DOF version of Haplink. Figure 3.6 shows Haplink's 2-DOF components, and Figure 3.2 illustrates how they are attached.
2. **Calibration features:** The encoders measure changes in position, not absolute position. Thus, it is important that the software knows the starting configuration of the device; i.e. the starting angle offsets (θ_{a0} and θ_{b0}). The software can then compute the position of the device at every time step by adding the change in angle to the starting angle offsets. (This process will be further described in Section 3.4.) The user then makes the starting angle offsets in software equal to the starting angles of the device when it is turned on by aligning the tick marks that were added to the Sectors every 10° to the extruded pointers at the Bases. Section 3.4.1 further discusses this procedure, and Figure 3.16 shows the calibration features.
3. **Grounding of the motors:** In order for Haplink to use inexpensive motors, it was necessary to reduce the load on the motors. One way we accomplished this was by grounding the motors so that the inertia of the second motor would not be carried by the first motor. This mechanism is described in detail in the following section.

Haplink Mechanism

Because Haplink is a teaching tool, it was designed to combine two 1-DOF mechanisms to create the 2-DOF device in a simple manner. This was accomplished using a novel serial linkage mechanism. The aim was for students to learn concepts in 1-DOF and then transfer those concepts to 2-DOF by combining two 1-DOF mechanisms to form the 2-DOF device. However, as mentioned in the previous section, in a typical serial linkage mechanism one motor rotates the inertia of the other motor. In order to

reduce the torque requirements of our motors, we designed a mechanism that grounds both motors by using a capstan drive transmission and grounding Motor B at Paddle A's center of rotation (point c shown in Figures 3.8 and 3.9). With this design, Motor B and Paddle B effectively contact at the point where the capstan cable crosses itself (c_{pe} , defined in Figure 3.9) which always lies along the \hat{a}'_y vector of the \hat{a}' frame (defined in Figures 3.8 and 3.9). c_{pe} can be projected onto physical points on Motor B and Paddle B (c_{pm} and c_{pp}). c_{pm} and c_{pp} will rotate with Motor B and Paddle B. This is illustrated in Figure 3.9, which references the frames defined in Figure 3.8. Due to the cable transmission, and the grounding of Motor B, Paddle B's motion model has two different rolling modes:

- **Rolling Mode 1** (Figure 3.9(a)): Motor B rotates changing θ_{mb} by $\delta\theta_{mb}$, and θ_{ma} stays constant, i.e. $\delta\theta_{ma} = 0$. In this mode frame \hat{a}' and the effective contact point c_{pe} do not move in the inertial reference frame (i.e. $\delta c_{pe} = 0$). Due to the rolling constraint, Paddle B rotates about point p by $\delta\theta_b$, and the instantaneous velocities of c_{pm} and c_{pp} (v_{pm} and v_{pp}) are equal and can be expressed as:

$$v_{pm} = -r_{mb} \dot{\delta\theta}_{mb} \quad (3.1)$$

$$v_{pp} = r_b \dot{\delta\theta}_b \quad (3.2)$$

where r_{mb} and r_b are the radii of Motor B and Paddle B. Combining Equations 3.1 and 3.2 and integrating with respect to time we obtain:

$$\delta\theta_b = -\frac{r_{mb}}{r_b} \delta\theta_{mb} \quad (3.3)$$

- **Rolling Mode 2** (Figure 3.9(b)): Motor B maintains a constant angle θ_{mb} , i.e. $\delta\theta_{mb} = 0$. Motor A rotates and thus θ_{ma} (Figure 3.8) changes by $\delta\theta_{ma}$ (and θ_a changes by $\delta\theta_a$). Due to the rolling constraint, the instantaneous velocities of c_{pm} and c_{pp} are equal as mentioned above. In this case, to simplify the

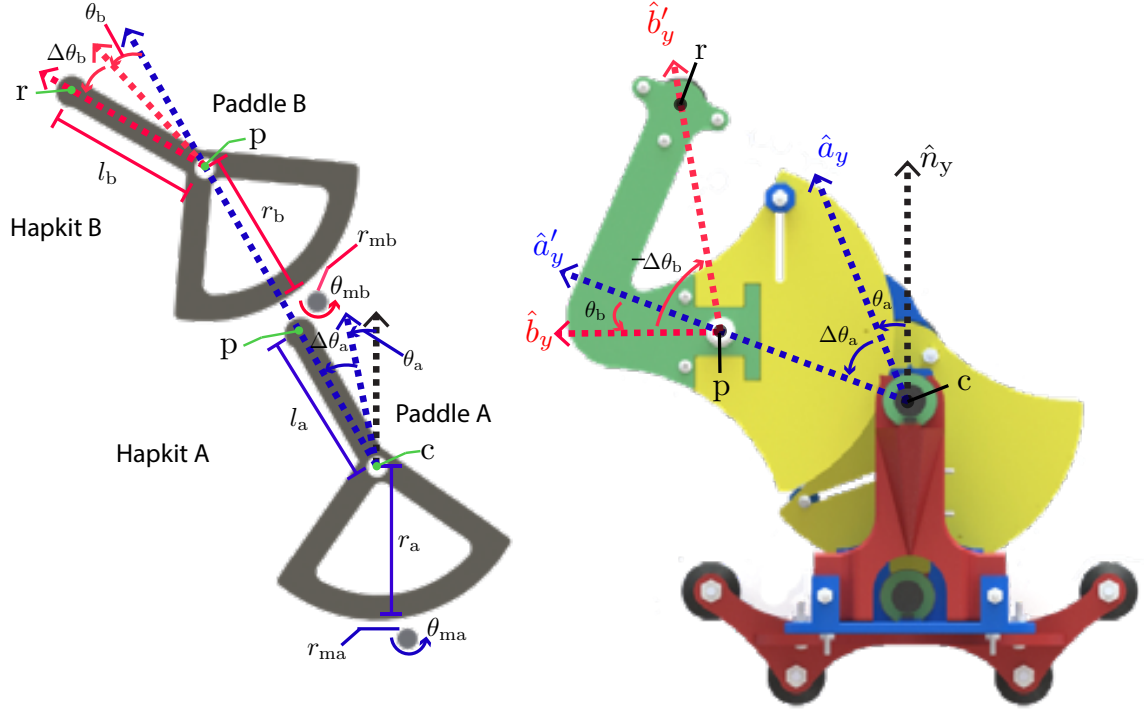


Figure 3.8: Kinematic model of Haplink. c is the origin of the device (0,0 point) and the center of rotation of Paddle A. p is the end effector of Paddle A as well as the center of rotation of Paddle B. r is the end effector of the device. l_a is the distance between the center of rotation of Paddle A and the center of rotation of Paddle B (c and p). l_b is the distance between the center of rotation of Paddle B and the end-effector (p and r). r_a and r_b are the radii of Paddles A and B respectively. r_{ma} and r_{mb} are the radii of motors A and B respectively. θ_{ma} and θ_{mb} are the angles of rotation of motors A and B. θ_a and θ_b are the angles of rotation of Paddles A and B. The frames in the system are defined as follows with their y components depicted in the image: \hat{n} is the inertial frame fixed to ground. Frame \hat{a} rotates about point c and its y component intersects the center point of Sector A. Frame \hat{a}' rotates about point c and its y component intersects point p . Frame \hat{b} rotates about point p and its y component intersects the center point of Sector B. Frame \hat{b}' rotates about point p and its y component intersects point r . $\Delta\theta_a$ and $\Delta\theta_b$ are the initial offset angles of the handles of Paddles A and B, and thus are the constant offsets between \hat{a} and \hat{a}' , and \hat{b} and \hat{b}' . (Adapted from [64] © 2017 IEEE)

representation, we can formulate these velocities relative to frame \hat{a}' [77]. Since θ_b is measured with respect to frame \hat{a}' , v_{cpp} can be expressed as:

$$v_{cpp} = r_b \dot{\delta\theta}_b \quad (3.4)$$

Frame \hat{a}' rotates with respect to the inertial frame with an angular velocity $\dot{\delta\theta}_a$. Since Motor B is fixed in the inertial frame, c_{pm} moves with an angular velocity $-\dot{\delta\theta}_a$ with respect to frame \hat{a}' . v_{cpm} can be expressed as:

$$v_{cpm} = r_{mb} \dot{\delta\theta}_a \quad (3.5)$$

Combining Equations 3.4 and 3.5 we obtain the following:

$$\dot{\delta\theta}_a r_{mb} = \dot{\delta\theta}_b r_b \quad (3.6)$$

Paddle A and Motor A (shown in Figure 3.8) only have one rolling mode equivalent to Paddle B's Rolling Mode 1. Therefore:

$$\delta\theta_a = -\frac{r_{ma}}{r_a} \delta\theta_{ma} \quad (3.7)$$

where r_{ma} is the radius of Motor A and r_a is the radius of Sector A. Combining Equation 3.6 and 3.7 and integrating with respect to time gives a relationship for $\delta\theta_b$ in terms of $\delta\theta_a$ as:

$$\delta\theta_b = -\frac{r_{ma}}{r_a} \frac{r_{mb}}{r_b} \delta\theta_a \quad (3.8)$$

Depending on Haplink's motion, one or both of the modes may be active, i.e. $\delta\theta_b$ is a function of both $\delta\theta_{mb}$ and $\delta\theta_{ma}$:

$$\delta\theta_b = f(\delta\theta_{mb}, \delta\theta_{ma}) \quad (3.9)$$

Combining equations 3.3 and 3.8 we then obtain:

$$\delta\theta_b = -\frac{r_{mb}}{r_b}\delta\theta_{mb} - \frac{r_{mb}}{r_b}\frac{r_{ma}}{r_a}\delta\theta_{ma} \quad (3.10)$$

This mechanism design allows us to model Haplink's kinematics as a serial linkage mechanism that is further described in Section 3.3.1.

3.3 Haplink Analysis

Rendering virtual environments using Haplink requires an analysis of its kinematics, position resolution, and force output capabilities. The following sections describe this analysis.

3.3.1 Forward Kinematics

Haplink can be modeled as a serial linkage mechanism (Figure 3.8) composed of two Hapkits (Hapkit A and Hapkit B) where the center of rotation of the paddle of Hapkit B (Paddle B) is the same point as the end effector of Hapkit A (point p) and the motor of Hapkit B (Motor B) is located in line with the center of rotation of the paddle of Hapkit A (point c). The forward kinematic equations are:

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} -l_a \sin(\tilde{\theta}_a) + c_x \\ l_a \cos(\tilde{\theta}_a) + c_y \end{bmatrix} \quad (3.11)$$

$$\begin{bmatrix} r_x \\ r_y \end{bmatrix} = \begin{bmatrix} -l_b \sin(\tilde{\theta}_a + \tilde{\theta}_b) + p_x \\ l_b \cos(\tilde{\theta}_a + \tilde{\theta}_b) + p_y \end{bmatrix} \quad (3.12)$$

where c is the center of rotation of Paddle A, p is the center of rotation of Paddle B, r is the end effector position, l_a is the distance between the center of rotation of Paddle A and the center of rotation of Paddle B, and l_b is the distance between the center of rotation of Paddle B and the end-effector. These parameters are also shown in Figure 3.8. $\tilde{\theta}_a$ and $\tilde{\theta}_b$ are the angles of Paddle A and Paddle B and are defined as:

$$\tilde{\theta}_a = \theta_a + \Delta\theta_a \quad (3.13)$$

$$\tilde{\theta}_b = \theta_b + \Delta\theta_b \quad (3.14)$$

where θ_a and θ_b are the angles of rotation of Paddles A and B, and $\Delta\theta_a$ and $\Delta\theta_b$ are the initial offset angles of the handles of Paddles A and B (shown in Figure 3.8. Each Hapkit's paddle rotates about its center of rotation (point c for Hapkit A and point p for Hapkit B) by rolling its paddle on its motor without slipping. Hapkit B's Paddle also rotates about point c due to Paddle A's motion. We can relate the angles of rotation of Paddles A and B (θ_a and θ_b) to the angles of rotation of Motors A and B (θ_{ma} and θ_{mb}) using:

$$\theta_a = -\frac{r_{ma}}{r_a}\theta_{ma} \quad (3.15) \quad \theta_b = -\frac{r_{mb}}{r_b}\left(\theta_{mb} + \frac{r_{ma}}{r_a}\theta_{ma}\right) \quad (3.16)$$

where r_a and r_b are the radii of Paddles A and B, and r_{ma} and r_{mb} are the radii of Motors A and B. Equation 3.16 is derived from Equation 3.10 and Equation 3.15 is derived from Equation 3.7.

3.3.2 Differential Kinematics

To relate the force at the handle to the torque at the motors, we obtain the Jacobian matrix by differentiating the relationship between the position of the handle (r_x, r_y) and the rotation at the motors (θ_{ma}, θ_{mb}). I.e. the forward kinematics described in equations 3.12 - 3.16:

$$\begin{bmatrix} \frac{dr_x}{dt} \\ \frac{dr_y}{dt} \end{bmatrix} = \begin{bmatrix} \frac{\partial r_x}{\partial \theta_{ma}} & \frac{\partial r_x}{\partial \theta_{mb}} \\ \frac{\partial r_y}{\partial \theta_{ma}} & \frac{\partial r_y}{\partial \theta_{mb}} \end{bmatrix} \begin{bmatrix} \frac{d\theta_{ma}}{dt} \\ \frac{d\theta_{mb}}{dt} \end{bmatrix} \quad (3.17)$$

where:

$$J = \begin{bmatrix} \frac{\partial r_x}{\partial \theta_{ma}} & \frac{\partial r_x}{\partial \theta_{mb}} \\ \frac{\partial r_y}{\partial \theta_{ma}} & \frac{\partial r_y}{\partial \theta_{mb}} \end{bmatrix} \quad (3.18)$$

$$J = \begin{bmatrix} \frac{r_{ma}}{r_a} \left(l_b \left(1 + \frac{r_{mb}}{r_b} \right) \cos(\tilde{\theta}_a + \tilde{\theta}_b) + l_a \cos(\tilde{\theta}_a) \right) & \frac{r_{mb}}{r_b} l_b \cos(\tilde{\theta}_a + \tilde{\theta}_b) \\ \frac{r_{ma}}{r_a} \left(l_b \left(1 + \frac{r_{mb}}{r_b} \right) \sin(\tilde{\theta}_a + \tilde{\theta}_b) + l_a \sin(\tilde{\theta}_a) \right) & \frac{r_{mb}}{r_b} l_b \sin(\tilde{\theta}_a + \tilde{\theta}_b) \end{bmatrix} \quad (3.19)$$

3.3.3 Haplink Workspace

We designed Haplink's workspace area contain to at least a square of 900 mm² in order for Haplink to render virtual environments such as a box and circle. If the workspace was significantly smaller, it would be hard for students to feel 2-DOF objects. For the device to be physically realizable and usable, the workspace must not overlap with the physical mechanism (i.e., if the workspace is inside of the footprint of the device, Paddle B would run into Motor A). Finally, in order to maintain compatibility with Hapkit 3.0 so that code and applications can be shared for the 1-DOF version of Haplink, the Sector radius was kept the same as Hapkit 3.0. The above requirements were achieved by selecting values for l_a , l_b , $\Delta\theta_a$, and $\Delta\theta_b$ (shown in Figure 3.8).

There are multiple ways to select the undetermined values of Haplink (l_a , l_b , $\Delta\theta_a$, and $\Delta\theta_b$). One approach, investigated here, is to use numerical optimization techniques. To run an optimization, three things are needed: design variables, an objective function, and constraints. The previous paragraph described the constraints (e.g., a 900 mm² box must fit within the workspace), and the design variables (e.g., l_a). We chose the total area of the workspace as an objective function to be maximized and added extra constraints to the link lengths to prevent the optimization algorithm from choosing values that would make the device too large. For example, in order for the device to fit on a desk, the link lengths should not be significantly larger than the radii of the Sector Pulleys (73 mm). Formally, this is written as follows:

$$\begin{aligned}
& \underset{l_a, l_b, \Delta\theta_a, \Delta\theta_b}{\text{maximize}} && A_{\text{workspace}} \\
& \text{subject to} && P_{\text{box}} \subset A_{\text{workspace}} \\
& && 75 \leq l_a \leq 100 \\
& && 75 \leq l_b \leq 100 \\
& && -90 \leq \Delta\theta_a \leq 90 \\
& && -90 \leq \Delta\theta_b \leq 90
\end{aligned} \tag{3.20}$$

where P_{box} is the perimeter of a 30 mm \times 30 mm box centered at a specified point in front of the device to ensure that the workspace and device footprint do not overlap. The center point of the box could also be a design variable in the optimization. We chose to specify it at (-15,125) after running simulations and determining experimentally that, with the established constraints on the link lengths, a workspace containing a 30 mm \times 30 mm box centered at (-15,125) would not overlap with Haplink's mechanism.

A MATLAB [78] simulation was developed to compute the objective function and the constraints for a given set of design variables, which was then coupled to a numerical optimizer to determine the geometric parameters of Haplink. This simulation takes in the design variables and then sweeps θ_a and θ_b from -60° to 60° , finding 1,000,000 points in the workspace. The boundary of the workspace is then found using MATLAB's **boundary()** function, and the area contained inside of that boundary is found using **polyarea()**. In order to calculate the constraints, multiple points along the perimeter of P_{box} were checked to make sure they were inside of the boundary. These methods are not smooth with respect to the design variables. Thus, a gradient-free method was required; we selected a genetic algorithm. The combination of the gradient-free method and simulation were very slow, which resulted in impractical optimization times. Thus, a feasible design space was found by recursively using the optimization to find feasible directions followed by hand tuning the various parameters. Our final design converged on $l_a = 87$ mm, $l_b = 100$ mm, $\Delta\theta_a = 50^\circ$ and $\Delta\theta_b = -80^\circ$. The dimensions of the final workspace achieved with that configuration are given in Figure 3.10.

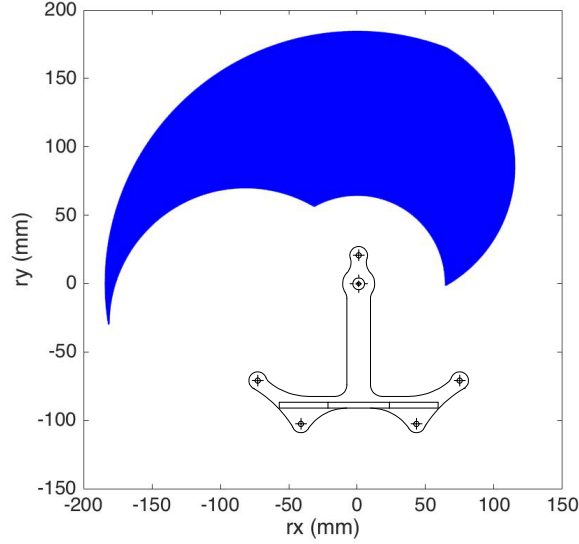


Figure 3.10: Haplink footprint and workspace with $l_a = 87$ mm, $l_b = 100$ mm, $\Delta\theta_a = 50^\circ$ and $\Delta\theta_b = -80^\circ$.

3.3.4 Inverse Kinematics

The inverse kinematics of Haplink give θ_a and θ_b as a function of the coordinates of the end effector, r_x and r_y . Mapping end-effector position to angle position is important when analyzing the effects that the angle resolution will have on rendering virtual environments in different areas of the workspace [79]. We perform this analysis here for desired trajectories in r_x and r_y . The inverse kinematics of the device were derived using basic trigonometry. Using these methods we obtain two solutions for configurations of the device. However, due to the physical constraints of the device, one of these configurations is infeasible. The resulting inverse kinematic equations are:

$$\theta_b = -\text{acos}\left(\frac{r_x^2 + r_y^2 - l_a - l_b^2}{2l_al_b}\right) \quad (3.21)$$

$$\theta_a = \text{atan}\left(-\frac{r_x}{r_y}\right) + \text{atan}\left(\frac{l_b \sin(-\theta_b)}{l_a + l_b \cos(-\theta_b)}\right) \quad (3.22)$$

3.3.5 Kinematic Conditioning

Using the 2-norm of the Jacobian, we can find an upper bound on the scaling from our encoder's nominal resolution to the nominal resolution of the position of the handle since:

$$\|\delta r\|_2 \leq \|J\|_2 \|\delta \theta\|_2 \quad (3.23)$$

Figure 3.11 shows the condition number of the Jacobian matrix throughout Haplink's workspace with $l_a = 87$ mm, $l_b = 100$ mm, $\Delta\theta_a = 50^\circ$ and $\Delta\theta_b = -80^\circ$. The Jacobian matrix is well conditioned. Its condition number has a value between 3.38 and 3.39 for most of the workspace reaching a maximum value of 3.40. The value is slightly lower at the edges of the workspace, reaching a minimum of 2.09.

3.3.6 End-Effector Position Resolution

As mentioned in Section 3.2.2, Haplink uses a Pololu 12 V motor with a 48 counts-per-revolution encoder which gives us a resolution of θ_{ma} and θ_{mb} ($\delta\theta_{ma}$ and $\delta\theta_{mb}$) of 7.5° . This angle resolution corresponds to an upper bound of 1.77 mm in position resolution. In order to provide a guideline for rendering different virtual environments, we analyzed the position resolution of Haplink throughout its workspace.

Knowing the minimum change in angle that we can measure, we can calculate the resolution of the end-effector position r throughout Haplink's workspace. We calculated the change in r_x and r_y (δr_x and δr_y) for 1442401 points in Haplink's workspace when changing θ_{ma} and θ_{mb} by $\delta\theta_{ma}$ and $\delta\theta_{mb}$. The results are illustrated in Figure 3.12. Figure 3.12(a) shows δr_x in millimeters at each point in Haplink's workspace with a change in θ_{ma} of $\delta\theta_{ma}$. Figure 3.12(b) shows δr_y in millimeters at each point in Haplink's workspace with a change in θ_{ma} of $\delta\theta_{ma}$. Figure 3.12(c) shows δr_x in millimeters at each point in Haplink's workspace with a change in θ_{mb} of $\delta\theta_{mb}$. Figure 3.12(d) shows δr_y in millimeters at each point in Haplink's workspace with a change in θ_{mb} of $\delta\theta_{mb}$. The best resolution value in the r_x and r_y directions is of 1.72^{-6} mm, and the worst resolution value is 1.77 mm. This analysis of the workspace will be further be referenced in Section 3.4 in order to inform the rendering of different

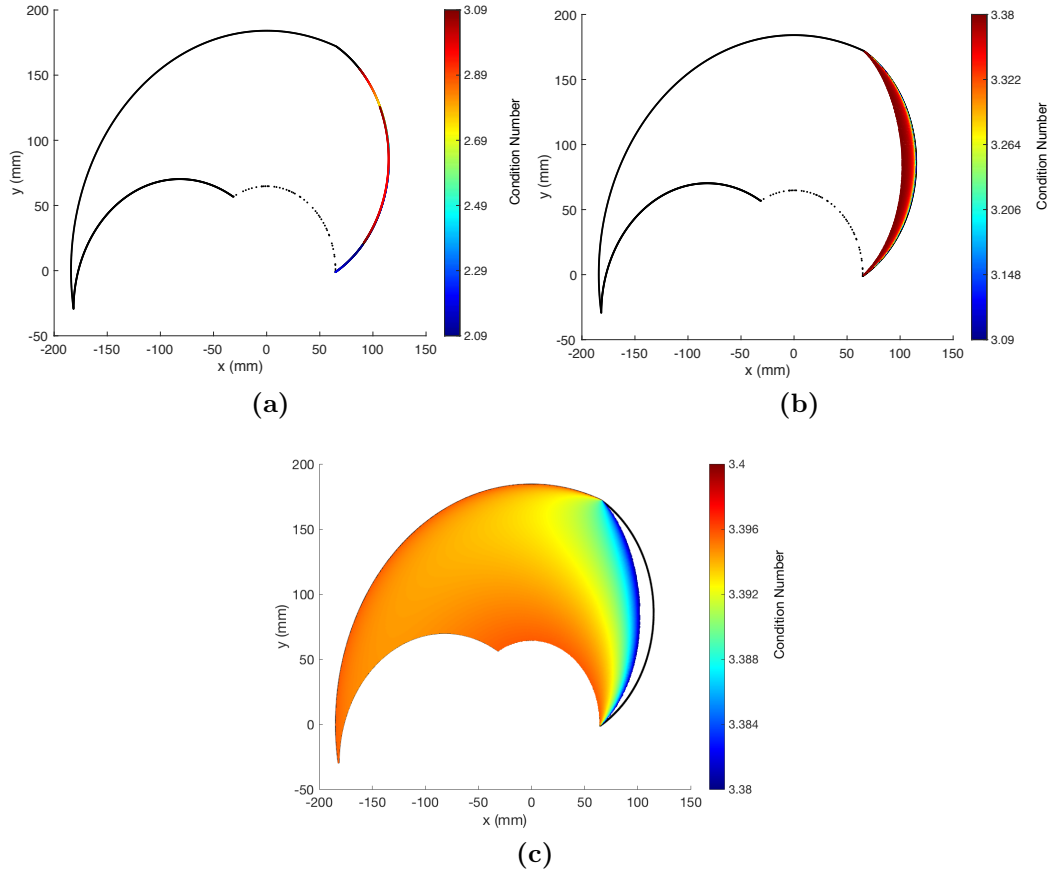


Figure 3.11: Haplink's Jacobian condition number with $l_a = 87$ mm, $l_b = 100$ mm, $\Delta\theta_a = 50$ deg and $\Delta\theta_b = -80$ deg. The workspace is divided into three plots for clarity, with the boundary illustrated in black: **(a)** Shows the area of the workspace where the condition number ranges between 2.09 and 3.09. **(b)** Shows the area of the workspace where the condition number ranges between 3.09 and 3.38. **(c)** Shows the area of the workspace where the condition number ranges between 3.38 and 3.40.

virtual environments.

3.3.7 Drive Electromechanical System Characterization

In order to obtain a transfer function between the PWM duty cycle from Haplink's microcontroller to output torques of the motors, a Nano-17 force sensor was attached to one of the motors and the output torque was measured between 0 and 1.0 duty

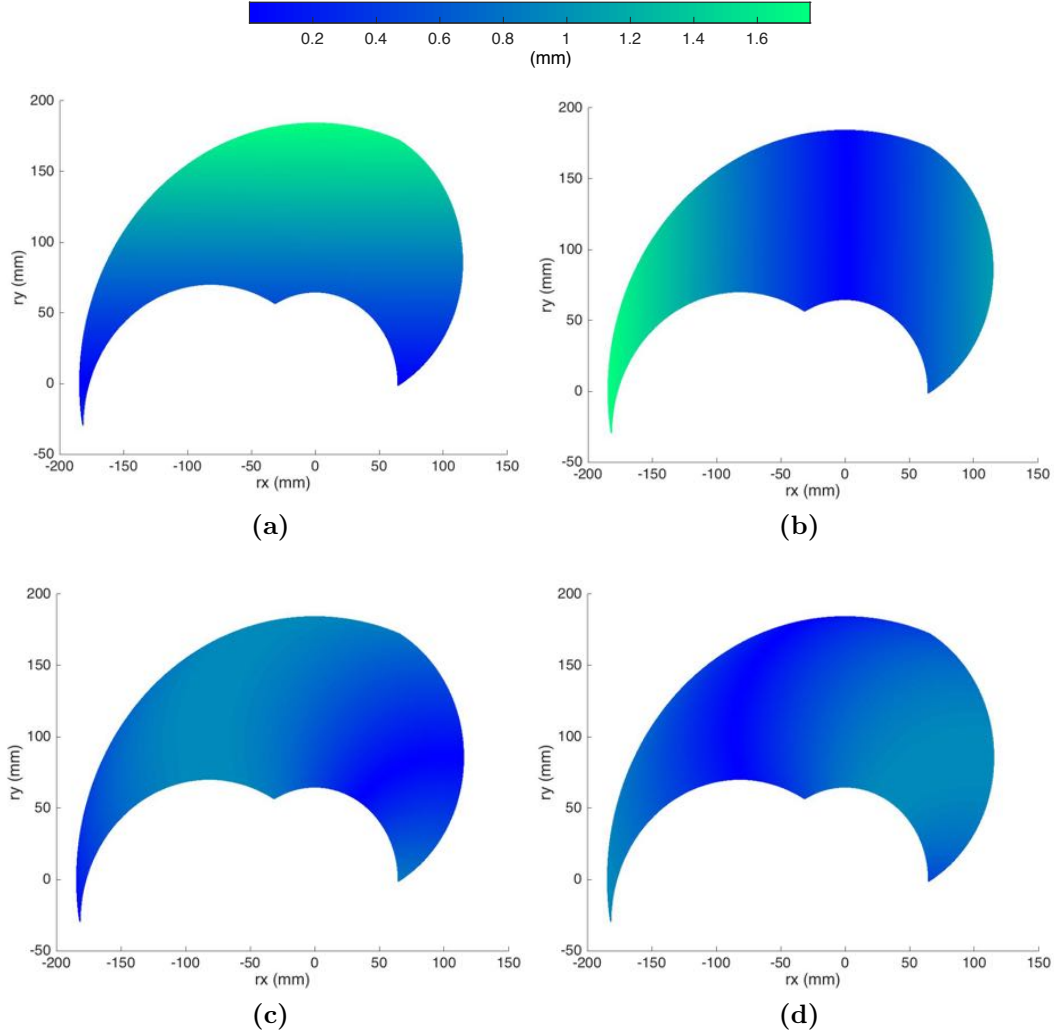


Figure 3.12: Haplink position resolution throughout its workspace: (a) Magnitude of the change in r_x (in mm), given a change of 7.5° in θ_{ma} . (b) Magnitude of the change in r_y (in mm), given a change of 7.5° in θ_{ma} . (c) Magnitude of the change in r_x (in mm), given a change of 7.5° in θ_{mb} . (d) Magnitude of the change in r_y (in mm), given a change of 7.5° in θ_{mb} .

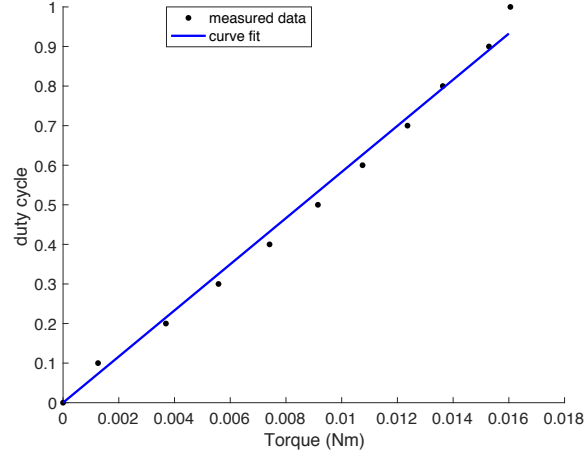


Figure 3.13: We measured the torque output of Haplink’s motor at different duty cycle values and used a linear curve to describe the relationship between the duty cycle and Torque output of Haplink’s motor. The linear fit is given in Equation 3.24.

cycle with 0.1 increments. A linear curve was then fit to the response and the resulting equation is:

$$duty = 58.29 T_m \quad (3.24)$$

where T_m is the desired torque output to the motor. Figure 3.13 shows the measured data as well as the linear fit. The linear fit has an R-squared of 0.99 and an RMSE (root mean square error) of 0.03.

3.3.8 Output Force Analysis

Haplink’s electronics combined with the Pololu MP 12 V motors can output a maximum torque of 0.016 Nm. This was measured using a Nano-17 force sensor. Using the maximum measured torque that the motors can output combined with the kinematic model described in Section 3.3.1, we simulated the maximum output forces that the device can render at various positions (r_x, r_y) . Figure 3.14 shows the maximum simulated force in the x and y directions (F_x and F_y) that can be output in different regions of the workspace.

Haplink’s kinematic model assumes that it has rigid links between all of the components. However, because Haplink is constructed from 3D-printed PLA and uses

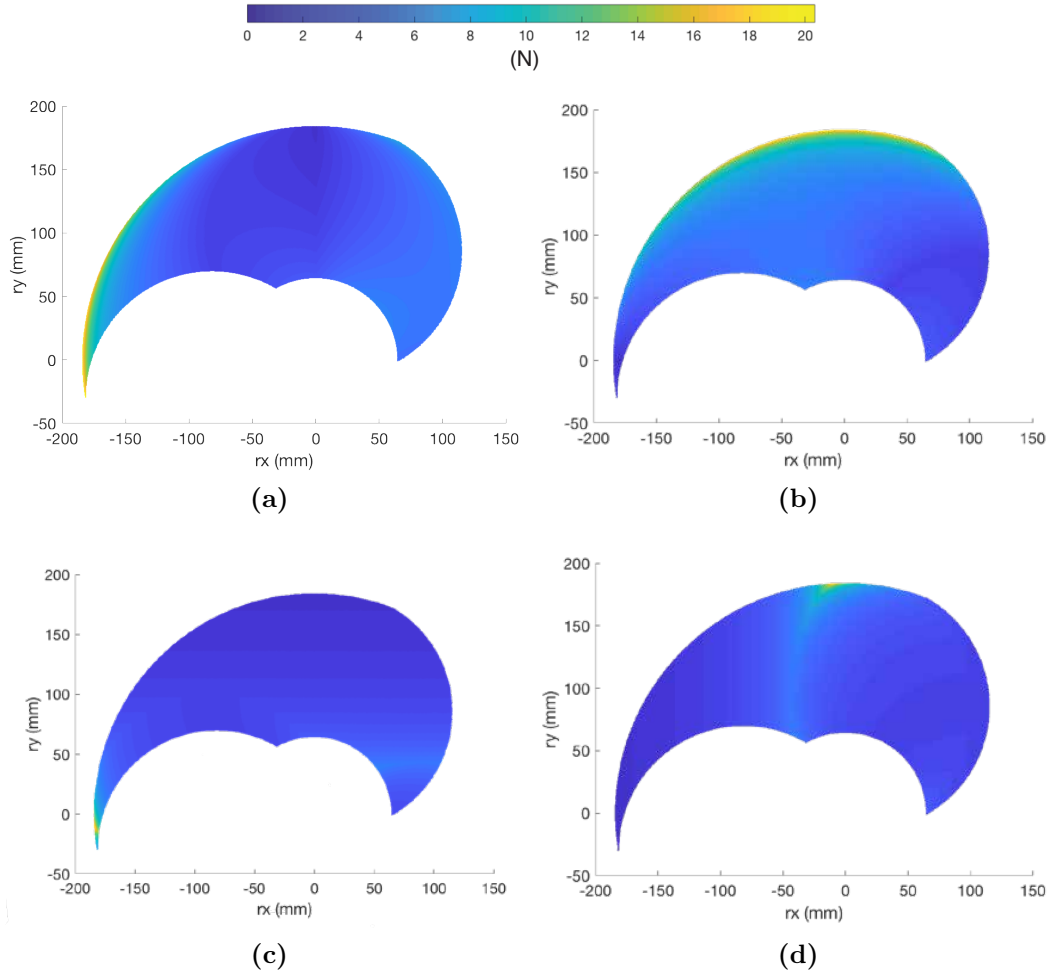


Figure 3.14: Haplink force output throughout its workspace at a given handle location (r_x, r_y) . **(a)** Maximum force in the x direction in Newtons. **(b)** Maximum force in the y direction in Newtons. **(c)** Maximum force in the x direction when constraining the force in the y direction to 0 N. **(d)** Maximum force in the y direction when constraining the force in the x direction to 0 N.

a cable transmission, its connections are not completely rigid. In order to measure the effects of Haplink's structural compliance on the rendered forces, we attached a Nano-17 force sensor between Haplink's handle and ground at 3 different locations (Location 1: $\theta_a = 38^\circ$ and $\theta_b = -14^\circ$; Location 2: $\theta_a = -35^\circ$ and $\theta_b = -33^\circ$; and Location 3: $\theta_a = -23^\circ$ and $\theta_b = 19^\circ$) and measured the output forces at 0.2 N increments from 0.2 N until the simulated maximum force at that location in the x and y

directions. Figure 3.15 shows the result of those experiments.

Rendering forces with Haplink requires the Jacobian (Equation 3.19), which is a function of the angles θ_{ma} and θ_{mb} . Figure 3.15(a) shows that the angle of the force output has errors between 0.6° and 20° at the analyzed locations for force magnitudes between 0.4 and 3 N. These errors are caused by a combination of compliance in the 3D-printed material, cable transmission, misalignment of parts due to manufacturing tolerances on the 3D printer, position resolution limitations in the workspace, and calibration errors. Due to the low resolution of the motor encoders, the measured end-effector position of Haplink can vary up to 1.77 mm from the actual position as discussed in Section 3.3.6. In addition, the calibration procedure described in Section 3.4.1 is prone to human error as it requires the alignment of two features by eye. Each system has a unique offset because these alignment features are susceptible to manufacturing tolerances. This error in angle can also be seen reflected in the measured force magnitude (Figure 3.15(b)). Lower force magnitudes (<0.4 N) which correspond to torques in one or both motors below 0.002 Nm depending on the location of the end-effector (i.e. in Location 3, a force of 0.2 N in the y direction corresponds to a torque of 0.0002 Nm in Motor A and 0.0008 Nm in Motor B) are more prone to error. This is because the relationship between the duty cycle and forces is nonlinear at low motor torques and is not captured in equation 3.24. These errors combined result in errors in direction and force magnitude.

3.4 Software and Control

Haplink is an impedance-type device controlled using a Nucleo-F446ZE microcontroller programmed through the mbed online compiler. Haplink's software reads the current values of θ_{ma} and θ_{mb} using the encoders on its motors, transforms the values to absolute rotation of the motors (Section 3.4.1), and then translates those values to the position of the end-effector (r_x, r_y) using the direct kinematic mapping of the device (Equations 3.12 - 3.16). The desired vector force F is calculated depending on

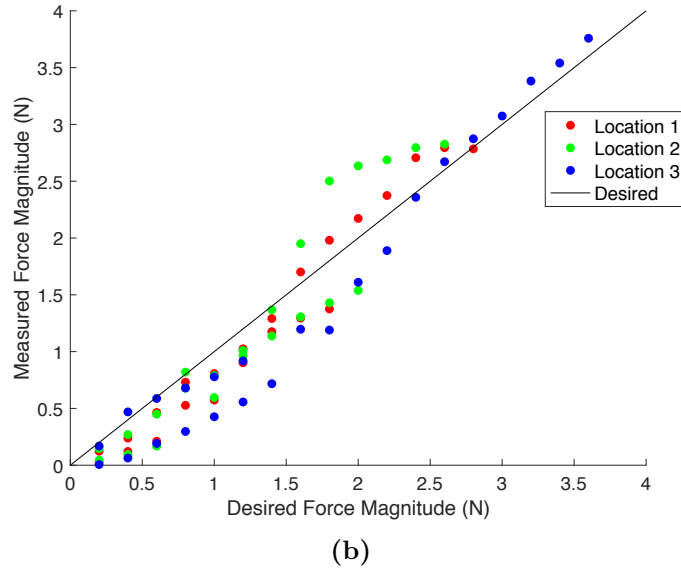
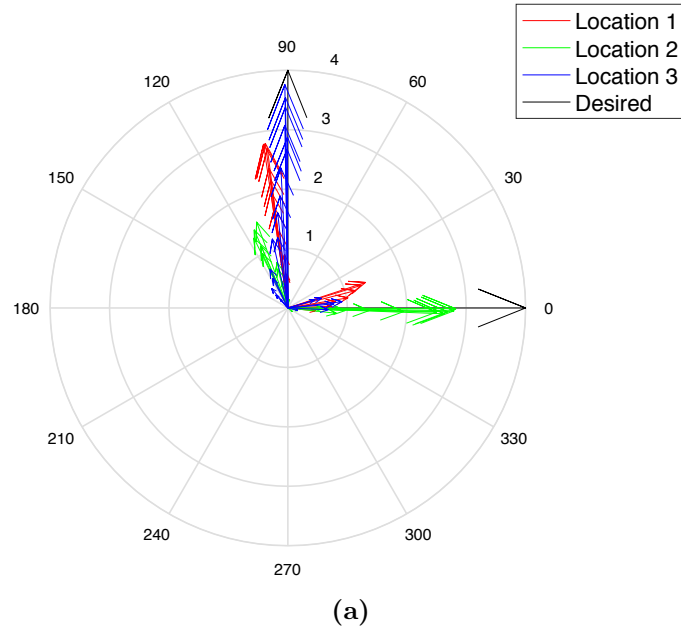


Figure 3.15: Haplink force output in the range 0.2 N to the maximum simulated force at each location, measured at three different locations, plotted against the desired force. Location 1: $\theta_a = 38^\circ$ and $\theta_b = -14^\circ$, Location 2: $\theta_a = -35^\circ$ and $\theta_b = -33^\circ$, and Location 3: $\theta_a = -23^\circ$ and $\theta_b = 19^\circ$. (a) Desired force angle compared to the measured force angle. (b) Desired force magnitude compared to the measured force magnitude.

the virtual environment being rendered. The desired torque at the motors is:

$$T_m = \begin{bmatrix} T_{ma} \\ T_{mb} \end{bmatrix} = J^T F \quad (3.25)$$

where J is the Jacobian matrix described in Equation 3.19. This desired torque is then translated to a pwm duty cycle for the motors using Equation 3.24.

3.4.1 Startup Calibration

To calculate the position of the end-effector, Haplink uses the encoders on the motors to obtain the relative motor angles θ_{ma} and θ_{mb} , and the starting position of the motor angles which can be changed in software during Haplink's startup calibration. This calibration (illustrated in Figure 3.16) needs to be performed before running Haplink software. It uses tick marks on Haplink's sectors which mark the positions of θ_a and θ_b respectively as well as arrows marked on the base of Haplink that should be centered on those tick marks. The calibration procedure consists of the following steps:

1. **Step 1:** Make sure both Paddles (A and B) are centered and at 0° starting angle. Align the markings on the bottom of Paddle B with the arrow on Base 2 so that the starting θ_b angle in software matches with the physical starting angle.
2. **Step 2:** Align the markings on the top of Paddle A with the arrow on Base 1 so that the starting θ_a angle in software matches with the physical starting angle.

The starting angles should be set in hardware and software depending on the area of the workspace being used. Section 3.4.2 will discuss the methods and criteria of finding the appropriate workspace area for different use cases.

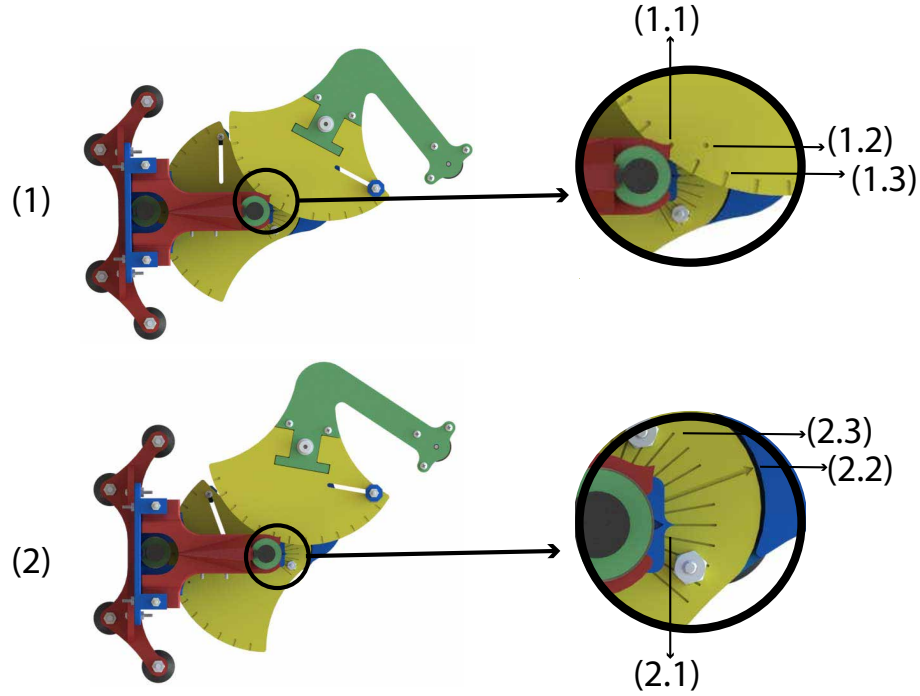


Figure 3.16: Haplink startup calibration procedure: **(1) Step 1:** Starting with both Paddles (A and B) centered at 0° , align the markings on the bottom of Paddle B with the arrow on Base 2. Calibration components: **(1.1)** Base 2 extruded arrow, **(1.2)** 0° mark, and **(1.3)** 10° tick marks. **(2) Step 2:** Align the markings on the top of Paddle A with the arrow on Base 1. Calibration components: **(2.1)** Base 1 extruded arrow, **(2.2)** 0° mark, and **(2.3)** 10° tick marks.

3.4.2 Rendering Virtual Environments

Section 3.3.6 showed the resolution of the end-effector position throughout Haplink's workspace. This section discusses how to analyze the resolution of the device in order to infer the best region to render certain virtual environments and analyzes the rendering capabilities of Haplink in three different areas of the workspace shown in Figure 3.17.

Area 1 (depicted in Figure 3.18) is a 5 mm horizontal cross-section of Haplink's workspace at $y = 100$ mm. Figure 3.18(b) shows the position measured by the device as a user moves along a vertical line at $y = 100$ mm. These errors in position have important implications when rendering a horizontal wall, since the force output is

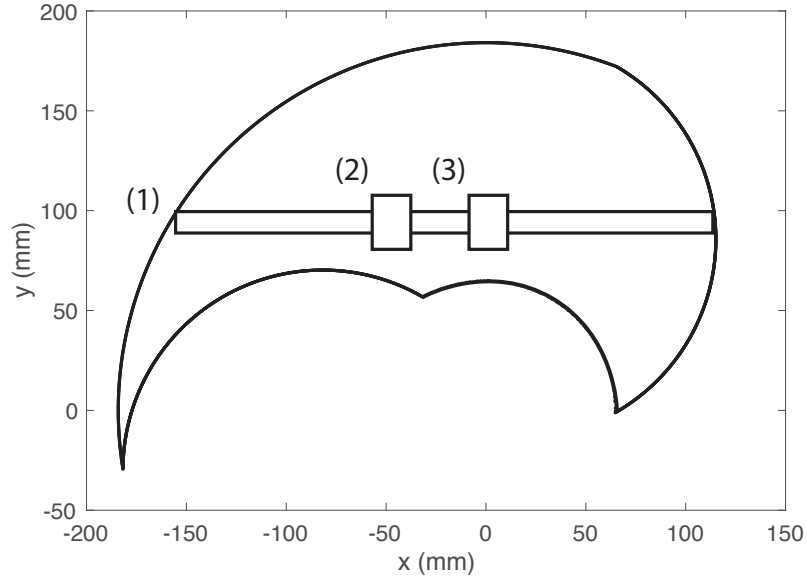


Figure 3.17: Three areas in Haplink's workspace where the rendering capabilities of the device are analyzed.

proportional to the position measured and the stiffness of that wall. Figure 3.18(c) shows a slab of 5 mm in height in r_y of the forces output by the device when rendering a wall of stiffness $k = 0.1$ N/mm at $r_y = 100$ mm. The errors in position measurement result in specific shapes in the forces rendered in specific regions of the workspace. Area 3 in the workspace (shown in Figures 3.17 and 3.18(a)) is a region where motion in r_x requires a change in motion in θ_b that is lower than the encoder resolution. As a result, the measured trajectory of the user is solely dependent on the measured θ_a position, resulting in an arc-like shape as can be seen in Figure 3.18(b). This results in the wall feeling arc-like as the reaction forces initially decrease and then increase as the device travels across this region. And, when a change in θ_b is detected, measured r_y jumps sharply, which disrupts the circular region on a horizontal trajectory as shown in Figure 3.18(b). This action results in the steps in reaction creating the boundaries of wave-like pattern that can be seen in Figure 3.18(c). Area 2 on the other hand, presents with similar resolution for both θ_a and θ_b , that result in a resolution of 0.5 mm in r_y . This low resolution will result in the virtual wall having a textured feeling, the amplitude of which will be proportional to the stiffness being rendered.

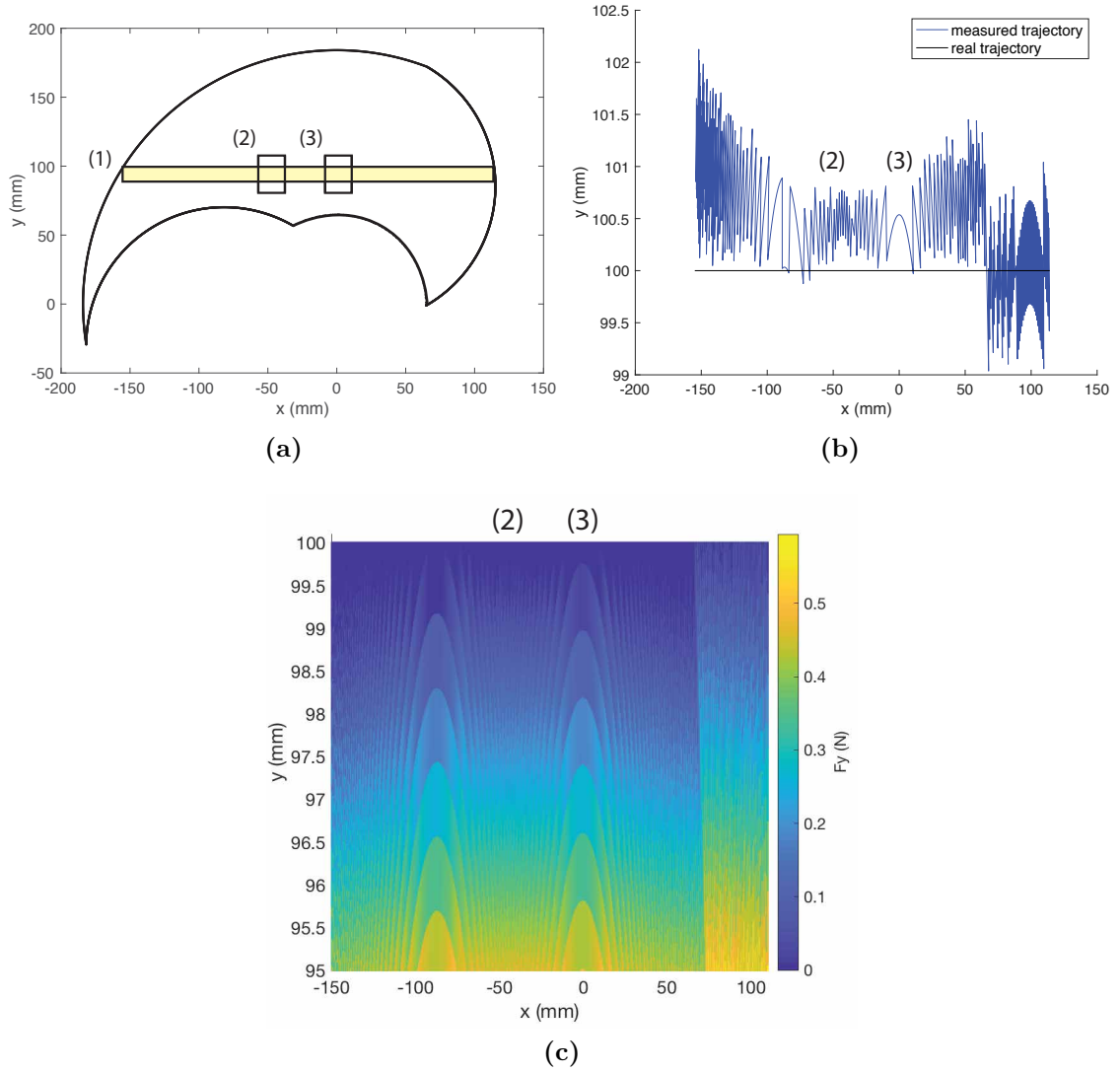


Figure 3.18: Rendering a virtual wall horizontally across Haplink's workspace (Area 1). (a) 5 mm cross-section of Haplink's workspace at $y = 100$ mm (b) Real and measured vertical trajectory at $y = 100$ mm. (c) Output force 5 mm deep into a virtual wall of stiffness $k = 0.1$ N/mm rendered in Area 1 at $y = 100$ mm.

Figures 3.19(a) and 3.19(b), and Figures 3.20(a) and 3.20(b) show the result of rendering inside of box of 20 mm in length in Areas 2 and 3. Rendering in Area 3 results in smoother top and bottom walls of the box, which will be circular in shape. A rendered box in Area 2 does not have curved walls, but the surfaces are less smooth than those rendered in Area 3. Figures 3.19(c) and 3.19(d), and Figures 3.20(c) and 3.20(d) show the result of rendering inside of a circle of 30 mm in diameter in Areas 2 and 3. As can be seen in the figures, the effects discussed above are not as pronounced when rendering circular trajectories.

Another important aspect of rendering virtual environments is the maximum force output that the device can render. Haplink has a maximum force output of approximately 2 N in x and y for most of its workspace, as seen in Figure 3.14 and discussed in Section 3.3.8. However, at the edges Haplink is capable of outputting very high forces in y and is not capable of outputting high forces in x (e.g. only 1.2 N in $x = 50$ mm, $y = 161$ mm). When rendering virtual environments, it is important to be aware of these limitations and either limit the force output of the device or avoid regions in the workspace where there is variation in Haplink's force rendering capabilities for different directions.

In the next section we will discuss how we used Haplink to teach an undergraduate freshman seminar on haptics.

3.5 Haplink Use in the Classroom

In the Fall quarter of 2017, Professor Allison Okamura taught an offering of the Introductory Seminar course on haptics for freshman undergraduate students (ME20N) described in Section 2.5.3, and included Haplink in the curriculum. This section describes the course and provides a discussion of our findings from using Haplink to teach ME20N.

Learning Objectives

The goals of the 2017 offering of ME20N are very similar to those described in Section 2.5.3, with the addition of learning about the Jacobian and how to render 2-DOF

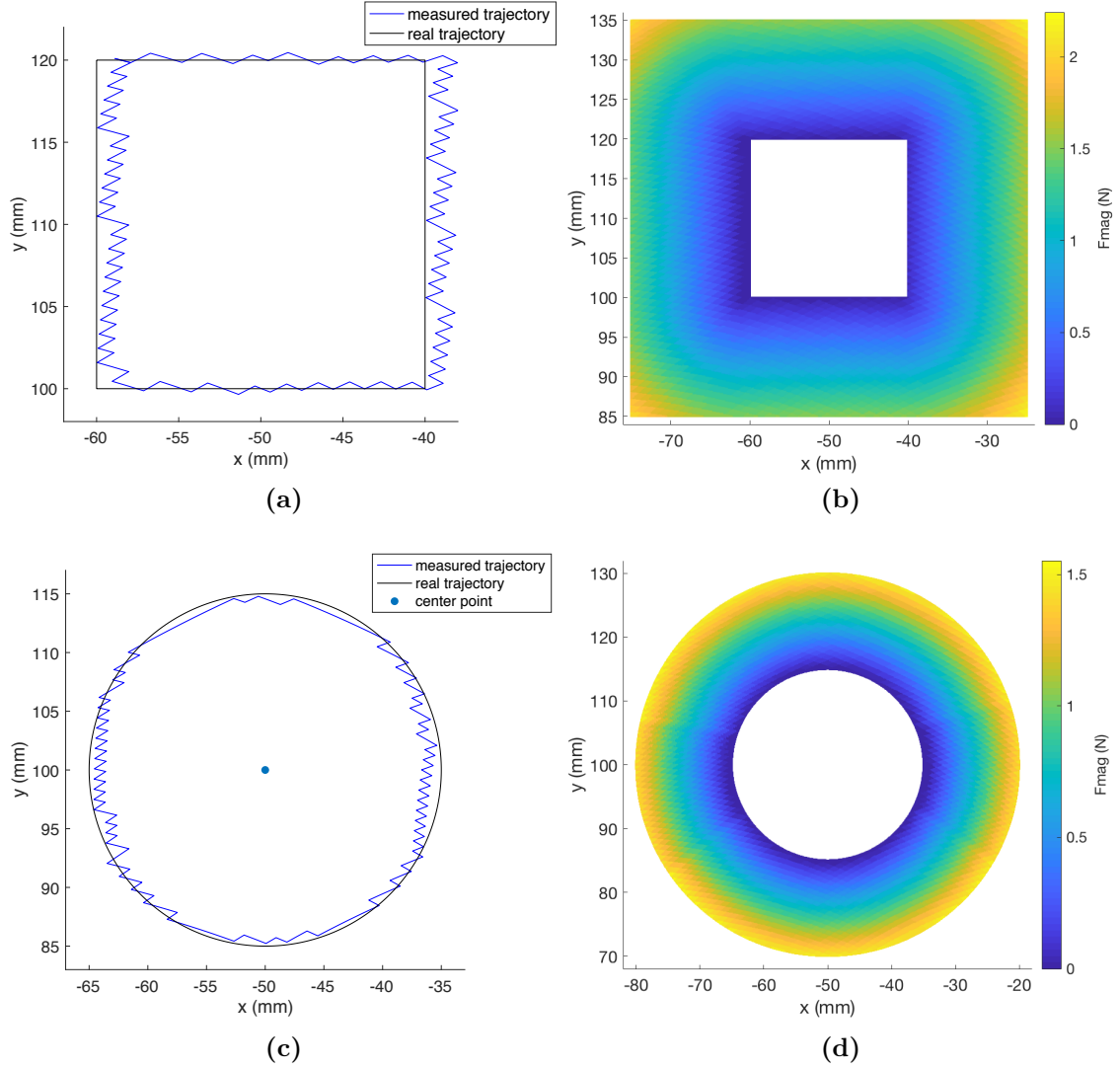


Figure 3.19: Rendering an inside of a box and inside of a circle virtual environments in Area 3 of Haplink’s workspace centered at $x = -50$ mm and $y = 100$ mm. **(a)** Real and measured rectangular trajectory of the box. **(b)** Output force of an inside of a box virtual environment of stiffness $k = 0.1$ N/mm. **(c)** Real and measured circular trajectory of the circle. **(d)** Output force inside of a circle virtual environment of stiffness $k = 0.1$ N/mm.

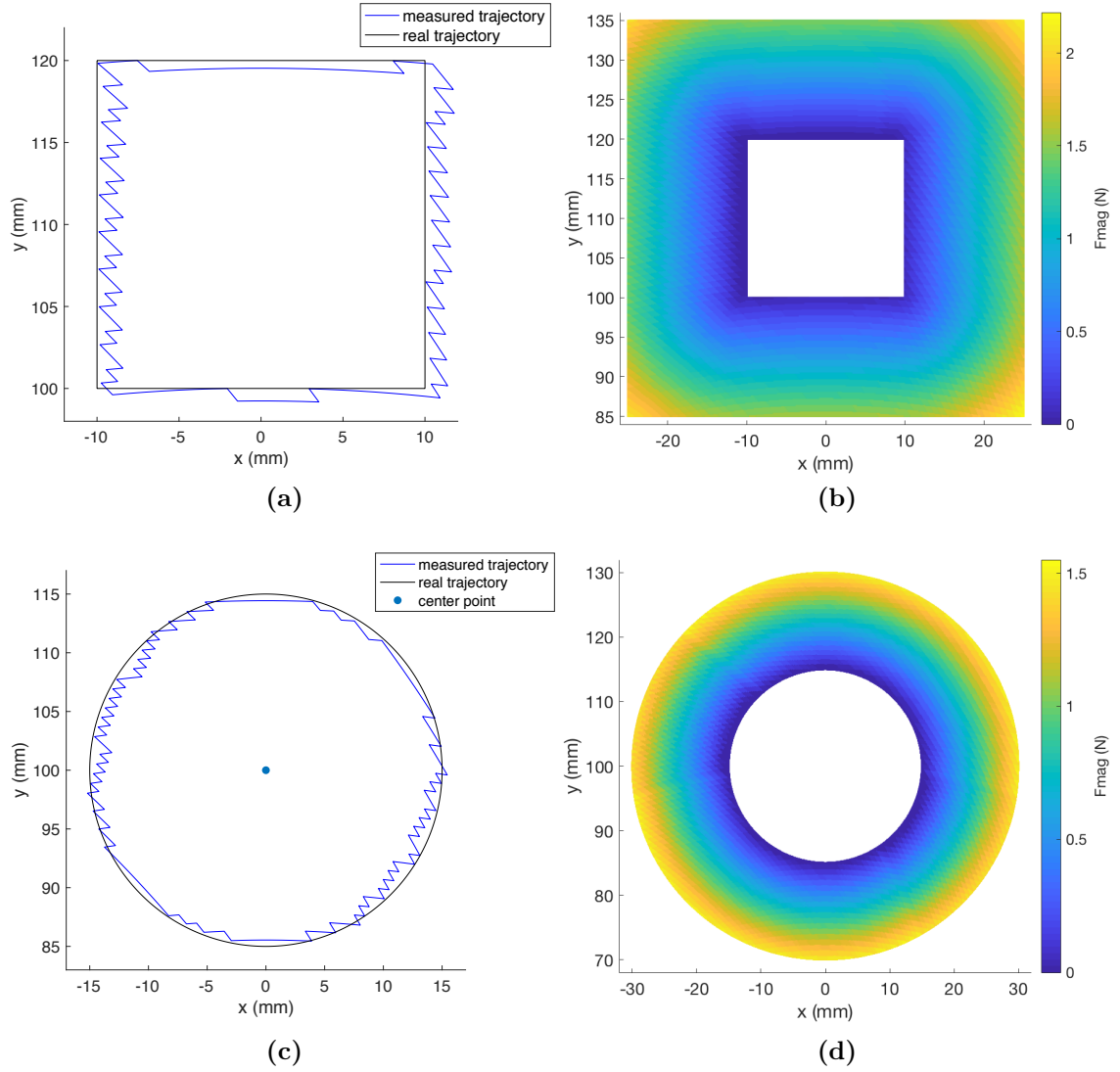


Figure 3.20: Rendering an inside of a box and inside of a circle virtual environments in Area 3 of Haplink's workspace centered at $x = 0$ mm and $y = 100$ mm. **(a)** Real and measured rectangular trajectory of the box. **(b)** Output force of an inside of a box virtual environment of stiffness $k = 0.1$ N/mm. **(c)** Real and measured circular trajectory of the circle. **(d)** Output force inside of a circle virtual environment of stiffness $k = 0.1$ N/mm.

virtual environments. By the end of this course students should be able to:

- Identify the primary mechanisms of human haptic sensing, as well as the capabilities and limitations of human touch.
- Describe the salient features of a haptic device design and the physics of a haptic mechanism.
- Understand methods for sensing the position of and actuating haptic interfaces.
- Assemble and program a 2-DOF haptic device to create compelling touchable virtual environments.
- Explain current and invent potential future applications of haptic devices.
- Design and build a new haptic device.

Course Content

The course content was modified from previous offerings in order to incorporate 2-DOF concepts. Class time was divided into lecture time and laboratory time where students build, program, experiment and discuss the material together with the instructors. The last three weeks of the quarter were spent working on a project in which students built their own haptic applications. Table 3.1 outlines the course's syllabus.

Results and Discussion:

In the 2013 and 2014 offerings of ME20N, the curriculum included an introduction to the field of haptics, basic mechatronic components, rendering of 1-DOF virtual environments using Hapkit, and experimenting with tactile haptics using vibration motors. By including Haplink in the curriculum in 2017, we were able to introduce freshman students to 2-DOF concepts which evolved from the 1-DOF concepts. The course started with a brief introduction to the field of haptics and its importance. It then moved on to describing basic mechatronic concepts used in haptics such as sensors and motors. In the next couple of weeks students were introduced to the 1-DOF version of Haplink. They built the device, designed their own handle, learned

Table 3.1: Modules of the 2017 Freshman Haptics Seminar Course.

Week	Syllabus	Laboratories
Week 1	<ul style="list-style-type: none"> - Introduction to haptics and human touch. - Experiments with human touch - Haptic device design. 	<p><i>Two-point discrimination:</i> Students determine, for a few locations on the body, the distance between two contact points at the threshold of when they are perceived as a single contact point versus two separate contact points.</p> <p><i>Create Haplink 1-DOF handle:</i> Students use Solidworks, customize, and 3-D print, the handle of the 1-DOF version of Haplink.</p>
Week 2	<ul style="list-style-type: none"> - Device kinematics, CAD and 3-D printing. - Product Realization Laboratory introduction 	
Week 3	<ul style="list-style-type: none"> - Device forces. - 1-DOF Haplink assembly. - Control board, sensors, and actuators. 	<p><i>Haplink 1-DOF Assembly:</i> Students assemble their 1-DOF version of Haplink using the handles they created and structural components and tools provided by the instructors.</p>
Week 4	<ul style="list-style-type: none"> - Haplink system sensing - Introduction to programming 	<p><i>Haplink testing:</i> Students learn to use the mbed IDE [76] to compile and write a program to read the output from the encoder sensor and calculate the Handle's position and output a force. They also learn to use the Arduino IDE's serial monitor to print information from Haplink to their computer's screens.</p>
Week 5	<ul style="list-style-type: none"> -Programming 1-D virtual environments 	<p><i>Haptic 1-D Rendering:</i> Students render a virtual spring, wall, damper and texture. They experiment with their devices and also create their own virtual environments.</p>
Week 6	<ul style="list-style-type: none"> - 2-D Haplink kinematics - 2-D Haplink assembly 	<p><i>Haplink assembly and testing:</i> Students transform their 1-DOF Haplinks into 2-DOF Haplinks and test their devices by outputting the position of the handle to the serial monitor.</p>
Week 7	<ul style="list-style-type: none"> -Programming 2-D virtual environments 	<p><i>2-D Virtual environments:</i> Students render inside of a box, outside of a circle, and a 2-D virtual environment of their own creation.</p>
Weeks 8-10	<ul style="list-style-type: none"> -Work on final project 	<p><i>Work on final project:</i> Students work in teams to create novel haptic applications.</p>

its kinematics and force output equations and rendered several virtual environments such as a spring, a wall and a damper. After the students were comfortable with the 1-DOF concepts, they were asked to build the 2-DOF device and taught the kinematics and force rendering equations in two degrees of freedom and how they are related to the 1-DOF concepts. The students then rendered inside of a box and circle and outside of a circle as well as other 2-DOF virtual environments of their own creation. The last two weeks of the course were spent on a class project, in which the students created novel haptic applications by building devices of their own ideation or designing their own customizations of Haplink. Figures 3.21 and 3.22 show two examples of final projects where students chose to build on their Haplink to create a novel haptic device.

In the Gear-Shifter project (shown in Figure 3.21), students used Haplink to mimic the feeling of changing gears in a manual transmission automobile. They customized Haplink's handle by adding a spherical knob to the top in order to resemble the shifter knob on the gear stick. They also 3-D printed a gear guide and created a pedal to act as a clutch. The pedal was equipped with a force sensor that could detect when the pedal was being pressed. The team used Haplink to generate force fields in six different locations of the workspace (gears), which prevented the handle from entering or exiting a gear location until the clutch was engaged after which the handle was free to move around in the guide.

In the Operation project (shown in Figure 3.22), students used Haplink to make their own version of the Operation board game. They created a graphical user interface which showed the silhouette of a person with a heart and a femur. They customized Haplink's handle by adding a vibration motor, a pair of tweezers, and a force sensor. Users could navigate the operation program using Haplink as the cursor and attempt to pick up the heart or femur by squeezing the tweezers, which would compress the force sensor. If the cursor was not located inside of the object's profile when attempting to pick it up, the vibration motor would engage. The students also added converging and diverging force fields around the objects that could be turned on and off in order to change the difficulty of the game.

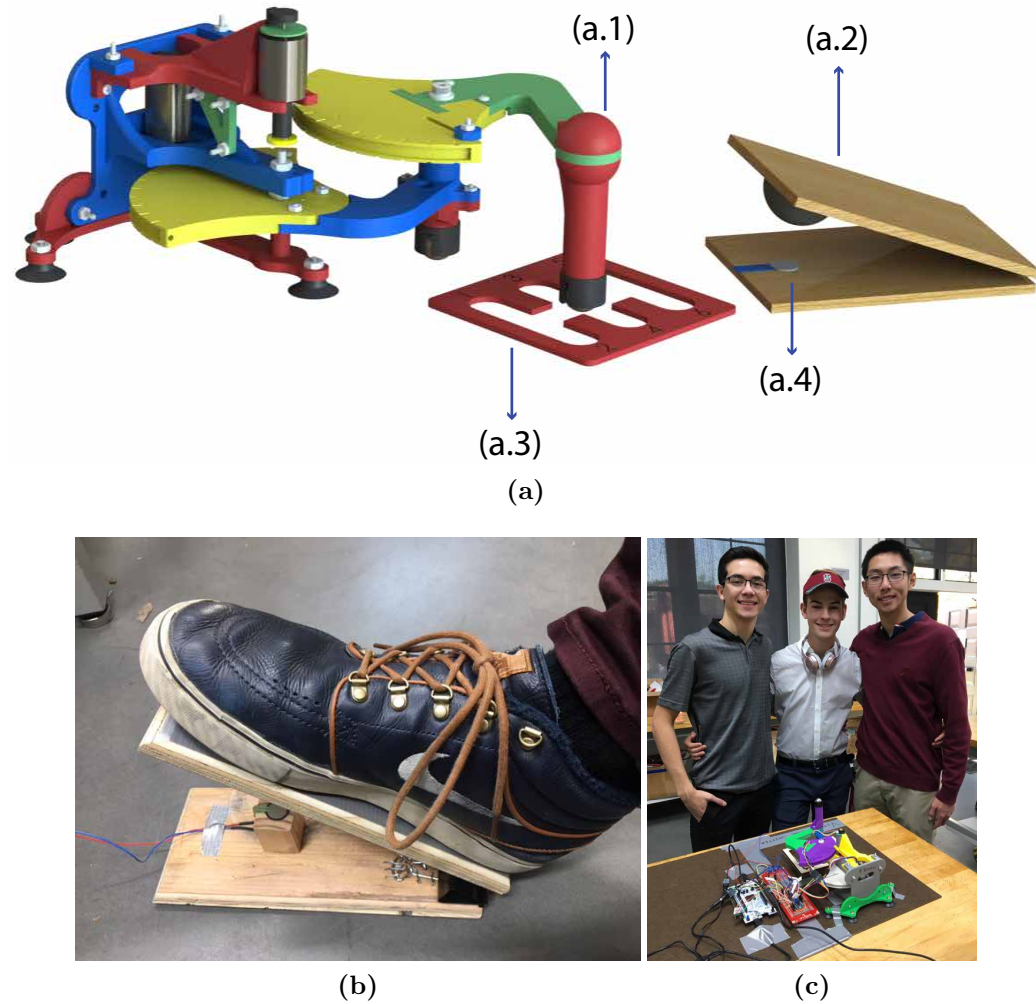
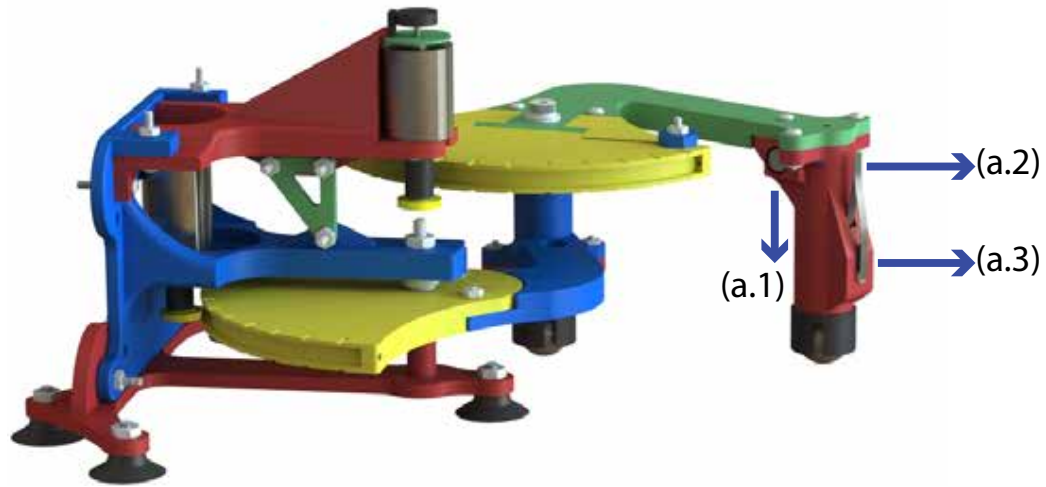
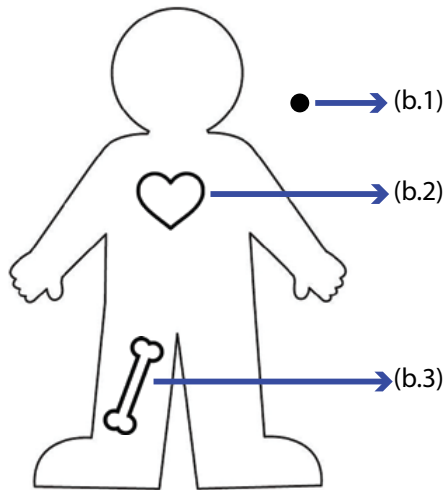


Figure 3.21: Students in ME20N created a Gear-shifter using Haplink. They customized Haplink's handle to act as a gear shifter and added a pedal with a force sensor (clutch) and a guide. The students then programmed Haplink with a force-field which impedes users from moving the handle from the gear unless you were pressing the clutch. (a) Customized Haplink: (a.1) Customized handle. (a.2) Pedal. (a.3) Gear guide. (a.4) Force Sensor. (b) Student using pedal. (c) Gear-Shifter student team.



(a)



(b)



(c)

Figure 3.22: Students in ME20N created a game of Operation using Haplink. They customized Haplink's handle and incorporated a vibration motor, tweezers and a force sensor to it. They also created a software interface which mimicked the classic Operation game. Users would then navigate in the software using Haplink and grab organs (a bone or a heart) using the tweezers. A vibration would let them know if they missed the organ. Students also used the force feedback to make the game easier or harder. One could turn a converging force field in the game to make it easier, or a diverging one to make the game more challenging. (a) Haplink customization: (a.1) Vibration motor. (a.2) Tweezers (a.3) Force sensor. (b) Software interface: (b.1) Cursor. (b.2) Heart. (b.3) Bone. (c) Operation student team.

3.6 Conclusions

In this chapter we presented a kinesthetic haptic device that uses a novel mechanism with serial kinematics and can transform between a 1-DOF and 2-DOF device. We also presented an analysis of its force rendering capabilities as well as its resolution. We then used this analysis to inform on Haplink's rendering capabilities and discussed how to render virtual environments using the device. Haplink's use in an undergraduate seminar on haptics was also discussed. Students in the class were able to learn 1-DOF and 2-DOF rendering of virtual environments as well as further customize their devices to generate virtual environments of their own creation.

In the next chapter, we shift our focus from open-source kinesthetic educational haptic devices to tactile haptic devices for educational applications. We explore the design of a tactile haptic device that is meant to be used by first graders to develop finger perception as they perform math activities.

Chapter 4

HapCaps: Finger Sense Training via Haptics and Movement

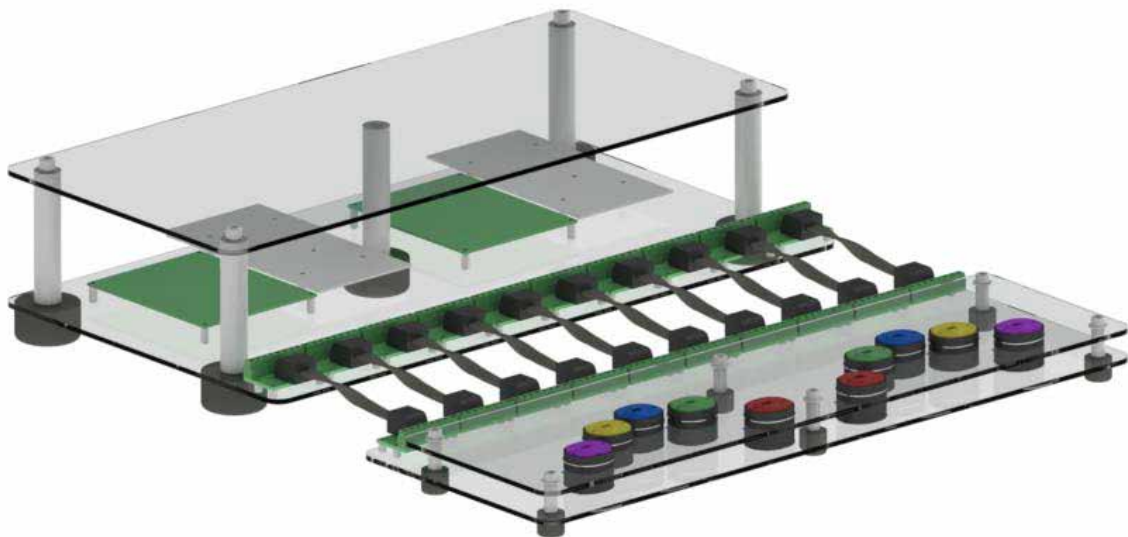


Figure 4.1: A HapCaps System: A system composed of 10 HapCaps (tactile haptic buttons) built to develop finger sense while a student performs math activities.

4.1 Introduction

In Chapters 2 and 3, we developed open-source kinesthetic haptic devices and discussed educational applications created using those devices. In this chapter, we develop an open-source tactile haptic device for finger sense training (HapCap) as well as a system using ten HapCaps (HapCaps System) to perform finger sense training and math learning simultaneously. In addition, we report the use of the HapCaps System in a pilot study conducted in a first grade classroom. We begin this chapter by motivating this work with an introduction of previous studies into the connection between finger perception and math learning (Section 4.1.1). We then review previous open-source tactile haptic devices (Section 4.1.2). Section 4.2 describes the design of HapCap, and Section 4.3 describes the design of HapCaps System. Section 4.4 explains how the HapCaps System control system works. Section 4.5 describes the tactile haptic feedback used by the HapCaps System. Section 4.6 describes a pilot study in which we tested the HapCaps System in a first grade classroom for four weeks.

4.1.1 Motivation

Finger sense (or finger gnosis, or finger perception) is defined as the ability to distinguish, name, or recognize the fingers [80]. Neuroscientists and education researchers have found evidence of a connection between finger sense and mathematical ability [80, 81, 82, 83, 84], but the mechanism for this connection is still unknown. In 1963, Kinsbourne and Warrington [81] observed a relationship between finger perception and dyscalculia (severe difficulty in making arithmetical calculations, as a result of brain disorder). Butterworth [83] postulated that “without the ability to attach number representations to the neural representations of fingers and hands in their normal locations, the numbers themselves will never have a normal representation in the brain” ([83] pp. 249-250). Noel et al. [80] performed a study that tested 41 students at the beginning of first grade on finger gnosis as well as their performance in two tasks evaluating global development (processing speed and use of the dominant hand). Fifteen months later, they found that performance in a finger gnosis

test was a better predictor of numerical skills than the global development test, but not of reading skills. In 2008, Gracia-Bafalluy and Noel [82] selected the 33 (out of 112) first-graders with the lowest scores on a finger gnosis test and divided them into two groups. One group received training in finger differentiation for eight weeks and the other group in reading comprehension. Their results showed that a finger intervention led to improved finger gnosis in young children as well as an improvement in numerical skill. In 2017, Jay and Betenson [84] conducted a study over 4 weeks with 137 students between 6 and 7 years of age. They divided the participants into four groups: one that received finger training exercises, one that played number games, one that received both interventions, and one that was a control group. They found that a combined intervention, incorporating both finger training and symbolic number games, significantly improved participants' numeration scores, but neither intervention by itself had a significant effect.

The activities used in [82] and [84] to develop finger perception ranged from children following paths with their fingers to touching different colored circles on a board. During these activities the somatosensory system is used to touch the different colored circles as well as follow the different paths with the fingers, so we postulate that a haptic device could also be used to develop finger perception. We designed an open-source tactile haptic button (HapCap) as a tool to combine learning in the classroom and finger sense training for children aged five to seven years of age, because finger sense development occurs in children up to approximately 7 years of age [81]. We then designed a system that uses ten HapCaps (HapCaps System) to develop finger perception on first grade students as they are performing math activities on a computer.

4.1.2 Open-Source Tactile Haptic Devices and Tactile Haptic Devices used for Educational Applications

To our knowledge, there have only been two open-source tactile haptic devices developed as of this writing. One is the Tpad Tablet project [37, 85] and the other is OpenGlove [86]. The Tpad Tablet [37, 85] incorporates a variable friction surface,

which uses piezoelectric actuators, with a tablet computer. The Tpad Tablet is made from open-source available technologies, and instructions for how to build one are on its website (www.tpadtablet.org). OpenGlove [86] is an open-source tactile haptic glove that uses vibration motors to give vibrotactile feedback to the user. Its CAD models and software are provided on its website (www.openglove.org). OpenGlove is a tool for developers to create haptic applications for virtual reality. Its hardware design allows designers to reconfigure the placement of 5 vibration motors along the fingers. The software API is provided in C#, Java, and Javascript, and it allows developers to use OpenGlove with virtual reality platforms such as Oculus Rift (www.oculus.com).

There has been limited research using tactile haptic technology for educational applications. The TPad Tablet technology in its phone format (TPad phone) was used by Hightower et al. [87] to develop a science journaling application, which allows learners to capture and annotate images with audio and haptic feedback. Behesti et al. [88] used a Tanvas Tablet [89], a commercial variable friction display, to develop TCircuit, an application that enables parent-child dyads to feel virtual electric current flowing through a circuit diagram by touching the display.

4.2 HapCap Design

The finger sense training developed by Noel et al. [82] focused on helping children differentiate their fingers. Children learned an association between each finger and a color, and performed activities with their fingers in which they traced different colored paths, pointed at different colored circles in a grid pattern, and touched different colored rectangles arranged similarly to piano keys, while using a specific finger to touch a specific color. During these activities, children developed finger mobility, visual-motor coordination, and an awareness of their fingers. Our goal was to achieve similar methods and results using novel haptic devices.

Noel et al. [82] used colored paper pieces to arrange the activities in front of the students and a researcher working one-on-one with the student would guide them through the exercises and provide feedback based on their performance. We propose

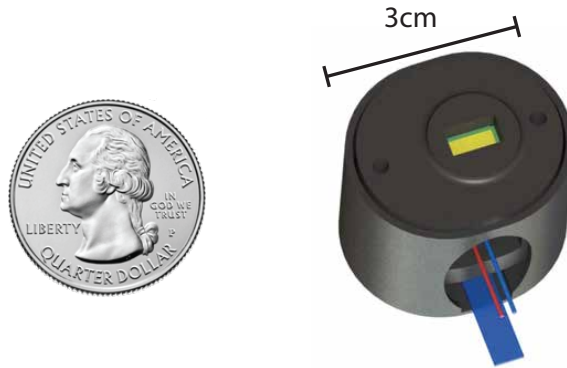


Figure 4.2: A HapCap is a haptic button that can be programmed to give vibrotactile feedback in response to a press. Here a HapCap is shown next to a quarter for size comparison.

that a series of colored buttons capable of differentiating different fingers and that can be arranged in different configurations similar to the activities with colored circles and rectangles used by Noel et al. [82], could also be used to develop children’s finger mobility, visual-motor coordination and an awareness of their fingers. We also propose that vibrotactile feedback could either substitute or reinforce visual and auditory feedback given by the researchers, such as telling students they are performing a certain activity correctly.

A HapCap (Figure 4.2) is a haptic button that can be programmed to give vibrotactile feedback in response to a finger press. It is equipped with a color sensor, which when used in conjunction with a glove that has colored fingertips, can determine which finger was used to press it. Our aim was to design a haptic button that could be combined with other buttons in different configurations to support various finger training activities for young children aged 4 to 7 years old. We chose this age range because previous finger perception studies [80, 82, 84], found a connection between finger sense and math learning in children aged 4 to 7 years. In this section we will describe the design considerations (Section 4.2.1) and design implementation of a HapCap (Section 4.2.2).

4.2.1 HapCap Design Goals

The design goals for a HapCap can be categorized into manufacturing and assembly requirements, and usability and rendering requirements. This section describes these goals.

Manufacturing and Assembly Requirements

Similar to Hapkit and Haplink, a HapCap is an open-source haptic device designed for educational applications. All of the standard files (parts lists, solid models) are freely available at <https://sites.google.com/view/hapcaps>. In contrast to Hapkit and Haplink, a set of HapCaps are not meant to be manufactured and assembled by students but rather by researchers and educators designing the finger training activities for the students. Thus, educators and researchers should be able to obtain and/or manufacture the parts themselves. Additionally, an unambiguous and robust assembly process is required as with Hapkit and Haplink. In contrast to the previous devices described in this thesis, we did not specifically design the assembly process so students could learn from it, because a HapCap is not meant to be assembled by students. A HapCap is also different from Hapkit or Haplink in that it is not meant for students to learn through observing its physical components interacting, so we did not design for the device to be uncovered; rather, we focused on protecting its fragile electronic components.

Usability and Rendering Requirements

Our aim was to create a haptic button that recognized when a user was pressing it, identified which finger was pressing it, and provided the user with a vibrotactile haptic cue. In order for a HapCap to recognize a press from a 4-to-7-year-old child, the force required to hold down the device should be small enough so that a child that age can exert it with their fingers, but not so small that an unintentional touch would be detected as a press. The force required was obtained experimentally; the process will be described in the following section.

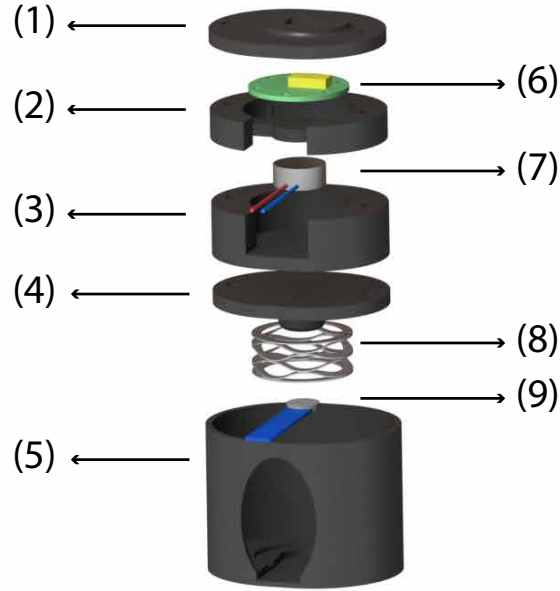


Figure 4.3: HapCap Elements: (1) HapCap Lid. (2) Color Sensor Holder. (3) Motor Holder. (4) HapCap Stud. (5) HapCap Shell. (6) Color Sensor. (7) Vibration Motor. (8) Disc Spring. (9) Force Sensor.

4.2.2 HapCap Design Implementation

The design of a HapCap was an iterative process consisting of creating a design, piloting the new design with children in the laboratory, learning from watching the children interacting with the device, and improving the design in order to meet the requirements outlined in Section 4.2.1. The HapCap design went through three rounds of iterations in that piloted on two separate occasions with a ten-year-old child and on one occasion with a four-year-old child and two seven-year-old children. From these pilots we learned that the lid of the HapCap should contain a feature that indicates to the children exactly where they should press the button, and the force that should be required to press a HapCap. These findings will be further described below, along with the the final design and components of a HapCap. These components, shown in Figure 4.3, are divided into off-the shelf components and 3D-printed structural components.

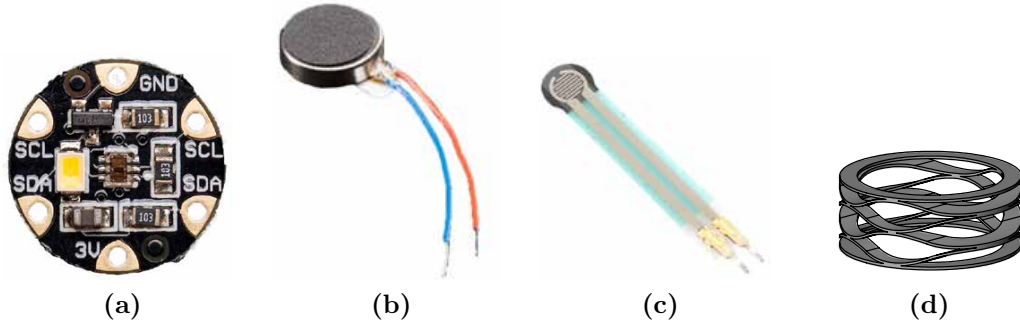


Figure 4.4: HapCaps hardware components: (a) Adafruit Flora color sensor. (b) Adafruit vibration motor. (c) Pololu force sensitive resistor. (d) McMaster stacked wave disk spring.

Off-the-Shelf Components

Figure 4.4 shows the off-the shelf components used to build a HapCap. All of the components can be purchased online from vendors such as Adafruit (www.adafruit.com), Pololu (www.pololu.com), and McMaster (www.mcmaster.com). Here we list the components and explain their purpose:

- Color Sensor:** An Adafruit Flora Color Sensor (Adafruit product ID 1356) at the top of the HapCap detects the color of the fingertip that is pressing the button. This sensor integrates a TCS34725 color sensing chip from TAOS (www.taos.cc) as well as a white illumination LED in a $15 \times 15 \times 2$ mm circuit board. This sensor shines the white illumination LED, and measures the unfiltered, red, green, and blue light intensities of the reflected light. It then transmits the measured light values as a package containing 4 16-bit numbers over an I2C bus (a synchronous, multi-master, multi-slave, packet switched, single-ended, serial communication bus). These color data can be used to determine the color of objects placed in front of the sensor (in our case the color of a painted finger on a glove), by comparing the intensity values of each measured color, to those of known colors (i.e. purple should be approximately 50% red and 50% blue.)
- Vibration Motor:** In order to provide haptic feedback to a user, a HapCap

uses a coin vibration motor from Adafruit (Adafruit product ID 1201). The vibration motor can operate at voltages between 2 and 5 V when turned on. The input voltage to the motor on a HapCap is 5 V, where it has a rotational speed of 11000 RPM and a load current of 100 mA. When turned off, the input to the motor is 0 V. The motor is 10 mm in diameter and 2.7 mm in thickness. Section 4.5 describes how the vibration motor is used to generate haptic feedback on a system that uses 10 HapCaps.

- Force Sensitive Resistor:** In order to detect when a HapCap is pressed, we use a force sensitive resistor (FSR) from Interlink Electronics and sold by Pololu (Pololu part ID 1695). This is a passive component that exhibits a decrease in resistance when there is an increase in the force applied to the 0.5 cm diameter active area. It can sense applied force between 0.2 and 20 N and exhibits no hysteresis. Because the FSR can have errors between 0.5 and 25% in absolute force measurement, it is not suitable for precision force measurements. Therefore we use it as a contact sensor; i.e. we specify the minimum force required to press the button.
- Disc Spring:** In order to specify the amount of force needed to press a HapCap and to make it feel like a button (i.e. move some distance before bottoming out), we use a stacked wave disc spring from McMaster (McMaster part number 1561T26). We chose a stacked wave disc spring over a coil compression spring in order to minimize the overall height of the button; A stacked wave spring occupies a smaller height profile for the same force and displacement requirements as a coil compression spring. Figure 4.5 shows the HapCap's button mechanism. When a force greater than or equal to 11.6 N is applied to a HapCap, the Disc Spring compresses, the HapCap travels 1.9 mm, and the extruded feature of the HapCap Stud (Figure 4.7(b), component (4)) makes contact with the FSR. The 1561T26 disc spring from McMaster has a spring constant of 6.1 N/mm and the total force required before the FSR is contacted was set to 11.6 N by setting the required displacement to 1.9 mm. The force required for a button press is a trade-off. On one hand, we want the force to be as large as possible

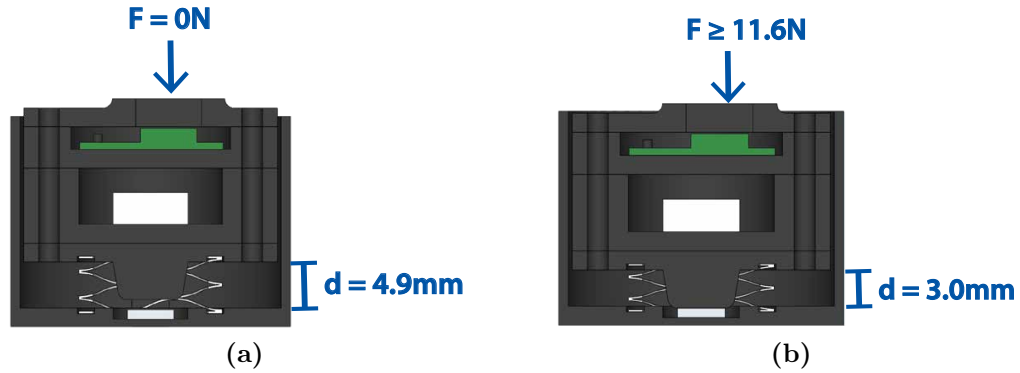


Figure 4.5: HapCap cross-section showing the Disc Spring compressing as a result of an applied force. **(a)** When no force is being applied, i.e. $F = 0\text{ N}$, the distance between the HapCap Stud and the Shell is 4.9 mm. The Disc Spring is not compressed. **(d)** When 11.6 N of force are applied to the HapCap, the Disc Spring compresses and the distance between the HapCap Stud and the Shell is reduced to 3.0 mm; i.e., the HapCap travels 1.9 mm until the extruded feature on the HapCap Stud makes contact with the force sensor and cannot travel further.

in order to prevent unintended button presses. On the other hand, we want the force to be small enough for young children to be able to press the button. The force required for a button press was chosen as a result of our pilots with young children in the laboratory. We chose a force of 11.6 N because we observed that with this amount of force it was a little difficult for the students to press the button, making sure that the presses were purposeful, but not so difficult that they did not have enough strength to accomplish the press.

Structural Components

A HapCap’s structural components (shown in Figure 4.6) are designed to be made with an FDM (extrusion-type) 3D printer. FDM 3D-printing technology is the least expensive 3D-printing technology available to date and there are many FDM 3D printers available for purchase at a low cost, such as Makerbot and FlashForge Creator Pro. The contour of a HapCap’s structural components is rounded, which reduces risk of warping during printing, and there are no overhangs, which allows for the print to be done without support material.

The HapCap structure was divided so each sub-component interfaces with one

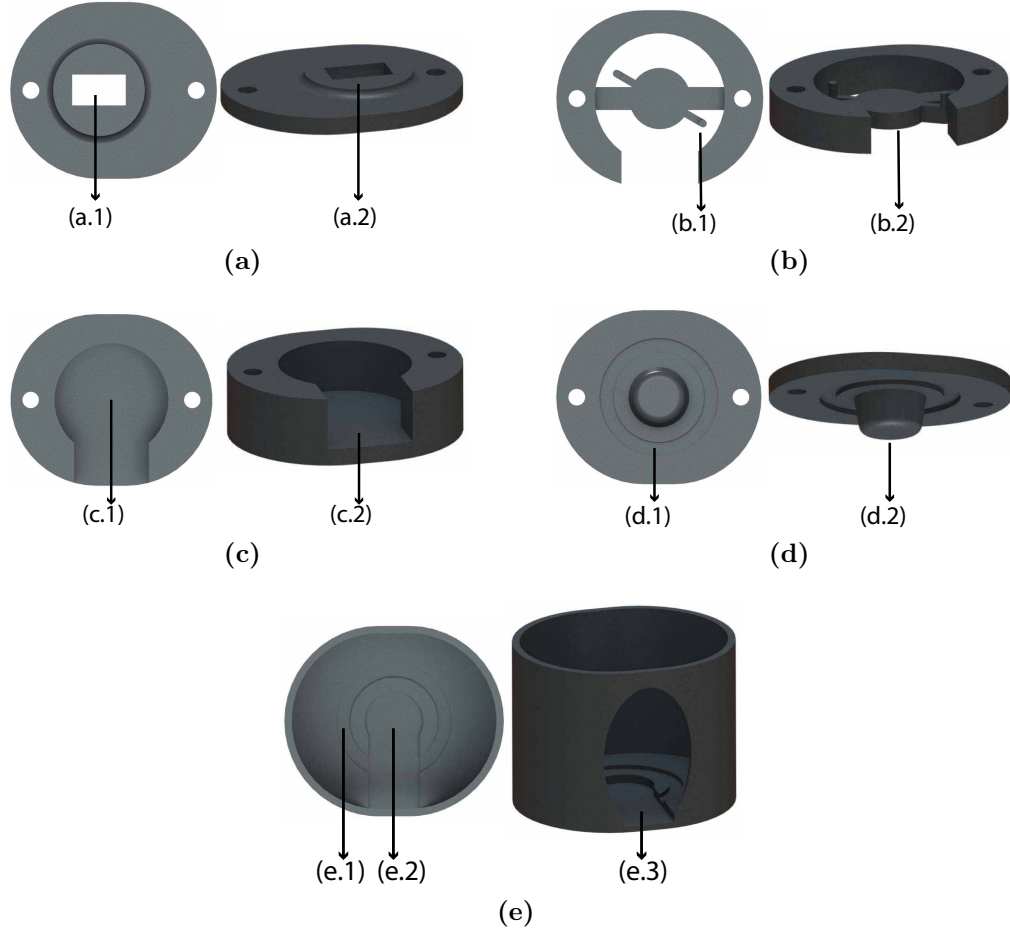


Figure 4.6: HapCaps 3D printed structural components: **(a)** HapCap Lid with (a.1) a slot for the light from the color sensor to pass through and (a.2) an extruded feature to indicate to the user where in the HapCap to press. **(b)** Color Sensor Holder with (b.1-b.2) features to hold the color sensor in place. **(c)** Motor Holder with (c.1) space to place the vibration motor in and (c.2) a slot for the wires from the color sensor and vibration motor. **(d)** HapCap Stud with (d.1) a slotted feature to place the compression spring in and (d.2) an extruded feature or knob that acts like a button to press the force sensor. **(e)** HapCap Shell with (e.1) a slotted feature to place the compression spring in, (e.2) a feature that holds the force sensitive resistor, and (e.3) a slot for the wires from the motor, FSR and color sensor.

electrical component. In order to assemble a HapCap, we first place each of the electronic components in its corresponding structural component (i.e. the motor is placed inside of the Motor Holder). In the final assembly process we stack the HapCap Lid, Color Sensor Holder, Motor Holder, and HapCap Stud on top of each other using the outer contour of each part to align them, install two screws to secure the stack, and place the stack inside of the Shell.

The color sensor detects color by analyzing light from a bright white LED that is reflected back. There is a slit on the HapCap Lid that allows the white LED light from the sensor pass through. The slit is surrounded by a circular raised area of 1 cm in diameter, which in addition to the slit, indicates to the user where to press the HapCap. We added the raised area on the HapCap Lid after piloting with young children and determining that without the raised area, it was not intuitive for children that they should press the button covering the entire slit. In the first test of the HapCap design with young children, the children pressed anywhere on the HapCap and would sometimes not completely cover the light emitted by the color sensor with their fingers which caused the color of the finger to be incorrectly detected. We then added a raised area of 1 cm in diameter and saw in the second pilot that the children correctly identified where on the HapCap they should press without having to be instructed. In order to prevent the light emitted by the color sensor from reflecting inside of the HapCap structure, which would create noise and affect the color sensor reading, the structural components of a HapCap are printed using black material. During our design iterations and tests with young children in the laboratory, we printed HapCaps using a Makerbot Replicator and black PLA.

4.3 HapCaps System Design

As described in Section 4.1, Jay and Betenson [84] found that students aged 6 to 7 years old who received an intervention in finger perception training as well as one in number games involving multiple representations of numbers showed improvement in quantitative skills. We propose that a device which combines finger perception training and math learning will aid in understanding the underlying mechanisms

behind Jay and Betenson's [84] findings.

As mentioned in Section 4.2, HapCaps were designed to be arranged in different configurations to enable finger perception training activities. The HapCaps System is a system composed of ten HapCaps designed to develop first-grade students' finger perception as they perform math activities on a computer. It combines math learning and finger sense training by associating each HapCap with a single finger as well as a color. Students then use the corresponding finger and HapCap to answer math activities on a computer (Figure 4.7). In this section we will describe the design considerations (Section 4.3.1) and design implementation of a HapCaps System (Section 4.3.2).

4.3.1 Design Goals

Our design goals for the HapCaps System can be categorized into manufacturing and assembly requirements, and usability and user requirements. This section describes these goals.

Manufacturing and Assembly Requirements

Similar to a HapCap, the HapCaps System is open-source and designed for educational applications. All of the standard files (parts lists, solid models, code) are freely available at <http://sites.google.com/view/hapcaps>. The HapCaps System was designed as a research tool to be used for studies in a first-grade classroom. If proven effective for finger sense training, it could also be deployed in the classroom as part of the standard educational practice. The assembly process of the HapCaps System is more complex than that of HapCaps, and it was designed for researchers and educators with some soldering and laser-cutting experience. Because the HapCaps System is an open-source device for research and educators, it follows that it should have an unambiguous assembly process, the hardware and electronic components should be easily procurable from vendors online, and the custom components should be able to be manufactured by the researchers and educators themselves using prototyping equipment available in academic facilities such as 3D printers and laser cutters, or

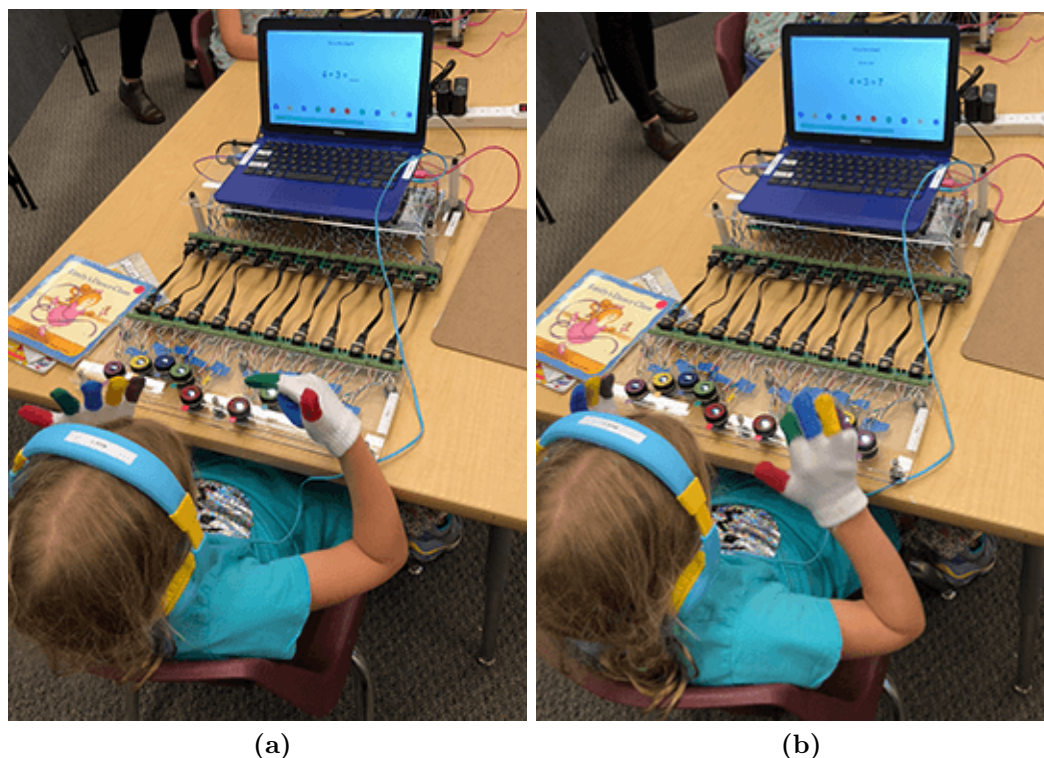


Figure 4.7: A student using a HapCaps System to perform math activities on a computer: (a) A student is presented with the question $4 + 3 = ?$ and determines that she should select her 7th finger (green) to press on the 7th button on the HapCaps System. (b) The student has pressed HapCap 7 with her 7th finger and obtained a right answer. The screen now shows that she has correctly answered the question $4 + 3 = 7$.

easily sourced from external vendors in low quantities (10-100 units).

Usability and User Requirements

The HapCaps System needs to be robust to use by first graders. We found that first graders (typically ages 5-7) are curious and handle the hardware roughly and touch the device in ways unrelated to its intended use. Thus, any fragile components should be inaccessible to users. The device also needs to be safe for young children to use: it should not require high voltage or current for the electronics, and the mechanical components should not contain sharp edges or pinch points. The HapCaps System

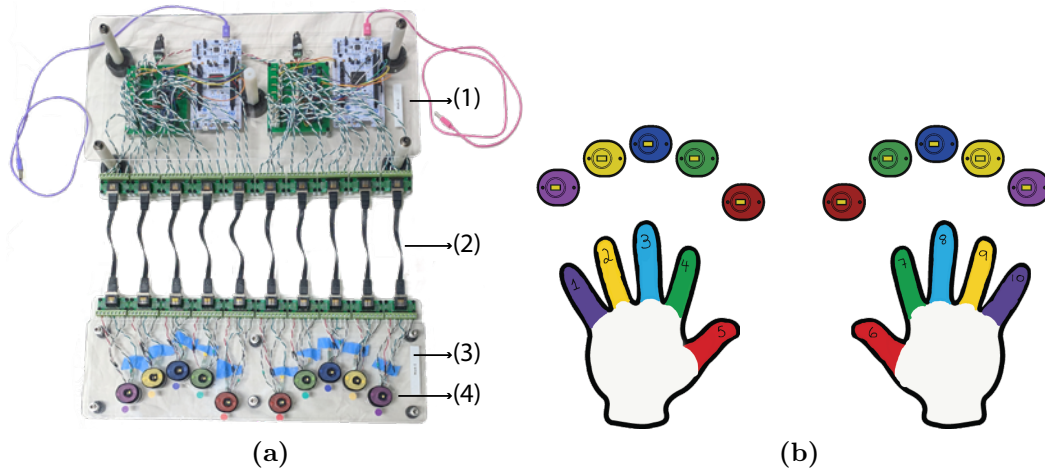


Figure 4.8: (a) HapCaps System components: (1) Circuit Enclosure. (2) Ethernet cables. (3) Button Enclosure. (4) HapCaps. (b) HapCaps gloves have colored fingertips which correspond to the colors of the HapCaps in order to detect which finger is pressing on the HapCap. The colors of the fingertips are red, green, blue, yellow, and purple, starting with the thumbs and ending in the little finger.

should have affordances that make it intuitive for first graders to use: by looking at the device, a student should know which HapCap is meant for which finger and where on the HapCap to press. The HapCaps system should be able to support first-grade math activities, so we designed a system with 10 buttons to support activities that involve the numbers 1-10. The HapCaps System connects to a computer which serves two purposes: First, this connection is used to program the microcontrollers on the system to control the buttons. Second, a user can program different math activities on the computer which interact with the buttons on the HapCaps System (e.g. a computer program can respond to a press of a button).

4.3.2 HapCaps System Design Implementation

We designed the HapCaps System (shown in Figure 4.8(a)) to contain ten HapCaps (one for each finger), which are positioned in an enclosure (Button enclosure) in a pattern that mimics the relative distances between the fingertips on a human hand. When interacting with the device, the user wears gloves with colored fingertips. The colors are red, green, blue, yellow, and purple from the thumb to the little finger on

each hand (Figure 4.8(b)). In addition, each HapCap is colored with the color of its designated finger, e.g. the HapCaps that are meant to be pressed by the thumbs are colored red. Both of these affordances help the user choose which finger should press each HapCap. As mentioned in Section 4.2, the top of the HapCaps have a color sensor that detects the fingertip color and verifies that the correct finger was used with the correct button.

In order to interface the HapCaps System to educational software running on a computer, an intermediary electronics system reads information from the HapCaps, communicates that data to a computer, and controls the active components of the HapCaps. This intermediary electronics system was placed in a separate enclosure (Circuit Enclosure) in order to keep the electronics outside of the user's natural reach. The Button Enclosure, which holds the buttons and was designed to withstand first graders pushing on it, connects to the Circuit Enclosure with 10 Ethernet cables, each of which bundles the communication and power lines of a single HapCap into a single cable. We chose Ethernet cables to transport the signals from the HapCaps to the Circuit Enclosure because standard Ethernet cables have 4 pairs of twisted wire, which is enough conductors to carry all 8 of the required signals from each HapCap. Ethernet cables were designed for communication purposes, and thus are well insulated and minimize cross-talk between signals. The latching connectors on standard Ethernet cables also prevent the cable from becoming unintentionally unplugged when the device is moved. The Circuit Enclosure supports the weight of a monitor or laptop that is used to run the educational software. Design details of the Button Enclosure and Circuit enclosure are discussed below.

HapCaps System Button Enclosure Design

The Button Enclosure (Figure 4.9) was designed to hold ten HapCaps in a specific layout. It is made from two sheets of acrylic, where the lower sheet acts as a platform to support the HapCaps, while the upper sheet holds the HapCaps in place using cutouts shaped like the HapCap contour. The two sheets are connected with five plastic posts, each of which is 9/16 inches long and has a rubber foot at its base.

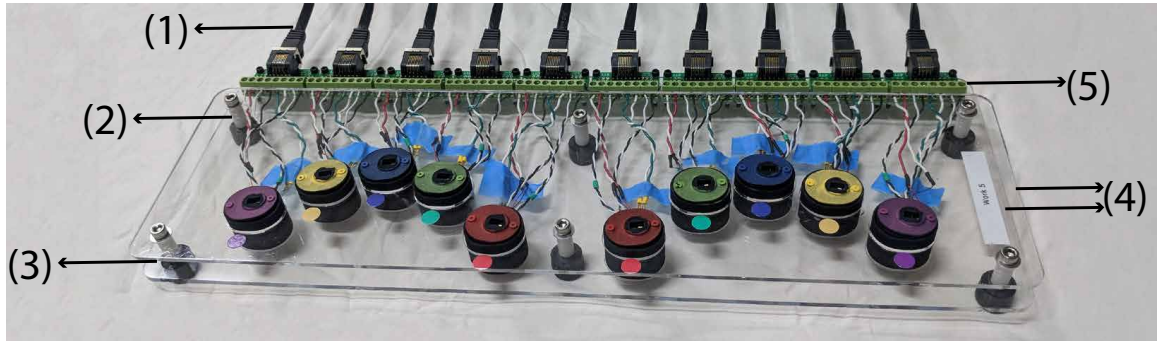


Figure 4.9: HapCaps System Button Enclosure components: (1) Ethernet cables. (2) Plastic posts. (3) Rubber feet. (4) Acrylic covers. (5) Ethernet breakout board.

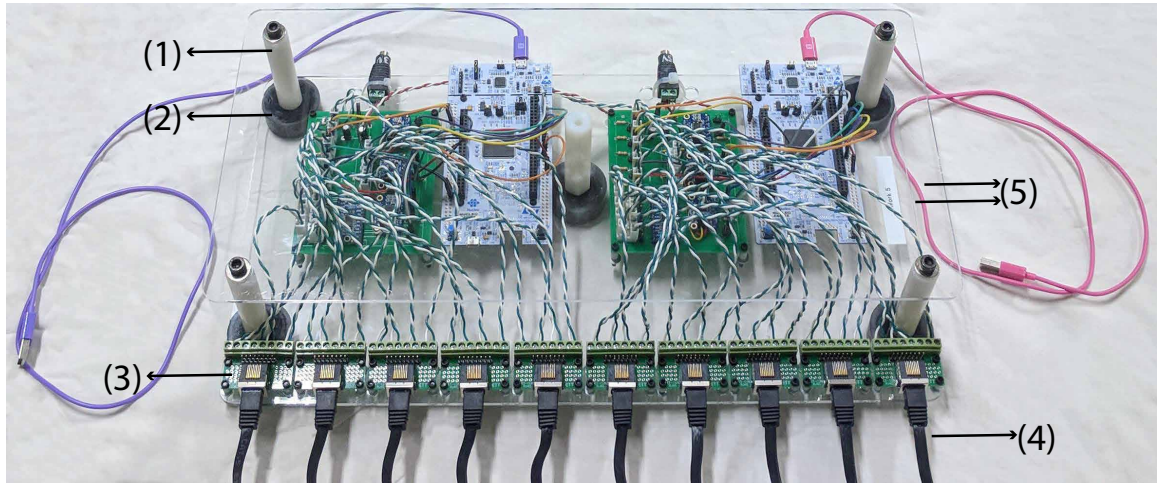


Figure 4.10: HapCaps System Circuit Enclosure components: (1) Plastic posts. (2) Rubber feet. (3) Ethernet breakout board. (4) Ethernet cables. (5) Acrylic covers.

These rubber feet provide a high-friction interface to the table that prevents the enclosure from slipping. We use ten RJ45 breakout boards made by Gravitech and sold by Mouser Electronics (Mouser part number 992-RJ45-TERM) in order to bundle the eight wires that come out of each HapCap into an Ethernet cable that transports the signals to the Circuit Enclosure.

HapCaps System Circuit and Circuit Enclosure Design

The Circuit Enclosure (Figure 4.10) is composed of two acrylic sheets supported by five plastic posts, each of which is $3/8$ inches long and has a rubber foot at its base.

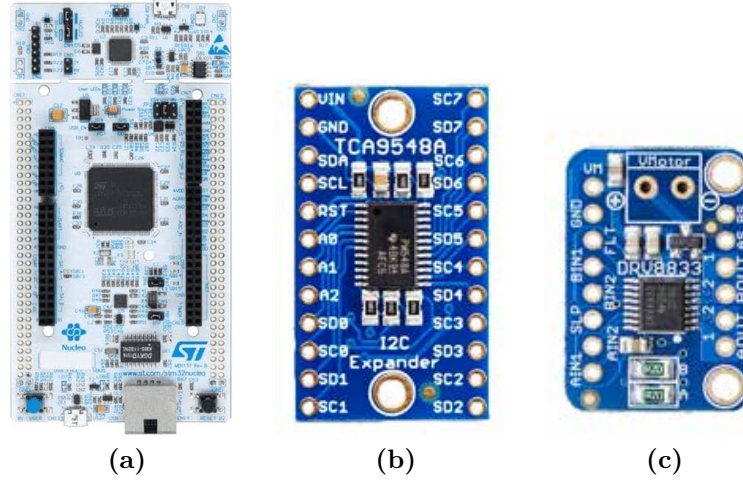


Figure 4.11: HapCaps Circuit electronic components: (a) Nucleo-F446ZE, (b) Adafruit I2C multiplexer, (c) Adafruit DRV8833 motor driver.

The Circuit Enclosure contains all of the electronics needed in order to interface a set of 10 HapCaps with a computer. It is also designed to support a laptop or monitor sitting on top of it. The electronics needed to interface a set of ten HapCaps with a computer are shown in Figure 4.11 and a circuit diagram is shown in Figure 4.12. These components are further described below:

- **Processor:** Each set of five HapCaps connects to one Nucleo-F446ZE microcontroller from ST Microelectronics (<https://www.st.com/>). The capabilities of the microcontroller are described in detail in Section 3.2. The Nucleo microcontroller continuously polls the FSR values for each of the 5 HapCaps it is connected to using, 5 different analog channels. When a change in the reading from an analog channel of 300 or more counts (out of 4095) is detected (i.e. a button press), the Nucleo queries that specific HapCap for color sensor output data, and uses that data to determine whether the color detected is red, blue, green, yellow, purple, or unknown. The Nucleo then communicates to the computer, over USB, which HapCap was pressed and the color detected, and then awaits further instructions from the computer. If the computer sends a message to the Nucleo that the HapCap and color combination was correct, the Nucleo plays a waveform on the vibration motor via the motor driver as

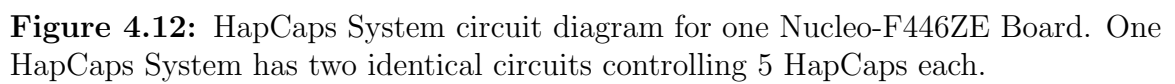
explained below.

- **I2C Multiplexer:** As mentioned above, each microcontroller is responsible for communicating with five color sensors. However, the I2C addresses of the Flora Color sensors from Adafruit are not programmable and are all the same (0x29). Since the Nucleo-F446ZE does not have 5 separate I2C buses, we decided to use the TCA9548A I2C multiplexer from Adafruit (Adafruit part number 2717) and a single I2C bus to connect to the five HapCaps.
- **Motor Driver:** We use an Adafruit DRV8833 motor driver (Adafruit part number 3297) to interface with the vibration motors. Because each motor driver is capable of driving two motors, we use 3 motor drivers for each set of 5 HapCaps.

4.4 HapCaps System Control

Figure 4.13 outlines the HapCaps System logic. Each HapCap is associated with a single finger as well as a color, which is used to answer the math questions on the computer. If a question is answered correctly, the student receives vibration feedback at the fingertip touching the button.

The HapCaps System uses a modular control scheme in which a set of five HapCaps is seen as a bundle and controlled as one item. To control each HapCap bundle, we use one Nucleo-F446ZE microcontroller and a custom fabricated electronics board, which contains the I2C multiplexer and motor drivers described in Section 4.3.2. Each microcontroller in the system communicates with educational software running on a computer over USB. The microcontroller informs the laptop of HapCap presses and which color was observed during the press, and accepts instructions from the laptop to activate the vibration motor in a specific HapCap. The firmware that runs on the Nucleo-F446ZE to control the HapCaps System is open-source and available at <https://sites.google.com/view/hapcaps>. The Nucleo-F446ZE has the capability for parallel programming through the use of two Direct Memory Access (DMA)



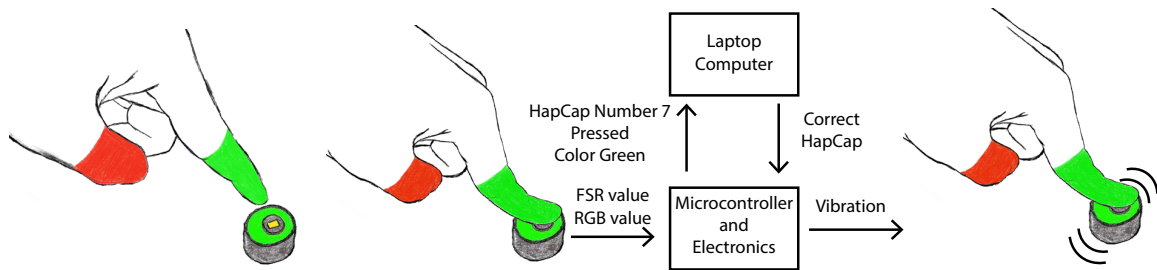


Figure 4.13: Flow chart for student pressing a HapCap: Once a student presses a HapCap, the microcontroller reads the force sensitive resistor values and determines that a press has happened. It then reads the RGB values from the color sensor and converts those values to a color value which is sent to a laptop computer along with which HapCap was pressed. The computer then determines if the correct HapCap was pressed for the current math activity and sends that information back to the microcontroller. If the correct HapCap was pressed, the microcontroller activates the vibration motor on the pressed HapCap.

buses with eight streams each. Using DMA, we split the program into three sections, which operate in parallel. Figure 4.14 shows the flowchart of the three sections.

In the first section (Figure 4.14(a)), one DMA stream (DMA channel 2, stream 0) handles the integration of the five ADC channels, which read the voltage across the five FSRs, and write the result into five different memory locations.

The second section (Figure 4.14(b)), handles the 5 pulse-width-modulation (PWM) timers, which are used to control the vibration motors. Five different DMA streams control the 5 different PWM timers. Each stream points to a location in memory where a user can program carrier waveforms as a stream of duty-cycle values. When a vibration motor is activated, the DMA stream corresponding to that PWM channel plays the programmed carrier waveform using the PWM channel to control the amplitude of the vibration of the vibration motors. The frequency of the motor's vibration will also vary with the amplitude of the carrier waveform. During pilot testing in the laboratory, we tested a sinusoidal carrier wave, a triangle wave, and simply turning the vibration motor on at a constant voltage. The vibration that we found most compelling was turning the vibration motor on for 100 ms at 5 V, which corresponds to a carrier waveform of a stream of 100% duty cycle for 100 ms. The resulting motor vibration is analyzed in Section 4.5.

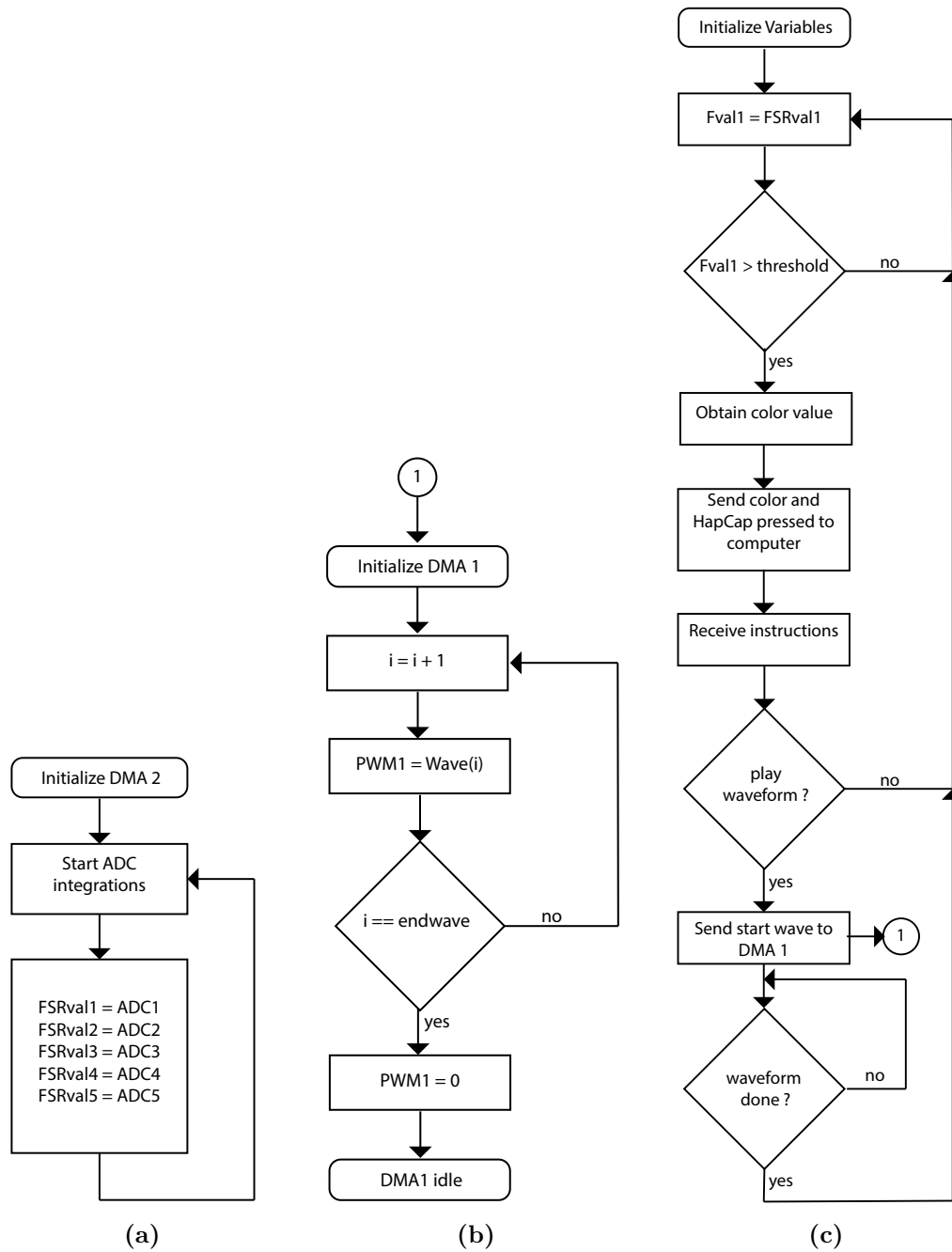


Figure 4.14: HapCaps System Control Firmware Flowcharts: (a) FSR flowchart. (b) PWM flowchart. (c) HapCap State Machine.

The third section (Figure 4.14(c)) handles the state machines for the five HapCaps. There is one state machine corresponding to each HapCap that determines whether the HapCap has been pressed and if it should turn the vibration motor on. Each state machine detects a HapCap press by comparing the FSR value to a threshold. When a press is detected, the software obtains the color detected by that HapCap and sends it to the computer via USB. The computer then instructs the HapCap to play a waveform or continue reading the FSR value. If instructed to play a waveform, the control software initializes DMA 2, which plays the waveform automatically, and waits for the waveform to complete before reading another FSR value. The state machines for the different HapCaps perform the same tasks and act independently of each other. With this control architecture, there is a 30 ms delay between the time a user presses a HapCap to the time the vibration motor is turned on.

4.5 HapCaps System Haptic Feedback

HapCaps give vibrotactile feedback to a user through eccentric mass vibration motors. The vibration motors are mounted inside the Motor Holder and are attached using glue and double-sided tape. As the motor vibrates, it excites the rest of the internal HapCap Button structure (HapCap Lid, Sensor Holder, Motor Holder, HapCap Stud), which is connected by the Disc Spring to the Shell. Because the waveform travels through the HapCap structure to reach the user, the waveform amplitude depends on the construction of the HapCap as well as the mounting of the motors. Factors include the density of the 3D-printed material and how tightly the 3D-printed structural components are clamped together by the screws, the angle of the spring in the Shell, and how well the structural components are aligned.

In order to characterize the vibration displayed by a HapCap, we performed two experiments. In both experiments we attached an accelerometer to the top of the HapCap and turned the vibration motor on for 100 ms using the PWM channel as described in Section 4.4. The first experiment is performed with no external forces acting on the Hapcap, i.e. the button was not pressed. In the second experiment, a person pressed the HapCap with a finger until a press was detected, and then

continued pressing the button while the vibration motor was on. With a finger on the HapCap, the experimental method was not repeatable, because it is difficult for a human to hold their finger steady and apply constant force over multiple experiments. Performing the experiment with a more accurate form of compression force such as placing a weight on the button would have been more repeatable, but would have introduced unwanted dynamics into the system.

We performed these experiments on the 10 HapCaps of one HapCaps System. The measured maximum amplitude of the vibration with no finger on the HapCap ranged from 2 to 6 g. The measured minimum acceleration output with a finger on the HapCap was 0.3 g, and the maximum was 2 g. The frequency of the output signal with and without a finger on the HapCap ranged between 180 and 200 Hz. This is likely due to variation in the vibration motors (they are rated to 11000 RPM, which is equivalent to 183 Hz) as well as nonlinear damping effects due to the non-rigid connections between the vibration motor and the HapCaps structure and the grounding of the HapCaps themselves. The vibration motor is attached to the Motor Holder using adhesive tape and glue, and the HapCaps are attached to their Shell through a disk spring which is glued to the HapCap Stud and the Shell as explained in Section 4.4.

Figures 4.15 and 4.16 show a sample waveform measured by the accelerometer with and without a finger on a single HapCap. We plotted the filtered and unfiltered data to see other measured dynamics introduced by a finger. In order to filter the data, we used two second-order Butterworth filters to create a band-pass filter: a low-pass filter with a cutoff frequency of 1000 Hz to remove high-frequency noise, and a high-pass filter with a cutoff frequency of 50 Hz to remove the DC bias due to gravity and the tremor due to the finger press. With a finger pressing on the HapCap (Figure 4.16), there is an envelope waveform introduced at 10 Hz, which agrees with Riviere et al. [90], who measured a human tremor to be around 8-12 Hz. From this experiment we conclude that the vibrotactile feedback output by the HapCaps System is of an amplitude and frequency that can be perceivable by users. However, the measured variability in the vibration waveforms from button to button suggests that the number of distinct cues that can be conveyed to a user using HapCaps is

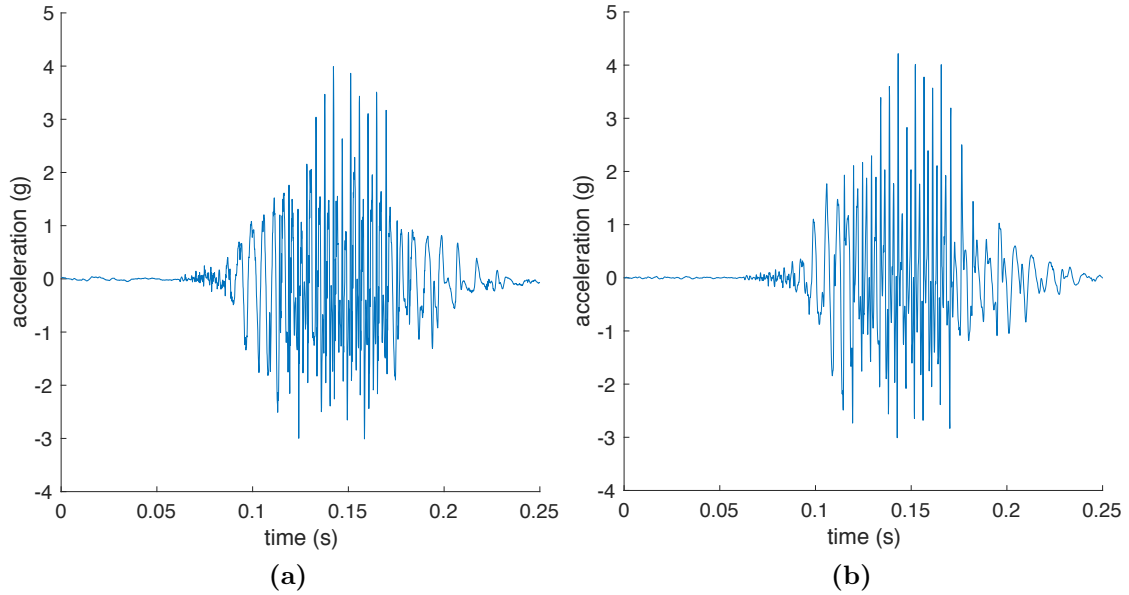
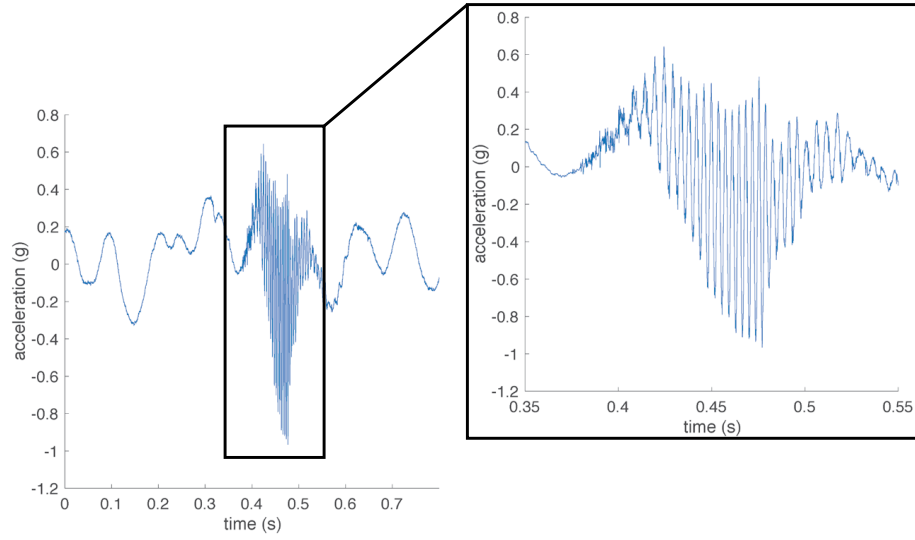


Figure 4.15: Haptic cue measured on one HapCap using an accelerometer without a finger on the HapCap. **(a)** Raw acceleration. **(b)** Filtered acceleration. In order to filter the data, we used two second-order Butterworth filters to create a band-pass filter: a low-pass filter with a cutoff frequency of 1000 Hz to remove high-frequency noise, and a high-pass filter with a cutoff frequency of 50 Hz to remove the DC bias due to gravity.

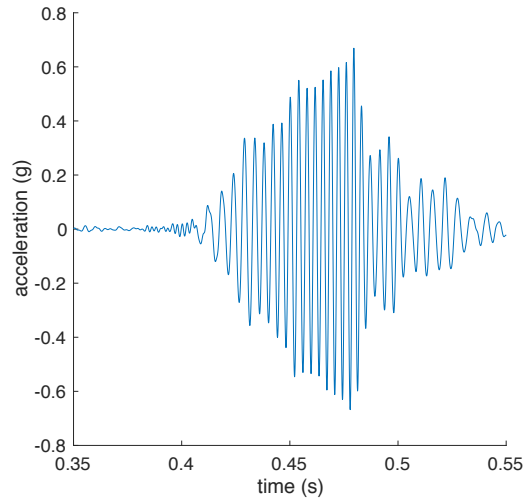
limited. In our case, we used the presence of haptic feedback to convey only one message: The user has responded the question correctly.

4.6 HapCaps System Use in the Classroom

As mentioned in Section 4.3, the HapCaps System was designed to combine finger sense training and math learning in the classroom for first-grade students. In Autumn 2018 we conducted a pilot study over four weeks with 19 first-graders in order to test the capabilities of the HapCaps System. The purpose of the experiment was to evaluate our device in a classroom environment as well as understand the logistics of using the HapCaps System hardware in a school setting. At the same time, we looked for any improvements in math and finger perception that would emerge from a short



(a)



(b)

Figure 4.16: Haptic cue measured on one HapCap using an accelerometer while a person is pressing the button. **(a)** Raw acceleration measured with finger press. **(b)** Filtered acceleration measured with finger press. In order to filter the data, we used two second-order Butterworth filters to create a band-pass filter: a low-pass filter with a cutoff frequency of 1000 Hz to remove high-frequency noise, and a high-pass filter with a cutoff frequency of 50 Hz to remove the tremor due to the finger press and the DC bias due to gravity.

study. In this section we describe the pilot study.

4.6.1 Methods

Participants

A total of 19 students in first grade from an elementary school in Menlo Park, CA participated in the experiment. All of the students belonged to the same first-grade class. Ten of the participants were male and nine were female. The age of the participants ranged from five to six years of age. All participants' parents gave informed consent, the participants gave assent, and the protocol was approved by the Stanford University Institutional Review Board (protocol # 46773).

The 19 participants were divided into two groups: The HapCaps group (treatment group) and the Computer group (control group). The participants were sorted by randomizing over sex and performance in a finger sense pre-test. 10 participants were placed in the HapCaps group and 9 participants were placed in the Computer group. Participants in the HapCaps group performed math activities using the HapCaps systems and participants in the Computer group performed the same math activities using a computer trackpad.

Data Collection

The main motivation behind this pilot study was to evaluate the HapCaps System in a first-grade classroom. In order to do so, we observed student's interactions with the devices, wrote notes during each session of hardware or software malfunctioning, and tested each HapCaps System before and after each session for functionality. In order to evaluate improvement in finger sense and math learning, all of the participants were pre- and post-tested in both finger sense and math.

Experimental Hardware: Workstations

For the study, we divided one classroom at the elementary school in half using a temporary wall and set up five Computer workstations on one side, and five HapCaps workstations on the other. Figure 4.17 shows an example of a HapCaps workstation.

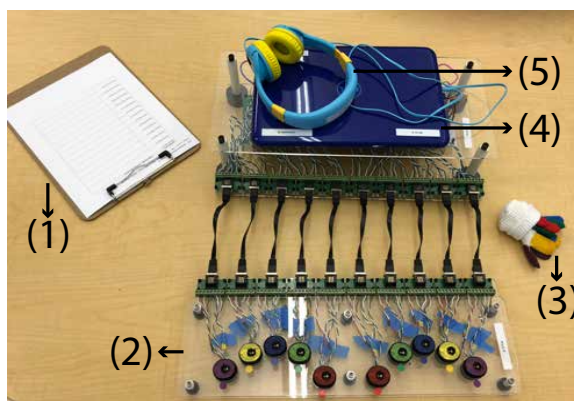


Figure 4.17: HapCaps Workstation components: (1) Clipboard used to write research notes. (2) HapCaps System. (3) Colored gloves. (4) Laptop computer. (5) Headphones.

A HapCaps workstation contained a HapCaps System, a Dell Inspiron 11 Laptop computer, which was used to run the HapCaps math software (the math software will be further described below), as well as headphones, colored gloves, and a clipboard with a form attached for researchers to make notes of observations during each test session. The headphones were used by the students in case they could not read a question. If a student was unable to read a question, they could press a key on the computer, and the software would read the question to them which they could hear through the headphones. The colored gloves were worn by the students, as explained in Section 4.3, and used by the system to detect which finger is pressing on a HapCap. We used the clipboards to take notes during the study of hardware or software failures, as well as the reasons why a student had to skip a question. If a student could not complete a question because they did not know the right answer, or because of technical difficulties, this was annotated on the forms attached to the clipboards. The Computer workstations contained all of the elements of the HapCaps workstations except for the HapCaps Systems.

Experimental Software: Math Software

Professor Jo Boaler and her team at youcubed developed a set of first-grade-level math activities. There are 28 categories of questions and they each have several variations. The different categories of questions are described in Appendix B. In order to support the 28 different question categories and the different variations of each type of question, we developed a software (Math Software) written in Processing [74]. Our software is open-source, and the code is available at <https://sites.google.com/view/hapcaps>. This section describes the components of the Math Software.

Graphical User Interface

The graphical user interface (GUI) is shown on the laptop screen during the activities. It presents the questions to be performed by the user. In the case of the students solving the activities without using the HapCaps systems, it is also the interface they use to answer the questions. It uses a horizontal layout that divides the screen into five areas. Figure 4.18 details the main elements of the GUI.

Software Structure

The main element of the Math Software structure is a state machine that initializes all of the variables, starts the communication with the HapCaps hardware if a Hapcaps System is connected, and then advances through the math questions. The state machine also handles special key presses (e.g. the right arrow key is used to move a student onto the next question) and writes the result of every question onto a file.

In software, the 28 different question categories are represented by a unique class type and the update function inside of each class contains a state machine that allows the question to have multiple answers. In order to program questions for a specific day, we write a .pde file (a Processing text file), where we instantiate each question as the appropriate class in the order we want the questions to appear and with the desired parameters. For example, for a simple addition question of $2 + 5 = ?$ to be the first question, we instantiate question one with the class “Simple Addition” and the parameters 2 and 5. The main state machine then extracts the questions from that file and runs them sequentially, advancing only when a correct answer is provided or

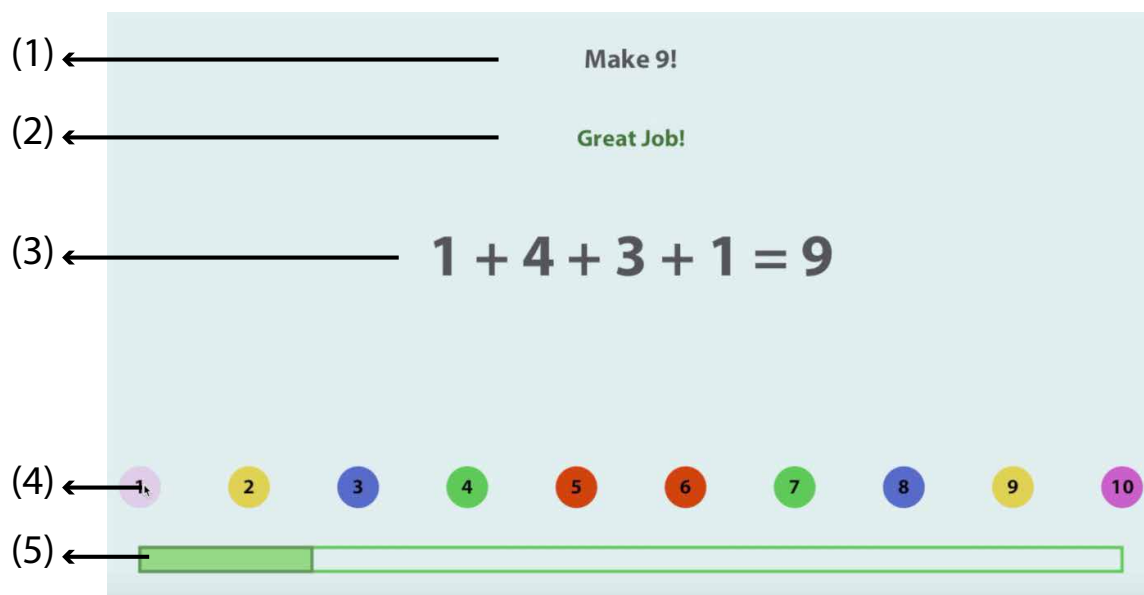


Figure 4.18: The HapCaps Graphical User Interface is divided into five elements. (1) Prompt: The top area of the GUI describes the math activity to the user. (2) Message: This area conveys messages to the user based on their performance. (3) Question: Poses the specific question and updates with user input. (4) HapCap Buttons: If there is no HapCaps system connected to the software, this area is used as virtual buttons to answer the questions. If there is a HapCaps system connected to the software, this area is used as a visual representation of the numerical value assigned to each HapCap. (5) Progress Bar: The progress bar shows how far the student has advanced in the math activities.

when a special key is pressed. During the study, we used the special key press to advance a student if they were unable to answer a question.

Procedure

Gracia-Bafalluy and Noël [82] conducted a longitudinal study over 8 weeks to understand the effects of finger training in young children’s numerical performance. During their study, they performed interventions with the young students twice a week. Because the main purpose of our pilot was to evaluate the HapCaps Systems, we decided to conduct interventions 3 times a week over 4 weeks. This way we would be able to test the capabilities of our devices over 75% of the sessions of a longitudinal study, and it was logistically easier to find a period of 4 weeks with no vacations (rather

than an 8-week period) during which an elementary school could accommodate our study. The study was divided into five phases: pre-test math, pre-test finger perception, math activities (with HapCaps Systems or computers depending on the group), post-test finger perception, and post-test math. In the following sections we describe each of the five phases.

Pre-Test Math

Two weeks before the math activities began, the 19 participants were given a math exam designed by Professor Jo Boaler's team at youcubed (www.youcubed.org), based on the first-grade math common core standards. The exam is presented in Appendix A, Section A.1.

Pre-Test Finger Sense

A week before the math activities began, all of the participants were evaluated in finger sense. The test, outlined in Appendix A, Section A.2, was an adaptation of those given in [80, 82]. During the finger sense test, participants sat across from examiners with one of their hands covered by a box. Examiners then performed 20 touches total of their fingers using brushes (10 individual touches, 5 simultaneous touches and 5 sequential touches). After each touch, participants were asked to identify the finger or fingers that were touched. The participants' finger perception was evaluated by counting the number of touches they were able to correctly identify without moving their hands. Figure 4.19 shows a student participating in the finger sense test.

Math Activities

The HapCaps and Computer groups performed the same math activities three times a week over four weeks for a total of 12 sessions. The HapCaps group used a HapCaps system (Figures 4.20(b) and 4.20(c)) to perform the activities and the Computer group used the trackpad on a laptop computer (Figure 4.20(d)) to perform the activities.

The first session (Day 1) was a 10-minute training session in which participants learned how to use the HapCaps Systems or the computers depending on which group they were in. After the initial session, participants answered 40 math questions in 20

minutes on Mondays and Fridays and 30 questions in 15 minutes on Wednesdays. If a student finished the activities early, the student read a book. The math activities were developed by Professor Jo Boaler and her team at youcubed and are provided in Appendix B. The questions for each session were chosen beforehand and arranged in increasing level of difficulty, incorporating at least 5 different types of activities in each session. Days 2 and 3 introduced the simplest version of each of the different types of activities. The questions increased progressively in their level of difficulty, culminating on the last day with the hardest level of difficulty for each type of question. The types of questions are described in Appendix B. As an example of how the questions progressed in difficulty, on day two students received three Single Digit Addition (Appendix B Section B.5) questions, on day three they received a Single Digit Addition with Missing Second Number (Appendix B Section B.6) question and a Simple Subtraction (Appendix B Section B.7) question, and on day four they received three more complicated Single Digit Addition with Missing Second Number questions. In this way, students worked up to answering Single Digit Addition with



Figure 4.19: Student participating in a finger sense test. During the test, participants (right) sit across from examiners (left) with one of their hands at a time covered by a box. Examiners then perform 20 touches total of their fingers using brushes (10 individual touches, 5 simultaneous touches and 5 sequential touches). After each touch, participants are asked to identify the finger or fingers that were touched. The participants are evaluated in finger perception by counting the number of touches they are able to correctly identify without moving their hands.

Missing Second Number questions using abstract representations of the numbers (Ten frames and Number Lines; Appendix B Sections B.9 and B.11), which were considered some of the hardest questions.

Students performed math activities in a special classroom that we prepared beforehand at the elementary school (Figure 4.20(a)). The students performed the activities in groups of five at a time and there were two researchers always present. We chose to limit the groups to five students to limit distractions and facilitate student focus on the activity. Before each session, the students were instructed to do the best they could, to have fun performing the math activities, and to take their time. We emphasized that the test was not a race, but instead was meant to challenge them. Participants were given minimal help while they were performing the math activities. If a student had a question, one of the researchers assisted with the requirements of a specific activity. However, researchers were not allowed to give a student the answer to the activity. If a student could not complete a specific activity, the researcher made a note on a form attached to a clipboard by the workstation and then moved the student to the next question.

Post-Test Finger Perception

A week after the study concluded, all of the participants were given the same finger perception evaluation as in the pre-test finger perception evaluation.

Post-Test Math

A week after the math activities concluded, all of the participants were evaluated in math using the same test that had been used in the pre-test math evaluation.

4.6.2 Results and Discussion

HapCaps Systems Robustness and Field Deployments

The HapCaps Systems were deployed successfully in a first grade classroom. No hardware broke, and participants were not put at risk. We built 7 systems in total in case we needed to replace any due to hardware failures, but no HapCaps Systems required



(a)



(b)



(c)



(d)

Figure 4.20: HapCaps Systems were used in a pilot study to develop finger perception in first graders as they perform math activities. (a) Researcher setting up the classroom with the HapCaps Systems. (b-c) Students using a HapCaps System to perform math activities. (d) Student using a laptop computer to perform math activities.

replacing, and we used the same five systems throughout the duration of the study. Before the study started, we transported 7 systems in the back of a car and stored them in cabinets at the school. Each day of the study, the HapCaps Systems were taken out of the cabinets, arranged on tables, used by students for an hour (every test day there were two sessions of 30 min each with two different groups of 5 students), and then put back in the cabinets at the end of the second session. Finally, at the end of the four weeks of the study, the HapCaps Systems were transported back to Stanford University where they were checked for functionality and proved to be functional.

HapCaps Systems Limitations

Although the hardware proved to be reliable, the Dell Inspiron computers used for the study were not fast enough and did not have enough RAM to run the HapCaps software and communications reliably. The computers were susceptible to freezing, which required them to be restarted. The effects of this were seen both in the HapCaps and Computer groups and it happened more frequently with the HapCaps group. Restarting a computer in the Computer group took approximately 2 minutes and in the HapCaps group up to 5 minutes. This caused the study experience to be different between students. For example, in one instance a computer froze for a student in the HapCaps group and, by the time it was restarted there were only a few minutes left in the session. As a result the student could not finish. Not having enough time to finish the activities visibly affected the emotional state of the participants. These types of problems might have had unpredictable effects on the math learning and finger sense improvement results.

Finger Sense and Math Improvement Results

Figure 4.21 shows the total correct points scored by each student in the finger and math pre-and post-tests. Most students improved both in finger perception as well as in math in the four-week period. Figure 4.22 shows the mean and standard deviation of the improvement in finger perception and math for the HapCaps and Computer groups. The mean improvement for the HapCaps group in finger perception was 5 points and 0.5 points for the students in the Computer group. A two-sample t-test

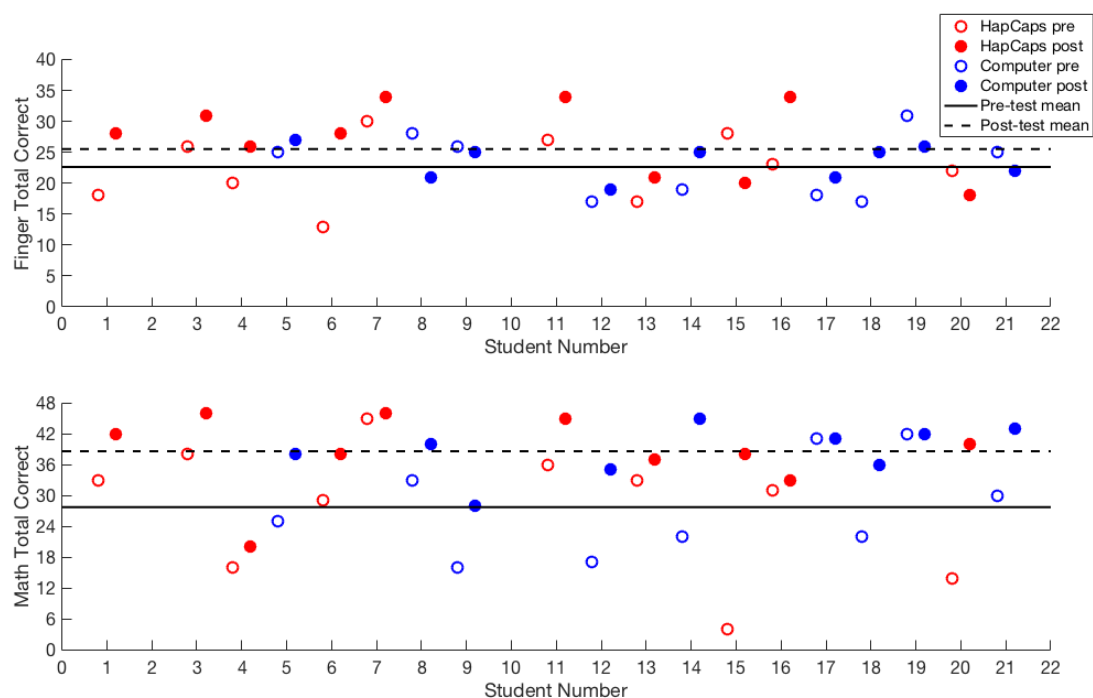


Figure 4.21: Finger perception and math pre-and post-test results by student.

showed that the groups are statistically different at the 12% significance level and a power analysis showed that at least 24 students would be needed in each group to obtain statistical significance at the 5% level. The mean math improvement for the HapCaps group was 10 points and 11 points for the Computer group. A two-sample t-test showed that there was no statistical significant difference between both groups in math achievement.

The study by Gracia-Bafalluy and Noel [82] was an 8-week intervention for the lowest performing students in math. We attempted an analysis of our data separating the lowest performers but we found that, because math performance data was not available at the time of the group selection, only three out of ten students with low performance were a part of the HapCaps group. With such a small sample size, statistical comparisons were not possible.

Gracia-Bafalluy and Noel showed improvement for the students who received the finger sense intervention in three numerical skills: subitizing, counting raised fingers, and ordinality judgement. However, they saw no difference between the students who

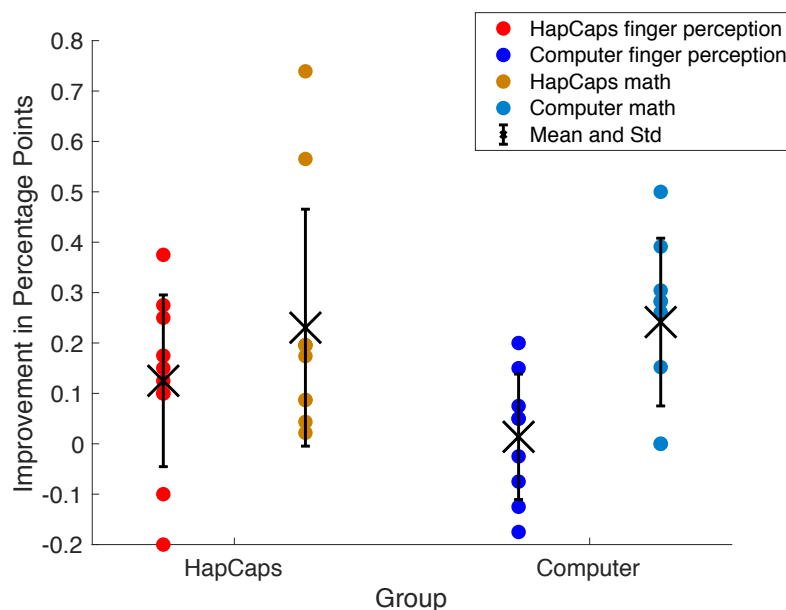


Figure 4.22: Finger perception and math improvement by group.

received finger training and those who did not in addition tasks and comparison of the magnitude of Arabic digits. Our study differed from that of Gracia-Bafalluy and Noel in that our students received finger training at the same time as they were performing complex math activities. Each activity did not focus on just one numerical skill. For example, the addition with missing addend question ($2 + ? = 5$) combines the concepts of counting, addition, understanding part whole relationships, and the understanding of a variable. In our short study we did not see statistical differences between the students who showed improved finger perception and those who did not in completing these complex mathematical tasks. We hypothesize that our study was too short (4 weeks) and that our sample size of participants was too small.

From this study we conclude there is evidence that a HapCaps system can be used as a tool for developing finger perception in children aged 5 to 6 years of age and that using fingers to perform math activities could lead to improvement in finger perception. A longer study with more participants and better computers is required to draw conclusions with statistical significance as well as any conclusions in math improvement results.

4.7 Conclusion

In this chapter we described the development of an open-source tactile haptic button (HapCap) as well as an open-source system that combines 10 of these buttons (HapCaps System), designed to enable first-grade students to perform finger perception training and math activities at the same time. We characterized the HapCaps System vibrotactile feedback and found that the vibration is perceptible but variable in the maximum waveform amplitude due to the construction of the HapCaps and their attachment to the rest of the system.

We conducted a four-week pilot study in which we tested the HapCaps System in a first grade classroom. The study used the devices to develop finger perception as students were performing mathematical activities. We determined that the HapCaps Systems' design was suitable to be used in a first grade classroom. However, we also found that the computers used with the HapCaps System did not perform adequately, and needed to be upgraded before conducting another study. The results of the study were not statistically significant due to the small sample size of students, but they do suggest that the HapCaps System can be used as a tool to develop finger perception. A larger study is required to verify this.

At the time of the writing of this thesis we have concluded an 8-week study with 100 participants, modeled after the study described in this dissertation. We took the learnings from the study described in this thesis and updated the computers to MacBook Air computers from Apple Inc, changed the duration of the study to 8 weeks, increased the number of participants to 100 students, and added math activities to support a longer study. We also changed the pre- and post-testing by adding questions to test for subitizing and number comparison ability as well as math knowledge in order to be able to better compare our results to those of [80] and [82].

Chapter 5

Conclusion

This dissertation has introduced the design of three open-source educational haptic devices. This chapter summarizes the contributions of the previous chapters and expands on ideas for future work.

5.1 Summary of Contributions

The major contributions of this dissertation summarized by chapter are:

Chapter 2:

- We present the design of Hapkit 3.0: an open-source, 3D-printed, kinesthetic, 1-DOF, customizable haptic device designed for educational applications and classroom use. Hapkit was used in numerous educational environments, including an online class on haptics and undergraduate and graduate classes at Stanford University in dynamics, controls and haptics. From the use of Hapkit 3.0 in haptics classes at Stanford, we found that students were able to customize and build on the device to create their own haptic applications. From the use of Hapkit in the online class, we found that Hapkit is a device that is accessible for a diverse community; students in the online class span a wide variety of age groups, education levels, and countries.

- We present a characterization of Hapkit dynamics for three versions of Hapkit in order to inform future design changes for Hapkit and other kinesthetic haptic devices. From this characterization we found that with Hapkit 3.0, users performed better in a stiffness discrimination task than with Hapkit 1.0, despite Hapkit 3.0 having an inexpensive motor that presents cogging torque.
- We introduce a novel application of haptic feedback in educational applications to substitute graphics in order to help students understand mathematical functions. We used Hapkit 3.0 to represent the graph of a trigonometric function with force feedback, and performed a study with two pairs of high-school students. From this study we found evidence that the students understanding of the connection between the trigonometric representations of the equation, graph, and unit circle, changed in tandem with their ability to correctly interpret the haptic representation.

Chapter 3:

- We present the design of Haplink: an open-source, 3D-printed, kinesthetic, customizable haptic device that can be used as a 1- and a 2-DOF device to teach science, math and engineering concepts incrementally. Haplink was used in an undergraduate freshman seminar on haptics where students learned kinematics and the rendering of virtual environments incrementally transitioning from 1 to 2 DOF. From Haplink's use in the freshman seminar, we found that students in the class were also able to customize and build on their Haplinks to create their own haptic applications.
- We present the design of a novel 2-DOF serial mechanism used in Haplink that allows for serial kinematics while permitting both motors to be grounded. We derive the kinematics of the new device, as well as an analysis of haptic rendering accuracy over the workspace.

Chapter 4:

- We present the design of HapCaps, an open-source, 3D-printed, tactile haptic button designed for finger sense training, as well as the design of HapCaps

System, a system that uses ten HapCaps to combine finger sense training and math learning for first-grade students. The HapCaps System was used in a pilot study in a first-grade classroom. During this pilot study, 19 first graders used the HapCaps System to perform math activities 3 times a week over 4 weeks. For the pilot study, we built 7 HapCaps Systems, brought them over to a first-grade classroom, and used them in the classroom for 4 weeks. We concluded that the HapCaps Systems were sufficiently robust to be used in a classroom environment with first graders.

- We found evidence that a HapCaps System can be used to improve finger sense in first graders as students who used the HapCaps Systems improved by 5 points on average in finger perception and those who did not only improved by 0.5 points on average. This results were statistically significant at the 12% level and a power analysis showed that a study with 48 participants would be required to attain results of higher statistical significance. During the study we also found that most participants improved significantly in math, but we did not find a statistically significant difference between the performance of the students who used the HapCaps Systems and those who did not.

5.2 Future Work

5.2.1 Educational Software for Haptics

In this dissertation, we designed open-source haptic devices for which we carefully considered the trade-offs between ease of making and performance in order to make them both accessible and useful. Making the accompanying software to a device accessible is also very important. The firmware of Hapkit, Haplink and HapCaps can be programmed using free coding environments (Arduino and mbed) using the C and C++ programming languages. However, the three devices would benefit from a simpler coding environment, similar to Scratch [91], that individuals with no software development experience could utilize to create haptic experiences. An example of a coding environment designed for simple generation of a haptic experience is Macaron

created by Schneider et al. [92]. This software is a web-based application that allows users to easily draw waveforms that can be displayed on vibrotactile devices. It would be very interesting to expand on this software interface and create a coding environment that allows a user to build their own educational application by dragging and dropping “haptic icons” and attaching these different haptic experiences to science, engineering and math concepts such as gears, viruses and cells. Minaker et al. [52] created a software interface that enables users to create their own spring exploration interface by allowing them to create springs of different stiffness and combine springs in series and in parallel. These are performed by simple drag and drop actions using a mouse. The resulting stiffness of the created springs can be felt through Hapkit 3.0. Even though the software interface created by Minaker et al. [52] only allows explorations using springs and is not a coding environment, it is a good example of a simple software interface for haptics concepts. It would be interesting to build on the work by Minaker et al. [52] and Schneider et al. [92] and create a haptics coding environment for individuals with no software experience.

It would also be beneficial if the educational haptic software was open source and written using the open-source software practices outlined by the Open-Source Software Foundation [93]. (I.e., it should include an open-source license, be available in a repository such as github, include clear documentation, etc.) Open-source haptic education software would enable a community of users with different expertise to contribute to the project and improve the software design from various perspectives.

5.2.2 Studying the effects of haptic device customizations in educational applications

As discussed in Section 1.2, current studies into the role of haptics in learning are limited due to the lack of customization capabilities offered by commercially available haptic devices. When these commercial devices are used for a study, the study has to be tailored to the device. With a customizable haptic device, the device could be tailored to the study. For example Wiebe et al. [15] used a Phantom Omni (a commercially available haptic device now marketed as the Touch) to enable students

to interact with a simulation of levers. We propose that the pen-like shape of the end-effector of the Phantom Omni is not the best form factor to interact with a simulation on levers, because other end-effectors could better transmit forces to the body in a way that mimics the interactions we have when pushing on levers. Hapkit 3.0 and Haplink have been successfully customized for a number of projects showcased in undergraduate and graduate courses at Stanford (some examples are shown in Sections 2.5 and 3.5). We propose that in future work it would be interesting to study the effects of different device customizations in learning outcomes.

5.2.3 Using Haptic Feedback to Represent Abstract Mathematical Concepts

Most haptic simulations for educational applications have been created to mimic physical experiences and concepts such as forces acting on a box [16], levers [15], and gears [7]. In Section 2.5 and published work [40], we propose that an interesting use of haptic feedback would be the creation of haptic representations of abstract mathematical concepts that are hard to teach in a laboratory environment such as trigonometry functions. This work could be expanded to create a framework for using haptics to represent other abstract concepts that are difficult to understand from visual feedback alone. For example, Piaget et al. [94] conducted experiments with students aged 6 to 7 years of age, and found that they have trouble understanding conservation of area principles as well as additive composition and groupings based on visual feedback alone. In order to aid students in understanding geometrical concepts related to area, it would be interesting to study the effects on learning of a haptic simulation where students explore concepts of groupings, conservation of area, and part-whole relationships. A simulation where the haptic feedback changes in proportion to the area would give the student not only visual feedback, but also haptic feedback to demonstrate the concept of changing areas.

There are other examples of math concepts that are hard to explain from visuals alone, such as vector fields, that may benefit from learning with other sensory modalities such as the sense of touch. It would be very interesting to conduct longitudinal

studies to understand if haptic feedback could replace or enhance visualizations of mathematical functions and what type of haptic feedback would be beneficial for different representations.

5.2.4 Improving Study Design on the Role of Haptics for Educational Applications

Most of the studies on the effects of haptic feedback in educational applications, as discussed in Section 1.2, lack a characterization of the haptic feedback used. We propose that haptic device design as well as the quality of the haptic feedback used in an educational simulation will influence the impact of haptic feedback in learning. This is supported by the findings of Jones et al. [11] who compared the effects of haptic feedback on learning in a lesson about viruses using a Phantom Desktop [24], a Microsoft Sidewinder Force Feedback Joystick, and a mouse, and found that students who used the Phantom Desktop outperformed the other two groups. We propose that in future educational studies into the role of haptics in education, a characterization and description of the haptic feedback used should be presented, and that variations on this haptic feedback and the impacts on learning should be explored.

The majority of the studies referenced in Tables 1.1 to 1.3 were performed as a single intervention in a laboratory setting, and lasted only a few hours. As a result, there is little research on the effects of using haptic devices for learning over longer time spans and in a traditional classroom setting. We propose that in order to gain further insights into the roles of haptics in learning science and physics concepts like the ones presented in the aforementioned studies, there should be educational studies conducted in a classroom environment over the course of several weeks, as we did with the HapCaps systems in Chapter 4.

5.2.5 Finger Sense Development and Math Learning

As discussed in Chapter 4, scientists have found evidence of the connection between finger sense and mathematical ability, but the mechanism for this connection is still

unknown. There are a number of different studies that can be conducted using HapCaps in order to better understand the mechanisms behind the connection between finger sense and math ability. Our first system designed using HapCaps (called the HapCaps System) was designed to support a finger sense training activity similar to the “piano” activity explored in [82]. We hypothesized that this particular configuration of ten HapCaps could aid student’s ability to map a number line onto the fingers, and thus promote math learning. However, other configurations of HapCaps could be explored. For example, a system where the HapCaps are arranged in a grid could support a different finger-based activity where students have to actively find the correct button to press. This activity may not promote the association of fingers with a number-line, but it could be better at developing finger differentiation. A study that compares outcomes from students performing finger sense and math activities with different configurations of HapCaps, like the “piano” and grid configurations, could provide insights into the development of finger perception and its relationship to math learning in young children. Studies into the specific role of haptics in finger perception could also be conducted by comparing conditions with and without haptics. Perhaps there are different finger perception activities that improve with haptics and others where the haptic feedback has no effect on outcomes.

Appendix A

Education Evaluations

In this appendix we present the evaluations used for the pilot study described in Section 4.6. Section A.1 presents the math evaluation and Section A.2 presents the finger perception evaluation.

A.1 Math Evaluation

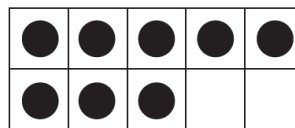
This section presents the paper and pencil math evaluation developed by Professor Jo Boaler and her team at youcubed based on standardized 1st grade level evaluations. This test was used as a math-level evaluation during our pilot study described in Section 4.6. The test was graded by the team at youcubed. Each question was worth 2 points. A correct answer was given 2 points and partial credit of 1 point was given at the discretion of the graders.

Question 1

$$4 + \boxed{} = 12$$

Question 2

What number is this?



Question 3

Which shape has more sides?
Circle it!



Question 4

$4+5=$

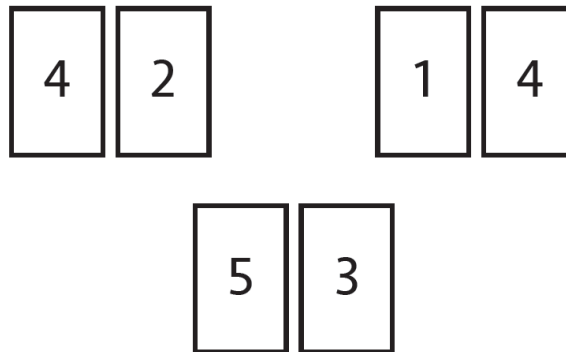
Question 5

What comes next?



Question 6

Which pair of cards adds up to 5?
Circle it!



Question 7

Which shape has the most sides?
Circle it!



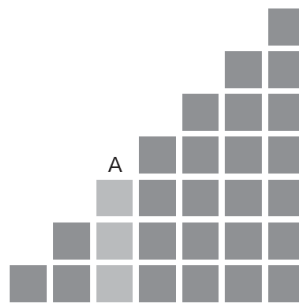
Question 8

Make 6

$$\square + \square + \square = 6$$

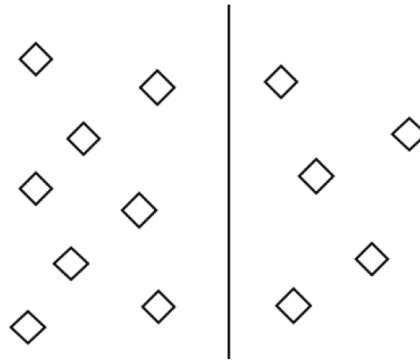
Question 9

How many squares high is step A?



Question 10

Which group is bigger?
Circle it!



Question 11

Count down from 10 by 2's

Question 12

$$\begin{array}{|c|c|c|c|c|} \hline \bullet & & \bullet & \bullet & \\ \hline & \bullet & & \bullet & \\ \hline \end{array} + \boxed{} = \begin{array}{|c|c|c|c|c|} \hline \bullet & \bullet & \bullet & \bullet & \bullet \\ \hline & \bullet & & \bullet & \\ \hline \end{array}$$

Question 13

$$18 - 7 = \boxed{}$$

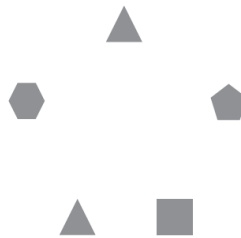
Question 14

How many are there?



Question 15

How many shapes have less than 5 sides?



Question 16

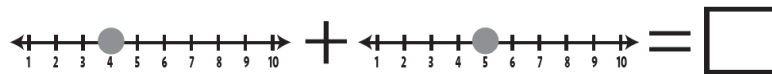
Make 12

Question 17

What numbers are missing?

10, , 6, 4,

Question 18



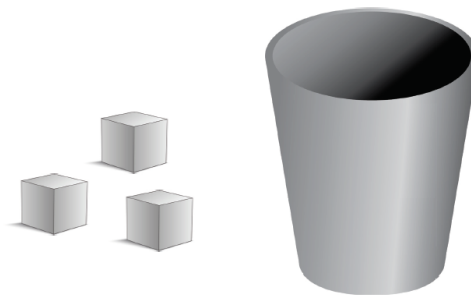
Question 19

Count up to 9 by 3's

Question 20

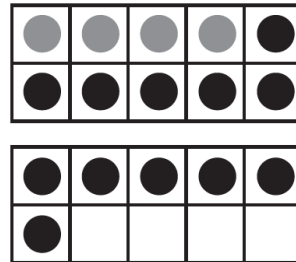
There are 7 cubes.

How many are inside the cup?

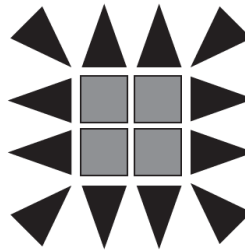


Question 21

We can show $4 + 12$
like this:



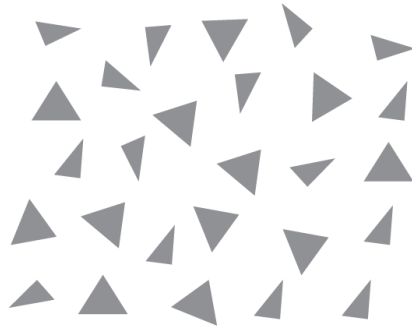
What does this show?



$$\square + \square = \square$$

Question 22

What does this show?



$$\square + \square = \square$$

Question 23

Draw $17 + 6$

A.2 Finger Gnosis Evaluation

This appendix outlines the procedure followed in Section 4.6 to evaluate finger perception. The appendix is written as a tutorial with the steps required to evaluate finger gnosis in children aged five to six years of age. This procedure is based on the one outlined in [80] but separates the simultaneous stimulation as done in [82].

A.2.1 Procedure

1. The first task that the child will perform is drawing a hand:
 - (a) Have the child place his non-dominant hand on his lap, underneath the table.
 - (b) Give the child a blank piece of paper in a clipboard and a marker
 - (c) Have the child draw a hand continuously, without lifting his marker up and without looking at his other hand.
2. Now the student will be asked to identify a number of touches on each hand. The tester will alternate whether students start with the dominant or the non-dominant hand. From now on we will call the first hand students use during the test Hand 1, and the second one Hand 2.
 - (a) First determine the student's handedness by which hand they used to draw the picture of the hand with.
 - (b) Check your sheet to see if you should test the dominant or the non-dominant hand first.
 - (c) Ask the student to face you, arm 1 extended with fingers spread out and the palm down. In order to hide the hand from the child's view, it is placed in a box in which a hole has been made on the child's side to enter the hand, and the examiner's side has been removed.
 - (d) Place an image of the right or left hand (depending on which one is being tested first) for the child to see with the fingers numbered and colored (the examiner also has such an image).

- (e) The child can answer using the number or the color of the finger that has been touched. The child keeps his hand inside of the box for the entirety of the test.
- (f) First perform test with isolated touches:
 - **Script:** *Please put your hand inside of the box. Spread your fingers on the table and don't move it. I'm going to lightly touch your fingers with the paintbrush and all I want you to do is just guess which finger I'm touching. All the guesses are good guesses and all I need you to do is just keep your hand very very still.*
 - i. Run a soft paintbrush from one side of the nail to the other touching a bit of skin as well lightly.
 - ii. Ask the child to name the finger that has been touched without moving his/her hand. The child can answer using the numbers or colors on the sheet in front of him/her.
 - iii. Say *good job!* after each guess, even if they answered incorrectly but record their answer in the scoring sheet. If they move their hand to answer, tell them that they should not move their hand and record that they moved their hand on the scoring sheet. A moved hand nullifies the question.
 - iv. Repeat the procedure 10 times so that each finger is touched 2 times in the order marked on the scoring sheet.
- (g) Then perform test with simultaneous stimulation:
 - **Script:** *Now I am going to touch two fingers at the same time and I want you to tell me which two fingers I touched. Remember to keep your hand very very still!*
 - i. Run two soft paintbrushes from one side of the nail of two fingers to the other touching a bit of skin as well lightly at the same time.
 - ii. Ask the child to name the fingers that has been touched without moving his/her hand. The child can answer using the numbers or colors on the sheet in front of him/her.

- iii. Say *good job!* after each guess, even if they answered incorrectly but record their answer in the scoring sheet. If they move their hand to answer, tell them that they should not move their hand and record that they moved their hand on the scoring sheet. A moved hand nullifies the question.
 - iv. Repeat the procedure 5 times so that each finger is touched 2 times following the order outlined in the scoring sheet.
- (h) Finally perform test with successive touches:
- **Script:** *Now I am going to touch one finger and then another and I want you to tell me which two fingers I touched in the order that I touched them.*
 - i. Run one soft paintbrush from one side of the nail of one fingers to the other touching a bit of skin as well lightly. Immediately after, do the same to another finger as outlined in the scoring sheet.
 - ii. Ask the child to name the fingers that were touched without moving his/her hand in the order that they were touched. The child can answer using the numbers or colors on the sheet in front of him/her.
 - iii. Say *good job!* after each guess, even if they answered incorrectly but record their answer in the scoring sheet. If they move their hand to answer, tell them that they should not move their hand and record that they moved their hand on the scoring sheet. A moved hand nullifies the question.
 - iv. Repeat the procedure 5 times so that each finger is touched 2 times as outlined in the scoring sheet
- (i) Repeat the test for Hand 2 (the hand that has not been tested yet, consult your sheet!).
- (j) For each touch record the students answers
- (k) Give a score of 1 for each task successfully completed (touches correctly identified and in the right order).

- (1) If a child moved his/her hand during the test, record it in the scoring sheet and this question will get a score of 0. The total number of points should be 20 for each hand, or 40 total.

A.2.2 Scoring Sheets

Figure 4.19 shows two testers evaluating a student in finger perception during the pilot study described in Section 4.6. The following sections contain the scoring sheets that were used to keep track of the results of the tests. Fingers are numbered 1-5 where 1 is the thumb and 5 is the little finger. Column 1 details the finger or fingers that should be touched. Column 2, is used to write the student's answer. A student may answer with a number or color to identify the finger. Column 3, is used to identify if the answer was correct (1) or not (0). Throughout the test, the hand should be held completely still, if the child moves the hand in order to identify which finger has been pressed, it is noted in column 3 and that answer is considered incorrect.

Hand 1: Dominant Hand

Subject Number:_____ Gender:_____ Handedness:_____

Birthdate:_____ Time Start of test:_____

Touch	Answer	Result	Move Hand?
Single Touch			
2			
5			
3			
1			
4			
2			
5			
3			
1			
4			
Simultaneous Touch			
1-4			
2-3			
1-5			
2-4			
3-5			
Successive Touch			
4-2			
3-5			
1-4			
2-5			
3-1			

Total Hand 1: _____ Time End of Test:_____

Hand 2: Non-Dominant Hand**Time Start of Test:**_____

Touch	Answer	Result	Move Hand?
Single Touch			
4			
1			
3			
5			
2			
4			
1			
3			
5			
2			
Simultaneous Touch			
3-5			
2-4			
1-5			
2-3			
1-4			
Successive Touch			
3-1			
2-5			
1-4			
3-5			
4-2			

Total Hand 2:_____ **Time End of Test:**_____**Total Points:**_____

Hand 1: Non-Dominant Hand

Subject Number:_____ Gender:_____ Handedness:_____

Birth date:_____ Time Start of test:_____

Touch	Answer	Result	Move Hand?
Single Touch			
2			
5			
3			
1			
4			
2			
5			
3			
1			
4			
Simultaneous Touch			
1-4			
2-3			
1-5			
2-4			
3-5			
Successive Touch			
4-2			
3-5			
1-4			
2-5			
3-1			

Total Hand 1:_____ Time End of Test:_____

Hand 2: Dominant Hand**Time Start of Test:**_____

Touch	Answer	Result	Move Hand?
Single Touch			
4			
1			
3			
5			
2			
4			
1			
3			
5			
2			
Simultaneous Touch			
3-5			
2-4			
1-5			
2-3			
1-4			
Successive Touch			
3-1			
2-5			
1-4			
3-5			
4-2			

Total Hand 2:_____ **Time End of Test:**_____**Total Points:**_____

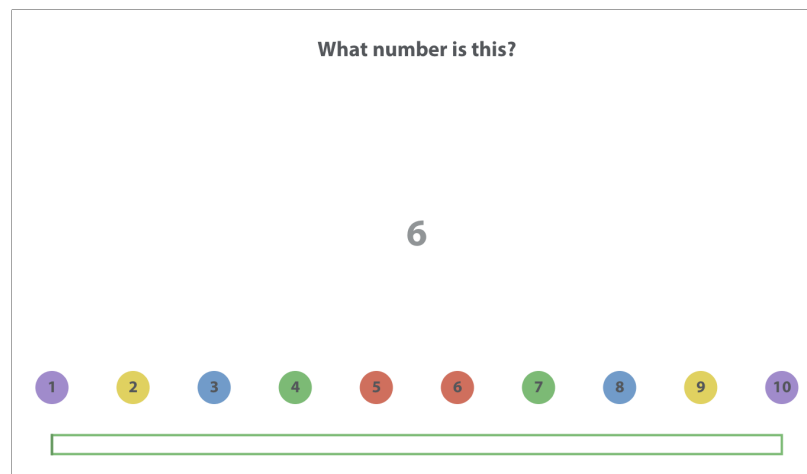
Appendix B

Math Activities

Professor Jo Boaler and her team at youcubed developed a set first-grade-level math activities. There are 28 question types in total. In this Appendix we describe the different types of questions as well as provide an example of each question and how it would be shown in the HapCaps software described in Section ??.

B.1 Identify Number

This question presents the user with a number and asks the student to press the HapCap corresponding to that number. The answer can be a number from 1 to 10.



The screenshot shows a digital interface for a math activity. At the top, the text "What number is this?" is displayed. In the center, the number "6" is shown in a large, gray font. Below the number, there is a horizontal row of ten colored circles, each containing a number from 1 to 10. The colors of the circles are: 1 (purple), 2 (yellow), 3 (blue), 4 (green), 5 (red), 6 (red), 7 (green), 8 (blue), 9 (yellow), and 10 (purple). Below this row of circles is a long, empty rectangular box with a green border, intended for the user to input their answer.

B.2 Identify Number Name on Screen

This question presents the user with the name of a number and asks to identify it. The answer can be a number from 1 to 10.

What number is this?

Seven

1 2 3 4 5 6 7 8 9 10

B.3 Identify Number in Ten Frame

This question presents the user with a number represented in a ten frame and asks to identify it. The answer can be a number from 1 to 10.

What number is this?


	•	•	•	
				•

1 2 3 4 5 6 7 8 9 10

B.4 Identify Number in Number Line

This question presents the user with a number represented in a number line and asks to identify it. The answer can be a number from 1 to 10.

What number is this?



A horizontal number line with arrows at both ends. It has tick marks labeled from 0 to 10. A small dot is placed on the tick mark for the number 8.

1 2 3 4 5 6 7 8 9 10

Below the number line are ten colored circles, each containing a number from 1 to 10. The colors are: 1 (purple), 2 (yellow), 3 (blue), 4 (green), 5 (red), 6 (red), 7 (green), 8 (blue), 9 (yellow), 10 (purple).

Below the circles is a green rectangular input field.

B.5 Single Digit Addition

This question asks the user to calculate the result of the addition of two numbers. The answer is a number from 2 to 10.

Fill in the blank!

$4 + 2 = \underline{\quad}$

1 2 3 4 5 6 7 8 9 10

Below the numbers are ten colored circles, each containing a number from 1 to 10. The colors are: 1 (purple), 2 (yellow), 3 (blue), 4 (green), 5 (red), 6 (red), 7 (green), 8 (blue), 9 (yellow), 10 (purple).

Below the circles is a green rectangular input field.

B.6 Addition with Missing Second Number

This question asks the user to enter the missing addend which, combined with the first, gives the resulting sum. The answer is a number from 1 to 10.

Fill in the blank!

$$2 + \underline{\quad} = 9$$

1 2 3 4 5 6 7 8 9 10

B.7 Simple Subtraction

This question asks the user to calculate the result of the subtraction of two numbers. The answer is a number from 1 to 10.

Fill in the blank!



$$5 - 2 = \underline{\quad}$$

1 2 3 4 5 6 7 8 9 10

B.8 Simple Addition with Ten Frame

This question asks the user to calculate the result of the addition of two numbers represented by ten frames. The answer is a number from 2 to 10.

Fill in the blank!

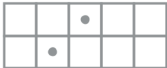


 $+$

 $=$

1
2
3
4
5
6
7
8
9
10

B.9 Addition with Missing Second Number with Ten Frame

This question asks the user to enter the missing addend which, combined with the first, gives the resulting sum. The numbers are represented by ten frames. The answer is a number from 1 to 10.

Fill in the blank!


 $+$ $=$


1
2
3
4
5
6
7
8
9
10

B.10 Simple Addition with Number Line

This question asks the user to calculate the result of the addition of two numbers represented by number lines. The answer is a number from 2 to 10.

Fill in the blank!

$\leftarrow \begin{array}{c} | & | & | & | & | & | & | & | & | & | \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \rightarrow + \leftarrow \begin{array}{c} | & | & | & | & | & | & | & | & | & | \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \rightarrow = \underline{\hspace{1cm}}$

1
2
3
4
5
6
7
8
9
10

B.11 Addition with Missing Second Number with Number Line

This question asks the user to enter the missing addend which, combined with the first, gives the resulting sum. The numbers are represented by number lines. The answer is a number from 1 to 10.

Fill in the blank!


$\leftarrow \begin{array}{c} | & | & | & | & | & | & | & | & | & | \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \rightarrow + \underline{\hspace{1cm}} = \leftarrow \begin{array}{c} | & | & | & | & | & | & | & | & | & | \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \rightarrow$

1
2
3
4
5
6
7
8
9
10

B.12 Number Flexibility

This question asks the user to make a number using as many as ten addends. The student can use numbers between 1 and 10 to answer.

Make 10!




1 2 3 4 5 6 7 8 9 10

B.13 Number Flexibility with Two Numbers

This question asks the user to make a number using two addends. The student can use numbers between 1 and 10 to answer.

Make 10!

___ + ___ = 10



1 2 3 4 5 6 7 8 9 10

B.14 Number Flexibility with Three Numbers

This question asks the user to make a number using three addends. The student can use numbers between 1 and 10 to answer.

Make 10!

$__ + __ + __ = 10$

1 2 3 4 5 6 7 8 9 10

B.15 Counting up to a Number

This question asks the user to count up to a number by 2's or by 3's.


Count up to 10 by 2's!

1 2 3 4 5 6 7 8 9 10

B.16 Counting down from a Number

This question asks the user to count down from a number by 2's or by 3's.


Count down from 10 by 2's!



B.17 Which Group is Bigger

This question asks the user to identify which group of dots is bigger. The dots can be arranged in a semi-random order, in a circle or in a ten-frame. The student answers the question using the color of the bigger group.

Which group is bigger?



B.18 Which Number is Missing

This question presents the user with a number series which can be in increasing or decreasing order and contains one missing element. It asks the student to identify the missing element. The answer can be a number from 1 to 10.

What number is missing?

10, __, 6, 4, 2

1 2 3 4 5 6 7 8 9 10

B.19 What Numbers are Missing

This question presents the user with a number series which can be in increasing or decreasing order and contains two missing elements. It asks the student to identify the missing elements. The answer can be numbers from 1 to 10.

What numbers are missing?

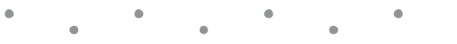
10, __, 6, __, 2

1 2 3 4 5 6 7 8 9 10

B.20 How Many are There

This question asks the user how many figures are on the screen. The figures can be dots or apples and they can be arranged in a semi-random pattern, in a circle, or in a ten frame. The answer can be numbers from 1 to 10.

How many are there?




1 2 3 4 5 6 7 8 9 10

B.21 How Many are Orange

This question presents the user with a set of grey dots mixed with orange dots arranged in a semi-random pattern and asks the user how many of the dots are orange. The answer can be numbers from 1 to 10.

How many are orange?



1 2 3 4 5 6 7 8 9 10

B.22 What Comes Next

This question presents the user with a pattern of figures and asks the user what comes next. The figures can be circles, squares or triangles. The pattern can be AABB, ABAB or ABBA. The student answers with the color of the figure.

What comes next?

▲ ● ● ▲ ▲ ● ● ▲ ▲ ● ● ▲ ▲

1 2 3 4 5 6 7 8 9 10

B.23 How Many Cubes in Cup

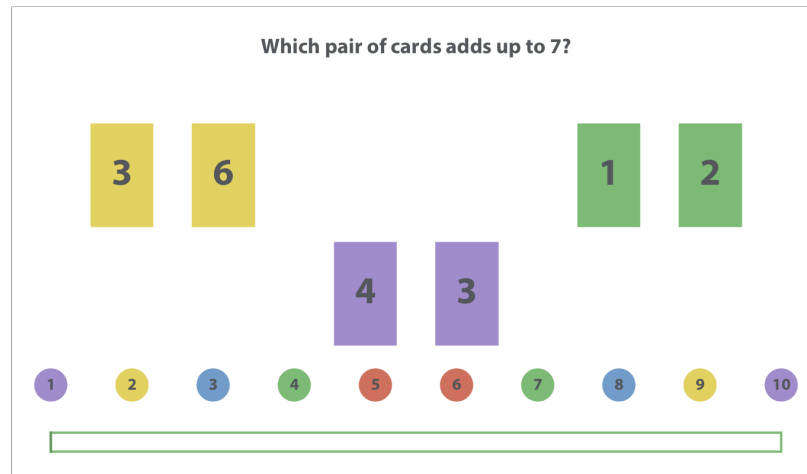
This question presents the user with a cup, a total number of cubes and a set of cubes on the screen. It asks how many cubes are in the cup. The answer can be numbers from 1 to 10.

There are 5 cubes. How many are inside the cup?

■ ■ ■

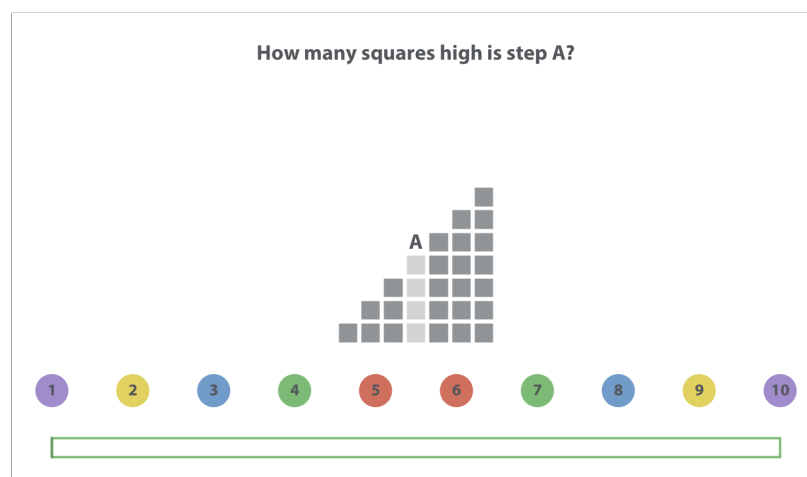
B.24 Pair of Cards Addition

This question presents the user with three pairs of cards of different colors and asks the student which pair of cards adds to the number shown on the screen. The student answers with the color of the pair of cards.



B.25 How Many Steps High


This question presents the user with a pyramid of gray steps where one of the columns has been highlighted. It asks how many steps high is the highlighted column. The answer can be numbers from 1 to 10.



B.26 Which Shape has More Sides

This question presents the user with two polygons of different color and asks the student which one has more sides. The student answers with the color of the polygons.

Which shape has more sides?




A red heptagon (7 sides) and a green diamond (4 sides) are shown side-by-side.

1 2 3 4 5 6 7 8 9 10

B.27 Which Shape has the Most Sides

This question presents the user with three polygons of different colors and asks the student which one has the most sides. The student answers with the color of the polygons.

Which shape has the most sides?




A purple triangle (3 sides), a yellow hexagon (6 sides), and a blue diamond (4 sides) are shown side-by-side.

1 2 3 4 5 6 7 8 9 10

B.28 How Many have More or Less than X Sides

This question presents the user with five polygons arranged in a circle and asks the student how many of the polygons have more or less than a given number of sides. The answer can be a number from 1 to 5.

How many of the shapes have more than 4 sides?



1 2 3 4 5 6 7 8 9 10

Bibliography

- [1] M. Orta Martinez, C. M. Nunez, T. Liao, T. K. Morimoto, and A. M. Okamura, “Evolution and Analysis of Hapkit: An Open-Source Haptic Device for Educational Applications,” *IEEE transactions on haptics*, 2019.
- [2] M. J. Mataric, N. P. Koenig, and D. Feil-Seifer, “Materials for Enabling Hands-On Robotics and STEM Education.” in *AAAI Spring Symposium: Semantic scientific knowledge integration*, 2007, pp. 99–102.
- [3] F. P. Brooks Jr, M. Ouh-Young, J. J. Batter, and P. Jerome Kilpatrick, “Project gropehaptic displays for scientific visualization,” in *ACM SIGGraph computer graphics*, vol. 24, no. 4, 1990, pp. 177–185.
- [4] M. Reiner, “Conceptual construction of fields through tactile interface,” *Interactive Learning Environments*, vol. 7, no. 1, pp. 31–55, 1999.
- [5] C. Dede, M. Salzman, R. B. Loftin, and K. Ash, “The design of immersive virtual learning environments: Fostering deep understandings of complex scientific knowledge.” in *Innovations in Science and Mathematics Education: Advanced Designs for Technologies of Learning*, M. J. Jacobson and R. Kozma, Eds. Lawrence Erlbaum Associates Publishers, 2000, pp. 361–413.
- [6] F. G. Hamza-Lup and M. Adams, “Feel the pressure: E-learning systems with haptic feedback,” in *IEEE Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2008, pp. 445–450.

- [7] G. Hallman, I. Paley, I. Han, and J. B. Black, "Possibilities of haptic feedback simulation for physics learning," in *EdMedia and Innovate Learning*. Association for the Advancement of Computing in Education (AACE), 2009, pp. 3597–3602.
- [8] P. Bivall, S. Ainsworth, and L. A. Tibell, "Do haptic representations help complex molecular learning?" *Science Education*, vol. 95, no. 4, pp. 700–719, 2011.
- [9] K. J. Schönborn, P. Bivall, and L. A. Tibell, "Exploring relationships between students' interaction and learning with a haptic virtual biomolecular model," *Computers & Education*, vol. 57, no. 3, pp. 2095–2105, 2011.
- [10] I. Han and J. B. Black, "Incorporating haptic feedback in simulation for learning physics," *Computers & Education*, vol. 57, no. 4, pp. 2281–2290, 2011.
- [11] M. G. Jones, J. Minogue, T. R. Tretter, A. Negishi, and R. Taylor, "Haptic augmentation of science instruction: Does touch matter?" *Science Education*, vol. 90, no. 1, pp. 111–123, 2006.
- [12] J. Minogue and G. Jones, "Measuring the impact of haptic feedback using the solo taxonomy," *International Journal of Science Education*, vol. 31, no. 10, pp. 1359–1378, 2009.
- [13] M. G. Jones, T. Andre, R. Superfine, and R. Taylor, "Learning at the nanoscale: The impact of students' use of remote microscopy on concepts of viruses, scale, and microscopy," *Journal of Research in Science Teaching*, vol. 40, no. 3, pp. 303–322, 2003.
- [14] J. Minogue, M. G. Jones, B. Broadwell, and T. Oppewall, "The impact of haptic augmentation on middle school students' conceptions of the animal cell," *Virtual Reality*, vol. 10, no. 3-4, pp. 293–305, 2006.
- [15] E. N. Wiebe, J. Minogue, M. G. Jones, J. Cowley, and D. Krebs, "Haptic feedback and students' learning about levers: Unraveling the effect of simulated touch," *Computers & Education*, vol. 53, no. 3, pp. 667–676, 2009.

- [16] D. Moore, R. L. Williams II, T. Luo, and E. Karadogan, “Elusive achievement effects of haptic feedback,” *Journal of Interactive Learning Research*, vol. 24, no. 3, pp. 329–347, 2013.
- [17] Z. C. Zacharia, “Examining whether touch sensory feedback is necessary for science learning through experimentation: A literature review of two different lines of research across k-16,” *Educational Research Review*, vol. 16, pp. 116–137, 2015.
- [18] M. Ouh-Young, M. Pique, J. Hughes, N. Srinivasan, and F. Brooks, “Using a manipulator for force display in molecular docking,” in *IEEE International Conference on Robotics and Automation*, 1988, pp. 1824–1829.
- [19] O.-Y. Ming, D. V. Beard, and F. P. Brooks, “Force display performs better than visual display in a simple 6-d docking task,” in *IEEE International Conference on Robotics and Automation*, 1989, pp. 1462–1466.
- [20] M. Ouh-Young, “Force display in molecular docking,” Ph.D. dissertation, Dept. of Computer Science, University of North Carolina, Chapel Hill, 1991.
- [21] F. L. Engel, P. Goossens, and R. Haakma, “Improved efficiency through i-and e-feedback: A trackball with contextual force feedback,” *International Journal of Human-Computer Studies*, vol. 41, no. 6, pp. 949–974, 1994.
- [22] D. Keyson, “Dynamic control gain and tactile feedback in the capture of cursor movements,” *IPO Annual Progress Report*, vol. 29, pp. 101–108, 1994.
- [23] 3D Systems Touch (Previously Immersion Phantom Omni). Accessed: 08.11.2019. [Online]. Available: <https://www.3dsystems.com/haptics-devices/touch>
- [24] 3D Systems Touch X (Previously Immersion Phantom Desktop). Accessed: 08.11.2019. [Online]. Available: <https://www.3dsystems.com/haptics-devices/touch-x>

- [25] M. Sato, X. Liu, J. Murayama, K. Akahane, and M. Isshiki, “A Haptic Virtual Environment for Molecular Chemistry Education,” in *Transactions on Edutainment*. Springer, 2008, pp. 28–39.
- [26] 3D Systems Phantom Premium (Previously Sensable Phantom Premium). Accessed: 08.11.2019. [Online]. Available: <https://www.3dsystems.com/haptics-devices/3d-systems-phantom-premium>
- [27] Logitech Force Feedback Joystick. Accessed: 08.11.2019. [Online]. Available: <https://www.logitechg.com/en-us/products/space.html>
- [28] Open Source Hardware Association. Accessed: 01.19.2019. [Online]. Available: <https://www.oshwa.org/>
- [29] R. Acosta, “Open Source Hardware,” Master’s thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2009.
- [30] Arduino. Accessed: 01.30.2019. [Online]. Available: <https://www.arduino.cc/>
- [31] M. Rubenstein, A. Cornejo, and R. Nagpal, “Programmable self-assembly in a thousand-robot swarm,” *Science*, vol. 345, no. 6198, pp. 795–799, 2014.
- [32] Soft Robotics Toolkit. Accessed: 01.30.2019. [Online]. Available: <http://softroboticstoolkit.com/>
- [33] RepRap. Accessed: 01.30.2019. [Online]. Available: <http://reprap.org/>
- [34] OPEN ROBOT HARDWARE: An Initiative for Open Robot Hardware. Accessed: 01.30.2019. [Online]. Available: <http://www.openrobothardware.org/>
- [35] C. Ramstein and V. Hayward, “The pantograph: A large workspace haptic device for a multi-modal human-computer interaction,” in *CHI’94 Conference on Human Factors in Computing Systems*, 1994, pp. 57–58.
- [36] C. Richard, A. M. Okamura, and M. R. Cutkosky, “Getting a feel for dynamics: Using haptic interface kits for teaching dynamics and controls,” in *ASME IMECE 6th Annual Symposium on Haptic Interfaces*, 1997, pp. 15–21.

- [37] TPad Tablet Project: A handheld haptic device with a variable friction surface. Accessed: 01.30.2019. [Online]. Available: <http://tpadtablet.org/>
- [38] M. I. Norton, D. Mochon, and D. Ariely, “The IKEA effect: When labor leads to love,” *Journal of Consumer Psychology*, vol. 22, no. 3, pp. 453–460, 2012.
- [39] M. Orta Martinez, T. K. Morimoto, A. T. Taylor, A. C. Barron, J. D. A. Pultorak, J. Wang, A. Calasanz-Kaiser, R. Lee Davis, P. Blikstein, and A. M. Okamura, “3-D Printed Haptic Devices for Educational Applications,” in *IEEE Haptics Symposium*, 2016, pp. 126–133.
- [40] R. L. Davis, M. Orta Martinez, O. Schneider, K. E. MacLean, A. M. Okamura, and P. Blikstein, “The haptic bridge: Towards a theory for haptic-supported learning,” in *Proceedings of the 2017 Conference on Interaction Design and Children*. ACM, 2017, pp. 51–60.
- [41] D. I. Grow, L. N. Verner, and A. M. Okamura, “Educational haptics,” in *AAAI 2007 Spring Symposium: Semantic Scientific Knowledge Integration*, 2007, pp. 53–58.
- [42] K. Bowen and M. K. O’Malley, “Adaptation of haptic interfaces for a labview-based system dynamics course,” in *IEEE Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2006, pp. 147–152.
- [43] J. L. Gorlewicz, L. B. Kratchman, and R. J. Webster III, “Haptic Paddle Enhancements and a Formal Assessment of Student Learning in System Dynamics,” *Advances in Engineering Education*, vol. 4, no. 2, pp. 1–31, 2014.
- [44] C. G. Rose, J. A. French, and M. K. O’Malley, “Design and characterization of a haptic paddle for dynamics education,” in *IEEE Haptics Symposium*, 2014, pp. 265–270.
- [45] A. Okamura, C. Richard, and M. Cutkosky, “Feeling is believing: Using a force-feedback joystick to teach dynamic systems,” *Engineering Education*, vol. 91, no. 3, pp. 345–349, 2002.

- [46] R. Gassert, J.-C. Metxger, K. Leuenberger, W. L. Popp, M. R. Tucker, B. Vigarù, R. Zimmermann, and O. Lambercy, “Physical student-robot interaction with the ETHZ haptic paddle,” *IEEE Transactions on Education*, vol. 56, no. 1, pp. 9–17, 2013.
- [47] A. Otaran, O. Tokatli, and V. Patoglu, “Hands-On Learning with a Series Elastic Educational Robot,” in *EuroHaptics Conference*, 2016, pp. 3–16.
- [48] T. K. Morimoto, A. Miller, P. Blikstein, and A. M. Okamura, “The Haptic Paddle: A Low-Cost Haptic Device for Online Education,” in *Open Hardware Summit*, 2013.
- [49] R. B. Gillespie, M. B. Hoffman, and J. Freudenberg, “Haptic Interface for Hands-On Instruction in System Dynamics and Embedded Control,” in *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2003, pp. 410–415.
- [50] B. Verplank, M. Gurevich, and M. Mathews, “THE PLANK: Designing a simple haptic controller,” in *Conference on New Instruments for Musical Expression*, 2002, pp. 1–4.
- [51] Hapkit: Open-hardware haptic device. Accessed: 01.30.2019. [Online]. Available: <http://hapkit.stanford.edu/>
- [52] G. Minaker, O. Schneider, R. Davis, and K. E. MacLean, “HandsOn: Enabling Embodied, Creative STEM e-learning with Programming-Free Force Feedback,” in *EuroHaptics Conference*, 2016, pp. 427–437.
- [53] C. Richard, A. M. Okamura, and M. R. Cutkosky, “Getting a feel for dynamics: Using haptic interface kits for teaching dynamics and controls,” in *ASME IMECE 6th Annual Symposium on Haptic Interfaces*, 1997, pp. 15–21.
- [54] N. K. Shimbun, *Poka-yoke: Improving product quality by preventing defects*. Crc Press, 1989.

- [55] Biomechatronics Lab at Colorado School of Mines, Haptic Paddle. Accessed: 01.27.2016. [Online]. Available: <http://inside.mines.edu/~ocelik/research.html>
- [56] Seeed Hapkit Board. Accessed: 01.29.2019. [Online]. Available: <https://www.seeedstudio.com/Hapkit-Board-p-1622.html>
- [57] J. J. Abbott, “Virtual fixtures for bilateral telemanipulation,” Ph.D. dissertation, Dept. Of Mechanical Engineering, The Johns Hopkins University, 2005.
- [58] C. G. Rose, J. A. French, and M. K. OMalley, “Design and characterization of a haptic paddle for dynamics education,” in *IEEE Haptics Symposium*, 2014, pp. 265–270.
- [59] R. Steidel, *An Introduction to Mechanical Vibrations*. Prentice Hall, 1993.
- [60] Z. Zhu, “A simple method for measuring cogging torque in permanent magnet machines,” in *IEEE Power & Energy Society General Meeting*, 2009, pp. 1–4.
- [61] G. Gescheider, *Psychophysics: The Fundamentals*, 3rd ed. Lawrence Erlbaum Associates, 1997.
- [62] C. G. Rose, C. G. McDonald, J. P. Clark, and M. K. O’Malley, “Reflection on system dynamics principles improves student performance in haptic paddle labs,” *IEEE Transactions on Education*, vol. 61, no. 3, pp. 245–252, 2018.
- [63] C. E. Wong and A. M. Okamura, “The Snaptic Paddle: A Modular Haptic Device,” in *IEEE World Haptics Conference*, 2005, pp. 537–538.
- [64] M. Orta Martinez, J. Campion, T. Gholami, M. K. Rittikaidachar, A. C. Barron, and A. M. Okamura, “Open Source, Modular, Customizable, 3-D Printed Kineshetic Haptic Devices,” in *IEEE World Haptics Conference*, 2017, pp. 142–147.
- [65] J. Forsslund, M. Yip, and E.-L. Sallnäs, “Woodenhaptics: A starting kit for crafting force-reflecting spatial haptic devices,” in *Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, 2015, pp. 133–140.

- [66] G. Campion, Q. Wang, and V. Hayward, “The Pantograph Mk-II: A Haptic Instrument,” in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2005, pp. 723–728.
- [67] J. Siira and D. K. Pai, “Haptic texturing-a stochastic approach,” in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, 1996, pp. 557–562.
- [68] G. Robles-De-La-Torre and V. Hayward, “Virtual surfaces and haptic shape perception,” in *Proceedings ASME IMECE Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, vol. 69, 2000, pp. 2–7.
- [69] S. M. Sketch, D. R. Deo, J. P. Menon, and A. M. Okamura, “Design and experimental evaluation of a skin-stretch haptic device for improved control of brain-computer interfaces,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 272–277.
- [70] R. B. Gillespie and A. M. Okamura, *Haptic Interaction for Hands-On Learning in System Dynamics and Controls*, 2008 (Accessed 05.19.2020). [Online]. Available: <http://www-personal.umich.edu/~brentg/Publications/Journal/HapticEducationCamera.pdf>
- [71] C. Gallacher, A. Mohtat, S. Ding, and J. Kövecses, “Toward open-source portable haptic displays with visual-force-tactile feedback colocation,” in *IEEE Haptics Symposium*, 2016, pp. 65–71.
- [72] Arduino Uno Rev3. Accessed: 08.07.2019. [Online]. Available: <https://store.arduino.cc/usa/arduino-uno-rev3>
- [73] Nucleo-F446ZE Development Board. Accessed: 08.07.2019. [Online]. Available: <https://www.st.com/en/evaluation-tools/nucleo-f446ze.html>
- [74] Processing. Accessed: 08.02.2019. [Online]. Available: <http://www.processing.org>

- [75] Arduino IDE. Accessed: 08.07.2019. [Online]. Available: <https://www.arduino.cc/en/main/software>
- [76] Arm MBED. Accessed: 08.07.2019. [Online]. Available: <https://www.mbed.com>
- [77] P. Mitiguy, *Advanced Dynamics & Motion Simulation*, 2013.
- [78] *MATLAB*, The Mathworks, Inc., Natick, Massachusetts.
- [79] N. Colonnese and A. M. Okamura, “Propagation of joint space quantization error to operational space coordinates and their derivatives,” in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2054–2061.
- [80] M.-P. Noel, “Finger gnosis: A predictor of numerical abilities in children?” *Child Neuropsychology*, vol. 11, pp. 413–430, 2005.
- [81] M. Kinsbourne and E. K. Warrington, “The Development of Finger Differentiation,” *Quarterly Journal of Experimental Psychology*, vol. 15, no. 2, pp. 132–137, 1963.
- [82] M. Gracia-Bafalluy and M.-P. Noel, “Does finger training increase young children’s numerical performance?” *Cortex (Special Issue on Numbers, Space, and Action)*, vol. 44, no. 4, pp. 368–375, 2008.
- [83] B. Butterworth, *The Mathematical Brain*. Macmillan, 1999.
- [84] T. Jay and J. Betenson, “Mathematics at your fingertips: testing a finger training intervention to improve quantitative skills,” in *Frontiers in Education*, vol. 2. Frontiers, 2017, pp. 22–30.
- [85] J. Mullenbach, C. Shultz, A. M. Piper, M. A. Peshkin, and J. E. Colgate, “Tpad fire: Surface haptic tablet,” in *HAID: Haptic and Audio Interaction Design*, 2013 (Accessed 05.19.2020). [Online]. Available: https://inclusive.northwestern.edu/Mullenbach_HAID13-Poster.pdf
- [86] OpenGlove: Haptics easy to build, develop and integrate. Accessed: 05.14.2020. [Online]. Available: www.openglove.org

- [87] B. Hightower, S. Lovato, J. Davison, E. Wartella, and A. M. Piper, “Haptic explorers: Supporting science journaling through mobile haptic feedback displays,” *International Journal of Human-Computer Studies*, vol. 122, pp. 103–112, 2019.
- [88] E. Beheshti, K. Borgos-Rodriguez, and A. M. Piper, “Supporting parent-child collaborative learning through haptic feedback displays,” in *Proceedings of the 18th ACM International Conference on Interaction Design and Children*, 2019, pp. 58–70.
- [89] Tanvas. Accessed: 07.30.2019. [Online]. Available: <http://tanvas.co/>
- [90] C. N. Riviere, W. T. Ang, and P. K. Khosla, “Toward active tremor canceling in handheld microsurgical instruments,” *IEEE Transactions on Robotics and Automation*, vol. 19, no. 5, pp. 793–800, 2003.
- [91] Scratch. Accessed: 08.14.2019. [Online]. Available: <https://scratch.mit.edu>
- [92] O. S. Schneider and K. E. MacLean, “Studying design process and example use with Macaron, a web-based vibrotactile effect editor,” in *IEEE Haptics Symposium*, 2016, pp. 52–58.
- [93] Open Source Software Foundation. Accessed: 05.12.2020. [Online]. Available: <http://www.opensource.com>
- [94] J. Piaget, B. Inhelder, and A. Szeminska, *Child’s conception of geometry*. Routledge, 1960, vol. 81.