University of Pennsylvania
ScholarlyCommons

Publicly Accessible Penn Dissertations

2018

Instrumentation, Data, And Algorithms For Visually Understanding Haptic Surface Properties

Alexander Burka
University of Pennsylvania, aburka@seas.upenn.edu

Follow this and additional works at: https://repository.upenn.edu/edissertations

Part of the Robotics Commons

Recommended Citation
Burka, Alexander, "Instrumentation, Data, And Algorithms For Visually Understanding Haptic Surface Properties" (2018). *Publicly Accessible Penn Dissertations*. 3099.
https://repository.upenn.edu/edissertations/3099

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/edissertations/3099
For more information, please contact repository@pobox.upenn.edu.

Instrumentation, Data, And Algorithms For Visually Understanding Haptic Surface Properties

Abstract
Autonomous robots need to efficiently walk over varied surfaces and grasp diverse objects. We hypothesize that the association between how such surfaces look and how they physically feel during contact can be learned from a database of matched haptic and visual data recorded from various end-effectors' interactions with hundreds of real-world surfaces. Testing this hypothesis required the creation of a new multimodal sensing apparatus, the collection of a large multimodal dataset, and development of a machine-learning pipeline.

This thesis begins by describing the design and construction of the Portable Robotic Optical/Tactile ObservatioN PACKage (PROTONPACK, or Proton for short), an untethered handheld sensing device that emulates the capabilities of the human senses of vision and touch. Its sensory modalities include RGBD vision, egomotion, contact force, and contact vibration. Three interchangeable end-effectors (a steel tooling ball, an OptoForce three-axis force sensor, and a SynTouch BioTac artificial fingertip) allow for different material properties at the contact point and provide additional tactile data.

We then detail the calibration process for the motion and force sensing systems, as well as several proof-of-concept surface discrimination experiments that demonstrate the reliability of the device and the utility of the data it collects. This thesis then presents a large-scale dataset of multimodal surface interaction recordings, including 357 unique surfaces such as furniture, fabrics, outdoor fixtures, and items from several private and public material sample collections. Each surface was touched with one, two, or three end-effectors, comprising approximately one minute per end-effector of tapping and dragging at various forces and speeds. We hope that the larger community of robotics researchers will find broad applications for the published dataset.

Lastly, we demonstrate an algorithm that learns to estimate haptic surface properties given visual input. Surfaces were rated on hardness, roughness, stickiness, and temperature by the human experimenter and by a pool of purely visual observers. Then we trained an algorithm to perform the same task as well as infer quantitative properties calculated from the haptic data. Overall, the task of predicting haptic properties from vision alone proved difficult for both humans and computers, but a hybrid algorithm using a deep neural network and a support vector machine achieved a correlation between expected and actual regression output between approximately $\rho = 0.3$ and $\rho = 0.5$ on previously unseen surfaces.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Electrical & Systems Engineering

This dissertation is available at ScholarlyCommons: https://repository.upenn.edu/edissertations/3099

First Advisor
Katherine J. Kuchenbecker

Subject Categories
Robotics

This dissertation is available at ScholarlyCommons: https://repository.upenn.edu/edissertations/3099

INSTRUMENTATION, DATA, AND ALGORITHMS FOR VISUALLY UNDERSTANDING HAPTIC SURFACE PROPERTIES

Alexander L. Burka

A DISSERTATION

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2018

Supervisor of Dissertation

Katherine J. Kuchenbecker
Associate Professor, Department of Mechanical Engineering and Applied Mechanics

Graduate Group Chairperson

Alejandro Ribeiro
Associate Professor, Department of Electrical and Systems Engineering

Dissertation Committee

Jan Van der Spiegel
Professor of Electrical and Systems Engineering

David Brainard
RRL Professor of Psychology

Johannes Burge
Assistant Professor of Psychology

Trevor Darrell
Professor in Residence of Computer Science
University of California, Berkeley

INSTRUMENTATION, DATA, AND ALGORITHMS FOR VISUALLY UNDERSTANDING HAPTIC SURFACE PROPERTIES

© COPYRIGHT

2018

Alexander Lawrence Burka

This work is licensed under the Creative Commons Attribution NonCommercial-ShareAlike 3.0 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/.

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of the University of Pennsylvania's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Updated 2018-07-23 20:43:13-04:00.

ACKNOWLEDGMENTS

The road to finishing this thesis was long and winding, and I could not have done it alone. First, thank you to my advisor, Katherine Kuchenbecker, for finding space for me in her research group, and for her supportive and patient mentoring, excellence in research, and meticulous paper editing. Thank you to my previous advisor, Dan Lee, for supporting me in the beginning of my time at Penn and encouraging me to pursue further research. Thank you as well to the other members of my thesis committee, Jan Van der Spiegel, David Brainard, Johannes Burge, and Trevor Darrell, for helping guide my thesis from proposal to this document. Thank you to my family and friends, especially Laura, for keeping me relatively sane and believing (even when I didn't) that I could get to this point.

Many people contributed to the Proton Pack project. Here is a list of some of them. The design phase involved Siyao Hu, Shweta Krishnan, Stu Helgeson, and Conor O'Brien, as well as our collaborators at UC Berkeley, Lisa Anne Hendricks, Yang Gao, and Trevor Darrell. Jennifer Hui, Naomi Fitter, and Rebecca Khurshid provided helpful advice in the early stages of the project. Sarah Allen and Abhinav Rajvanshi wrote important software components during their internships.

We would not have such a large dataset without the help of Eric Young and Myles Cai, who collected data with the Proton Pack, as well as Heather Culbertson, whose surface samples were the first entries in our dataset, Michaela Wieland, who collected materials at the Max Planck Institute and shipped them across the ocean, Matti Strese, who not only welcomed me into his lab at TU Munich to gather data but loaned me a number of surface samples, and Willow Dressel, who arranged access to the Material ConneXion library at Princeton.

Lastly, I need to acknowledge Dougal Sutherland and Antal Spector-Zabusky, who improved this manuscript by answering my endless questions about statistics and LaTeX, respectively.

It's been a great opportunity to work in the GRASP Lab, particularly with the Haptics Group both at Penn and at MPI. During my PhD I was supported by the NSF under the Complex Scene Perception IGERT program, award number 0966142, later under grant 1426787 as part of the National Robotics Initiative, and finally by the Max Planck Institute.

ABSTRACT

INSTRUMENTATION, DATA, AND ALGORITHMS FOR VISUALLY UNDERSTANDING HAPTIC SURFACE PROPERTIES

Alexander L. Burka

Katherine J. Kuchenbecker

Autonomous robots need to efficiently walk over varied surfaces and grasp diverse objects. We hypothesize that the association between how such surfaces look and how they physically feel during contact can be learned from a database of matched haptic and visual data recorded from various end-effectors' interactions with hundreds of real-world surfaces. Testing this hypothesis required the creation of a new multimodal sensing apparatus, the collection of a large multimodal dataset, and development of a machine-learning pipeline.

This thesis begins by describing the design and construction of the Portable Robotic Optical/Tactile ObservatioN PACKage (PROTONPACK, or Proton for short), an untethered handheld sensing device that emulates the capabilities of the human senses of vision and touch. Its sensory modalities include RGBD vision, egomotion, contact force, and contact vibration. Three interchangeable end-effectors (a steel tooling ball, an OptoForce three-axis force sensor, and a SynTouch BioTac artificial fingertip) allow for different material properties at the contact point and provide additional tactile data.

We then detail the calibration process for the motion and force sensing systems, as well as several proof-of-concept surface discrimination experiments that demonstrate the reliability of the device and the utility of the data it collects. This thesis then presents a large-scale dataset of multimodal surface interaction recordings, including 357 unique surfaces such as furniture, fabrics, outdoor fixtures, and items from several private and public material sample collections. Each surface was touched with one, two, or three end-effectors, comprising approximately one minute per end-effector of tapping and dragging at various forces

and speeds. We hope that the larger community of robotics researchers will find broad applications for the published dataset.

Lastly, we demonstrate an algorithm that learns to estimate haptic surface properties given visual input. Surfaces were rated on hardness, roughness, stickiness, and temperature by the human experimenter and by a pool of purely visual observers. Then we trained an algorithm to perform the same task as well as infer quantitative properties calculated from the haptic data. Overall, the task of predicting haptic properties from vision alone proved difficult for both humans and computers, but a hybrid algorithm using a deep neural network and a support vector machine achieved a correlation between expected and actual regression output between approximately $\rho = 0.3$ and $\rho = 0.5$ on previously unseen surfaces.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iii

ABSTRACT v

CONTENTS vii

LIST OF TABLES ix

LIST OF CODE EXCERPTS x

LIST OF FIGURES xi

CHAPTER 1 INTRODUCTION 1

1.1 Motivation 2

1.2 Contributions 3

CHAPTER 2 PROTON PACK HARDWARE 7

2.1 Background 7

2.2 Architectural Considerations 12

2.3 Handheld Rig 14

2.4 Backpack 23

2.5 Summary 29

CHAPTER 3 SOFTWARE 30

3.1 Data Collection 30

3.2 Postprocessing 42

3.3 Summary 50

CHAPTER 4 SURFACE DISCRIMINATION WITH ENGINEERED FEATURES . 51

4.1 Background 52

4.2 Types of Features 52

4.3 Scan-Dependent Feature Pipeline . . . 53
4.4 Initial Proof-of-Concept Experiment . . . 55
4.5 Design Improvements . . . 60
4.6 Comparison with Scan-Free Features . . . 72
4.7 Summary . . . 85

CHAPTER 5 DATASET . . . 86
5.1 Description of Data . . . 88
5.2 Episode Length Experiment . . . 96
5.3 Data Format . . . 105
5.4 Summary . . . 108

CHAPTER 6 VISUO-HAPTIC MACHINE LEARNING . . . 109
6.1 Background . . . 110
6.2 Task Definition . . . 113
6.3 Human-Subject Baseline Study . . . 114
6.4 Machine Learning . . . 119
6.5 Summary . . . 148

CHAPTER 7 CONCLUSION . . . 150
7.1 Contributions . . . 151
7.2 Future Directions . . . 154

APPENDIX A CALIBRATION . . . 157

APPENDIX B SURFACE LIST . . . 167

APPENDIX C TERMINOLOGY . . . 186

BIBLIOGRAPHY . . . 188

LIST OF TABLES

TABLE 2.1 Human/robot operator pros and cons 13

TABLE 3.1 Format of `structure_times.csv` and `bluefox_times.csv` files . . . 43
TABLE 3.2 Format of `april.csv` files 43
TABLE 3.3 Format of `teensy.ft.csv` files 44
TABLE 3.4 Format of `teensy.acc.csv` files 44
TABLE 3.5 Format of `teensy.gyro.csv` files 44
TABLE 3.6 Format of `teensy.mag.csv` files 44
TABLE 3.7 Format of `optoforce.csv` files 44
TABLE 3.8 Format of `biotac.csv` files 45

TABLE 4.1 Best hyperparameters in initial proof-of-concept 59
TABLE 4.2 Model performance in initial proof-of-concept 60
TABLE 4.3 End-effector diameter of various haptic sensing systems 64
TABLE 4.4 End-effector diameter comparison under Vicon tracking 65
TABLE 4.5 Test set performance of small end-effector with Vicon tracking 66
TABLE 4.6 Motion tracking algorithm comparison 69
TABLE 4.7 Optimal classifiers by end-effector and tracking method 71
TABLE 4.8 Performance of the 9.525 mm end-effector with onboard tracking . . . 71
TABLE 4.9 Classification accuracies using scan-dependent and scan-free features 81

TABLE 6.1 Automatic peak-finding parameters 124
TABLE 6.2 Train/test split 126
TABLE 6.3 Cross-validation learning results (without feature selection) 130
TABLE 6.4 Final learning results (without feature selection) 131
TABLE 6.5 Cross-validation learning results (without feature selection) 132

TABLE 6.6 Final learning results (without feature selection) 133
TABLE 6.7 Human rating performance metrics 148

TABLE A.1 Calibration results. 165

LIST OF CODE EXCERPTS

3.1 BioTac flow script 41
3.2 BioTac flow record 41
3.3 Pose transformation 49
3.4 Force/torque transformation 49
5.1 Loading and plotting data in MATLAB 108

LIST OF FIGURES

FIGURE 2.1 Strain gauge 9
FIGURE 2.2 BioTac sensor internals 10
FIGURE 2.3 Sensors on the handheld rig. 15
FIGURE 2.4 Proton operation 16
FIGURE 2.5 End-effector with high-bandwidth accelerometers 18
FIGURE 2.6 Vicon marker fixture 20
FIGURE 2.7 Proton Pack end-effectors 21
FIGURE 2.8 Wiring diagram of rig and backpack 25
FIGURE 2.9 Schematic of power distribution circuit 26
FIGURE 2.10 Schematic of Teensy daughterboard and peripherals 27
FIGURE 2.11 Schematic of ATI Mini40 conversion circuit 27
FIGURE 2.12 Schematic of analog sensor acquisition circuit 28
FIGURE 2.13 Two-layer PCB layout 28

FIGURE 3.1 Modular software architecture 31
FIGURE 3.2 Teensy packet structure 36
FIGURE 3.3 Smartphone web app 38
FIGURE 3.4 Flowchart showing steps for gathering a dataset 38
FIGURE 3.5 Coordinate frames 48

FIGURE 4.1 Coordinate frames 54
FIGURE 4.2 Surface samples used in the 2016 proof-of-concept experiment . . . 56

FIGURE 4.3 Sample data recording from the 2016 proof-of-concept experiment . 57
FIGURE 4.4 Scan-dependent feature pipeline 58
FIGURE 4.5 Confusion matrix from initial proof-of-concept 59
FIGURE 4.6 6.35 mm tooling ball end-effector with external accelerometers . . . 61
FIGURE 4.7 Surface samples used in the 2017 validation experiment 62
FIGURE 4.8 Comparison of accelerometer mounting positions 62
FIGURE 4.9 Confusion matrix for small end-effector with Vicon tracking 65
FIGURE 4.10 Fiducial frame 67
FIGURE 4.11 Graphical comparison of motion tracking systems 70
FIGURE 4.12 Confusion matrix for 9.525 mm end-effector with onboard tracking . 71
FIGURE 4.13 Confusion matrix for LMT classification using scan-free features . . 79
FIGURE 4.14 Comparison of classifiers using scan-dependent and scan-free features 82
FIGURE 4.15 Feature sensitivity to scan-time parameters 84

FIGURE 5.1 Proton Pack operation 90
FIGURE 5.2 Trace of end-effector position during a typical surface recording . . 92
FIGURE 5.3 Data gathered during a typical surface recording 93
FIGURE 5.4 End-effector dragging force and speed histograms 94
FIGURE 5.5 End-effector tapping force histograms 94
FIGURE 5.6 Inputs and outputs of bilinear friction model 100
FIGURE 5.7 Coefficients of online learned friction models 101
FIGURE 5.8 Model error of online learned friction models 102
FIGURE 5.9 Evaluation of online learned friction model at stopping point 103
FIGURE 5.10 Normal force histogram 104
FIGURE 5.11 Tangential force histogram 104
FIGURE 5.12 MATLAB dataset visualizer 106
FIGURE 5.13 Freeze-frame of pinewood interaction video 107

FIGURE 6.1 Moravec's paradox as illustrated by XKCD 110

FIGURE 6.2 Top-5 error rates on ILSVRC . . . 112
FIGURE 6.3 Some surface images in the human-subject study dataset. . . . 115
FIGURE 6.4 MTurk participant interface, showing a cork surface. . . . 115
FIGURE 6.5 Correlation matrices from the human-subject study . . . 117
FIGURE 6.6 Selected surface images with scan-free features . . . 121
FIGURE 6.7 Box plots of scan-free features aggregated over material categories . 122
FIGURE 6.8 T-CNN architecture . . . 125
FIGURE 6.9 Feature selection performance . . . 128
FIGURE 6.10 Haptic property correlation plots in CV training (no FS) . . . 134
FIGURE 6.11 Haptic property correlation plots in CV testing (no FS) . . . 135
FIGURE 6.12 Haptic property correlation plots in training (no FS) . . . 136
FIGURE 6.13 Haptic property correlation plots in testing (no FS) . . . 137
FIGURE 6.14 Haptic property correlation plots in CV training (with FS) . . . 138
FIGURE 6.15 Haptic property correlation plots in CV testing (with FS) . . . 139
FIGURE 6.16 Haptic property correlation plots in training (with FS) . . . 140
FIGURE 6.17 Haptic property correlation plots in testing (with FS) . . . 141
FIGURE 6.18 Human rating confusion matrices in CV training (no FS) . . . 142
FIGURE 6.19 Human rating confusion matrices in CV testing (no FS) . . . 142
FIGURE 6.20 Human rating confusion matrices in training (no FS) . . . 143
FIGURE 6.21 Human rating confusion matrices in testing (no FS) . . . 143
FIGURE 6.22 Human rating confusion matrices in CV training (with FS) . . . 144
FIGURE 6.23 Human rating confusion matrices in CV testing (with FS) . . . 144
FIGURE 6.24 Human rating confusion matrices in training (with FS) . . . 145
FIGURE 6.25 Human rating confusion matrices in testing (with FS) . . . 145

FIGURE A.1 Coordinate frames . . . 158
FIGURE A.2 Vicon calibration fixture . . . 159
FIGURE A.3 Vicon calibration sphere fit . . . 160
FIGURE A.4 Gravity compensation calibration setup . . . 162

FIGURE A.5 Free-body diagram 162
FIGURE A.6 Gravity compensation calibration sphere fit 163

CHAPTER 1 : INTRODUCTION

The world is covered in surfaces. In going about our daily lives, we interact with many different materials and have an intuitive understanding of how to do so. Cooking breakfast, we pick up a raw egg with a precisely calibrated grip force to neither crush nor drop it, and avoid touching the red-hot stove burner. Getting into the shower, we step on a rug or hold a handrail because wet tiles are slippery. Venturing outside, we don shoes with rubber soles (high coefficient of friction with most materials) and step around reflective patches of the sidewalk, which indicate puddles or ice patches.

These finely calibrated physical interactions are the result of lifelong learning. Humans aren't born with a database of material properties and interactions, but by exploring the world in childhood we amass a repertoire of experience with different objects and environments, allowing us to extrapolate typical material properties. Later, encountered objects can either be interpreted as a member of a known class, in which case the properties are remembered, or viewed as novel but with properties inferred from known classes that seem to be similar. Naturally, either of these heuristics can misfire – stale bread and wood-textured plastic are examples of materials whose haptic properties may not agree with a visual impression – but humans are generally quite good at picking up on patterns of this type.

In this thesis we are interested specifically in patterns that can be identified in visual input and give a hint of haptic properties. This cross-modal inference is both useful and common among humans. Many of the examples above involve *anticipation* of haptic percepts. After the first few times, you don't need to make contact with a patch of ice in order to decide to walk, drive, or ski around it.

1.1. Motivation

Many robotics applications can benefit from haptic sensing, and in particular relating haptic sensing to other sensing modalities. Typical mobile robots attempt to navigate primarily based on visual feature tracking, but this strategy neglects many dimensions of sensory experience that we use to understand our environment. When walking and manipulating objects, humans (and other animals) exert forces that are finely calibrated to the properties of what they are interacting with. Many of those properties are haptic in nature, such as weight, hardness, roughness, slipperiness, and temperature. Let us consider several examples.

A self-driving car needs to cope with a variety of road conditions, which vary with weather, road type, local highway maintenance status, and other factors. The appearance of the road changes drastically over the course of a day. However, it's the haptic properties that are important for a safety-conscious driver: besides gross geometric features like potholes, roughness indicates an unpaved road section which deserves care, or rumble strips warning of highway boundaries; yet a patch of ice or a thin layer of water is to be avoided for its slipperiness. In a different scenario, a kitchen assistant robot needs to deal with objects sporting a wide variety of surface materials, from a knife (rubber handle, strong power grip) to an ice cube (very slippery, strong precision grip) to a raw egg (same size as the ice cube but a different material, gentle precision grip required).

Crucially, the common feature between these different situations is that the robotic agent must adapt to the haptic properties of surfaces, and it's helpful if that adaptation can begin by anticipating the properties before coming into contact with the surface [1]. Humans make these judgments and calibrations mostly subconsciously, relying on lifelong experience to build up a virtual database of material categories and their haptic properties. But even beyond the "database lookup", faced with a novel surface we can make reasonable predictions by comparing it visually to known materials. This cross-modal inference is a little-studied skill that would be quite useful for robots interacting in the real world.

The physical world is full of a vast menagerie of materials of all shapes, sizes and textures. As part of perceiving these complex scenes, robotics researchers should consider the importance of haptic properties. Often it's advantageous to try to infer such properties from vision, for several reasons. First, many more robots have cameras than have haptic sensing, so with an algorithm for visual inference such robots could still get some haptic information. Furthermore, even in the case of robots that do have haptic sensors, there are benefits to making the prediction before coming into physical contact with an object. For example, the original Roomba vacuuming robot maps its surroundings by bumping into things, but more advanced mobile robots have non-contact sensors (lidar or cameras) that improve reliability and make for less risk of knocking things over.

1.2. Contributions

The main contributions of this thesis are the design of a device and operating software for recording matched visuo-haptic information about surfaces in the field, a substantial database of such recordings, and analysis of what can be automatically learned from the data. We present the initial large-scale dataset gathered with this device along with our experiences at each stage of the project, which should be useful for future researchers looking to extend the dataset or collect similar data. The hardware is discussed in Chapter 2 and software in Chapter 3. To demonstrate effectiveness, we studied the performance of existing classification algorithms on Proton Pack data, covered in Chapter 4. After that, we turned our focus to collecting a large-scale dataset of surface interactions, described in Chapter 5. Chapter 6 discusses our initial efforts to apply modern machine learning techniques to our dataset to fulfill the goal of inferring surface properties from visual input. Finally, Chapter 7 presents conclusions and directions for future research.

Here is an overview of the contributions, chapter by chapter:

- Chapter 2: Proton Pack Hardware

  - We chose to design and build the Proton as a portable, self-contained, human-operated sensing system. A literature review situates the Proton in the context of other custom haptic sensing devices.

  - Each of the sensors is described, as well as the supporting hardware that we use to interface with the Proton.

  - The end-effectors, while interchangeable, are always carried on the Proton itself in a "parking lot" fixture which allows the onboard computer to detect which end-effector is active.

- Chapter 3: Software

  - The Proton is run by a custom supervisor program written in Rust, which communicates with each sensor and records data, interfacing with the human operator via a self-hosted web interface.

  - A "flow" system allows orchestration of several sensors and operator inputs in sequence, for constructing experimental setups in a simple configuration language.

- Chapter 4: Surface Discrimination with Engineered Features

  - An initial proof-of-concept experiment (published in the proceedings of MFI 2016 [2]) classified five surfaces using quantitative features calculated from the vibration data with the tooling ball end-effector. The experiment was successful but revealed several problems with the Proton design.

  - A second study (published in the proceedings of ICRA 2017 [3]) classified five different surfaces with higher accuracy after improving the hardware design by moving the accelerometers closer to the contact point. This study also verified a new egomotion tracking algorithm using the onboard camera, avoiding the dependency on a grounded motion tracking system.

  - We compared the effectiveness of classification with different types of features calculated from the vibration data: the "scan-dependent features" used in the previous two studies, and the "scan-free features" developed by Strese et al. [4, 5]. This work was published in the proceedings of the 2017 IEEE World Haptics Conference [6].

- Chapter 5: Dataset

  - A preliminary study (presented at the 2017 AAAI Spring Symposium on Interactive Multi-Sensory Object Perception for Embodied Agents [7]) with one surface found that one minute of data collection was sufficient to pin down estimation of the friction coefficient.

  - We present a new database of matched visual and haptic data from recordings of three end-effectors interacting with 357 surfaces. We describe the format of the data, which will be publicly released, and present sample code to parse it.

- Chapter 6: Cross-modal Machine Learning

  - A human-subject study (presented as a work-in-progress paper at Haptics Symposium 2018) conducted online using Amazon Mechanical Turk investigated the extent to which human participants agreed with the Proton operator on numerical ratings of basic haptic properties (hardness, roughness, stickiness, and warmness) when examining close-up surface images. Correlation between operator and participant ratings was found to be relatively low, indicating that the vision-based inference task is difficult even for humans.

  - We approached the same task using machine learning: given images as input, infer the numerical ratings of haptic adjectives. Performance was mixed but consistent with the human-subject study, perhaps indicating that the task is not feasible as defined.

  - We further trained machine learning regression models to infer the values of the quantitative vibration features from visual images.

- Chapter 7: Conclusion

  - We detail suggestions for future work in this area, including ideas for researchers designing their own custom haptic sensing systems, collecting multimodal datasets, and building machine learning pipelines.

- Appendix A: Calibration

  - Extrinsic calibration: we present a procedure that uses a grounded Vicon motion tracking system to infer the relationship between various coordinate frames attached to the Proton from data recorded while rotating the apparatus around a fixed end-effector.

  - Gravity compensation: a second calibration routine, using data from the Proton being moved in free space without touching anything, analyzes Vicon motion track and ATI Mini40 six-axis force/torque sensor output to discover the constant-magnitude offset in the latter due to the end-effector mass.

- Appendix B: Surface List

  - All surfaces contained in our publicly released dataset are listed, with the available sensor streams tabulated.

CHAPTER 2 : PROTON PACK HARDWARE

We designed custom hardware and software for the Proton Pack so that we could tailor it to the kind of data we intended to collect while improving upon prior designs in specific ways.

The design most closely resembles a larger version of Culbertson et al.'s Haptic Camera [8], though with three specific differences: the attached cameras allow for recording synchronized video, the entire device is portable, and in some configurations there are haptic sensors at the contact point. Portability is achieved by carrying all supporting hardware in a backpack and tracking motion via the onboard camera instead of a stationary system such as Vicon or magnetic tracking.

The following sections first describe the handheld sensing rig, in the context of other haptic sensing devices, followed by the supporting hardware contained in the backpack. A description of the software may be found in the next chapter.

Over the course of development of the Proton, design iterations caused several changes in the hardware configuration. These changes and their motivations are discussed as well, but the most weight is given to the final design.

2.1. Background

The field of mechanical haptic sensing remains rather primitive compared to the high-resolution, distributed sensing organ that we call skin. Some researchers are working on skin-type sensors for robots [9] (see [10] for discussion and review), but their focus is often on discrete sensorized end-effectors. Common sensing technologies used as proxies for replicating the human sense of touch include acceleration, force/torque, pressure and temperature sensors. In this section I will give a brief overview of several underlying technologies.

An accelerometer measures the acceleration of itself and any object to which it is rigidly attached. One common application is in measuring an object's quasistatic acceleration due to gravity, which provides information about orientation. No doubt due in part to the proliferation of smartphones and other devices with inertial measurement units (IMUs), small solid-state accelerometers are available with a range of bandwidths. We choose a sensor with multiple axes and a high bandwidth so that we can filter out the constant offset due to gravity and capture the high-frequency vibrations caused by different materials in contact with each other. Notably, Strese et al. have developed a series of "Texplorer" tactile sensing devices centered around accelerometers [11, 12].

The Texplorer devices also use force-sensing resistors (FSRs) to measure the forces exerted on a surface. These are uniaxial devices that measure applied force by changing their resistance. In our apparatus, we use a multi-axis sensor based around a similar technology. A force/torque sensor such as the ATI Mini40 six-axis force/torque sensor uses pairs of *strain gauges* (sometimes spelled *gages*) as the basic sensing unit. A strain gauge is a uniaxial elastic deformation measurement device. Essentially, it is a variable resistor that changes resistance at a rate directly proportional to change in length (see Fig. 2.1 on the following page for an illustration of the operating principle). With a rosette of strain gauges, multi-axial strain can be measured. Finally, six strain gauges allow for the calculation of torque as well. Force/torque sensors therefore consist of a stiff material with known stress-strain relationship equipped with these strain-gauges. Force and torque are calculated by taking the difference between the readings of each gauge and transforming through a known calibration matrix.

There is another way to measure force and torque, without using strain gauges. The key idea, tracking the deflection of a material whose stiffness is known, remains crucial. Instead of attaching sensors directly to this material, we illuminate it and train a camera or light sensor on it. Examples of this type of sensor include the OptoForce, the TacTip end-effector [14], and the GelSight sensor [15]. All consist of a rubber or gel material lit from within. The

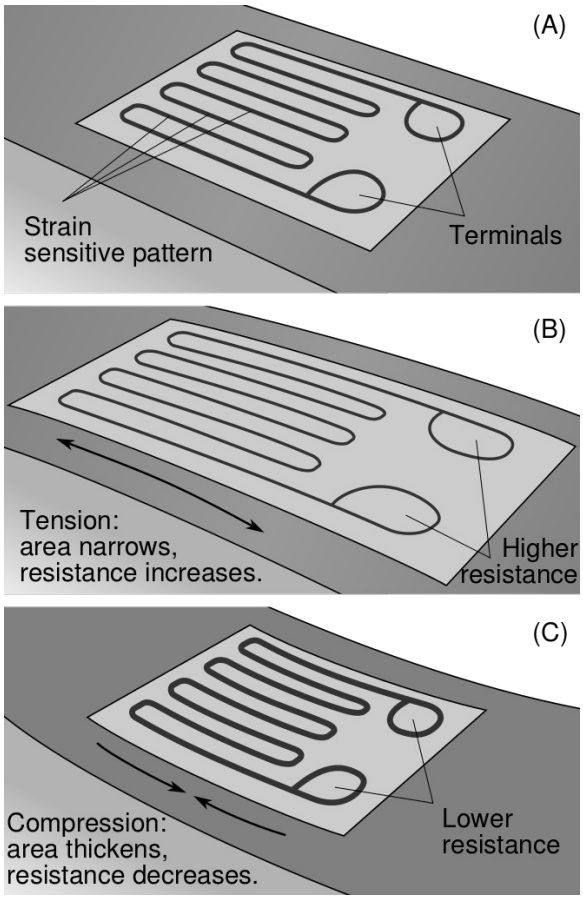


Figure 2.1: Strain gauge schematic. Image from [13].

OptoForce bounces light from a laser diode off of the reflective inner surface and measures the light intensity reflected onto planar-distributed sensors. The TacTip is similar in structure, but the inside of the rubber dome features an array of pins which deform when the dome contacts a surface, and an internal camera observes the geometrical pattern to learn about the surface shape. In the case of the GelSight sensor, a block of gel with a retroreflective surface is illuminated by different colors, and a camera captures the change in appearance when the gel is deformed by an object or surface.

Lastly, the most complex haptic sensor featured on the Proton is the SynTouch BioTac, which does not directly measure force or vibration at all. Instead, it tries to mimic the human fingertip morphologically, having a flexible textured membrane filled with incompressible conductive fluid (see Fig. 2.2 on the next page). When touching an object, the membrane

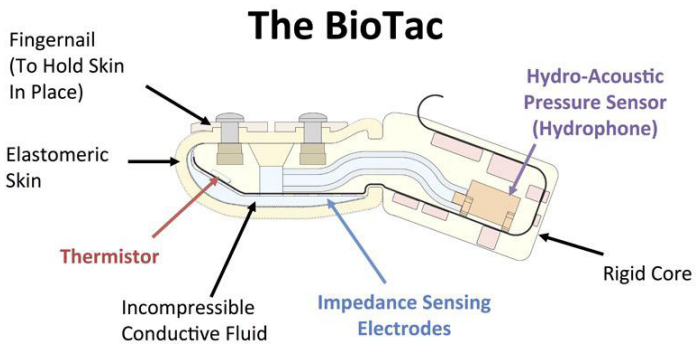


Figure 2.2: BioTac sensor internals. Image from [16].

deforms and pushes the fluid around, changing the conductance seen by an array of electrodes on the underside of the fingertip. The sensor also measures global pressure and temperature of the fluid. By analyzing this multimodal data, detailed information about the surface can be extracted.

2.1.1. Related Work

A 1986 paper by Stansfield [17] was one of the first to try to adapt the principles inherent to the human tactile system to create a robotic haptic perception system. While researchers had previously thought of touch in terms of passive sensing, Lederman and Klatzky had just discovered that humans experience and identify surfaces via "exploratory procedures" [18], specific movements that elucidate the desired haptic property of the object. For example, pressing into an object reveals its softness, while moving one's finger along a surface shows its roughness and stickiness. Stansfield programmed a PUMA 560 robot arm equipped with a 160-element tactile array and a force/torque sensor to physically interact in specific ways with objects. Following such an approach requires one to design the movements, control the robot to execute the desired interaction, and interpret the resulting haptic data.

Following in the vein of Stansfield's early work, several haptic sensing systems have recently been developed for material or texture classification by autonomous robots. These systems use many kinds of haptic sensors, such as an ATI Nano17 force/torque sensor [19], high-bandwidth accelerometers [19, 20], and custom artificial fingers that include strain gauges

or polyvinyledine fluoride (PVDF) sensing elements embedded in a soft material [21, 22]. Other researchers employ the commercially available SynTouch BioTac biomimetic tactile sensor [23, 24], described above. As established by Lederman, Klatzky, and colleagues [25, 26], changing the speed or force of the contact interaction typically alters the amplitude and frequency of the resulting haptic signals, although the surface continues to feel the same to the person executing the movement. Consequently, these haptic sensing projects all carefully controlled the physical interactions between the sensing system and the surface in question, specifically using a custom test rig [23], a robotic arm [22], or an upper-torso humanoid [20, 19].

More recent work in this domain focuses on active perception. For example, Xu et al. [27] used a robot-mounted BioTac to collect haptic surface data via exploratory movements such as pressure, lateral sliding, and static contact. When attempting to identify an object, the robot selects the motion that will provide the most disambiguation between likely object candidates. Similarly, Lepora et al. [28] compared the performance of passive and active perception under uncertainty, showing that active touch can somewhat compensate for noisy sensors. Having a robot wield the sensing system using a set of pre-programmed motions typically yields consistent contact force and scanning movements and therefore repeatable haptic signals. However, significant effort must be expended to program the interaction controller to function well for a wide variety of surface properties, which are unknown at the start. One must also ensure that the motion of the robot itself does not generate significant tactile vibrations, as they may obscure the sensations caused by contact [29]. These robotic touch systems must typically be closely supervised by a human experimenter and are not easily portable.

The other major approach to capturing haptic surface data is to use a handheld sensing system wielded by a human experimenter. For example, Pai and Rizun created the WHaT, a wireless haptic texture sensor that includes a pair of miniature two-axis accelerometers mounted at right angles and a miniature piezoresistive force sensor integrated in a pen-

shaped shell [30]. This design allows measurement of natural surface interactions through a probe at the WHaT's tip. Being wireless allows this sensing system to have a large workspace; however, one key disadvantage of this design is that the sensor cannot track its global position or orientation. To solve this problem, Andrews et al. [31] added the ARTag optical motion tracking system [32] to a WHaT probe, and they used a linear Kalman filter to compensate for noise affecting the motion tracking measurements. This system was used to measure both stochastic and patterned textures and to estimate surface compliance [31].

Another haptic sensing device is the ThimbleSense [33], which is worn on a human or robotic fingertip. It includes an ATI Nano17 six-axis force/torque sensor within a thin shell, which provides enough information to calculate the contact point. An external Vicon system is used to track the position of each fingertip, although interactions with grasped objects can occlude these markers. Another force-sensor-based haptic capture system is that of Höver et al. [34]; a human operator can intuitively explore real objects using a PHANToM Desktop haptic interface that is equipped with a custom probing tool and a Nano17. The captured force data can be interpolated and used to model arbitrary non-linear materials with visco-elastic behavior [34]. Finally, Culbertson and colleagues created a haptic recording device that has some similarities to the one presented in this chapter. Their device is small and pen-shaped, containing a six-axis force/torque sensor, high-bandwidth accelerometers, and an external magnetic tracking system for position measurement [8]. Because the magnetic system is not portable, textures to be recorded must be brought to the sensing system, and the operator cannot record surfaces in situ. Handheld haptic recording systems show great promise for efficient collection of naturalistic surface interaction data, but existing haptic datasets such as the Penn Haptic Texture Toolkit [35] tend to contain very limited visual information about the included surfaces.

2.2. Architectural Considerations

Before discussing the handheld part of the Proton, some discussion must be given to the design decision to use a handheld sensing rig instead of a mobile manipulator. As discussed in

	Characteristic	Human	Robot
1	Repeatable path	no	yes
2	Controllable force and speed	somewhat	yes
3	Adapt interaction to surface	yes	maybe
4	Avoids parasitic vibrations	yes	no
5	Setup difficulty	easy	hard
6	Flexibility of procedure	low	high
7	Robustness	high	low
8	Cost	low	high

Table 2.1: Pros and cons of using a human or robotic approach to operating the Proton.

the previous section, both solutions are possible and we determined that handheld operation would have more advantages for our intended application.

Both human-operated and robot-mounted tactile sensing strategies have been tried before. A robot will be able to follow a scripted interaction every time, with a selectable range of applied forces and movement speeds, but it will have trouble tailoring the interaction to a specific surface (for instance, satisfying force or positioning requirements while in contact with a surface of unknown properties) unless extensive relative processing is done. A human operator, on the other hand, will generally make non-repeatable motions but is much more able to adapt motions to a particular surface. The human can make these adjustments to avoid the end-effector damaging the surface (or vice versa), as well to gather more informative data. To extract more information from the interaction, exploratory procedures can be adjusted according to estimated properties of the material. For example, Van Hoof et al. [36] present an information gain framework for doing this adjustment with a robotic platform, but a human operator provides a shortcut. Importantly, robots typically have characteristic vibrations caused by their actuators (though the severity of this issue can be traded for cost), while humans do not owing to a completely different mechanism of automation. These pros and cons are summarized in Table 2.1.

Of these, row 1 and row 2 seem to favor a robotic approach, while row 3 and row 4 are current technological limitations that are likely to be overcome in the future. However, further design considerations serve to shift the balance. Since we are trying to extract the maximum possible amount of information from vibrations generated during surface interactions, row 4

is of prime importance. Furthermore, the disadvantage implied by row 2 can be mitigated by *monitoring* the force and speed (referred to as "scan-time parameters") applied by the operator, and collecting sufficient amounts of data to ensure a good distribution. Lastly, nonrepeatability, though initially a seeming problem, can be spun as a desirable human touch since we want to focus the data collection on the areas of scan time parameter-space that will give rise to the most information, and humans have pre-existing intuition about where that is. For the type of flexible data collection that we envisioned, consideration of the cost and complexity of setting up a robotic system tipped the balance in favor of r a human operator.

Considering the overriding concern of avoiding parasitic vibrations, and having mitigated or disregarded all of the concerns, we designed the Proton around a human operator. A rigid assembly holding all the sensors in a constant configuration relative to each other is to be carried by hand and used as a tool for touching surfaces in various ways, while supporting hardware (i.e., battery, power distribution, data recording and storage) is located in the backpack. No physical display or control interface is built in – instead, configuration and control are done over the Internet via a wireless network hosted by the backpack computer.

2.3. Handheld Rig

The handheld rig, shown in Fig. 2.3 on the next page, is designed around the operator's hands and contrasting goals for the visual and haptic modalities: the haptic sensors must obviously touch the surface under study, while the cameras should be a significant distance away in order to get a wide field-of-view. In particular, to allow for the use of structured-light depth cameras such as the Structure Sensor [37], the camera mount is placed 40 cm above the end-effector/surface contact point.

The base of the handheld rig is cylindrical and encloses an IMU. Above that, parallel plates made of medium-density fiberboard (MDF) have the camera mount at the top and a handle

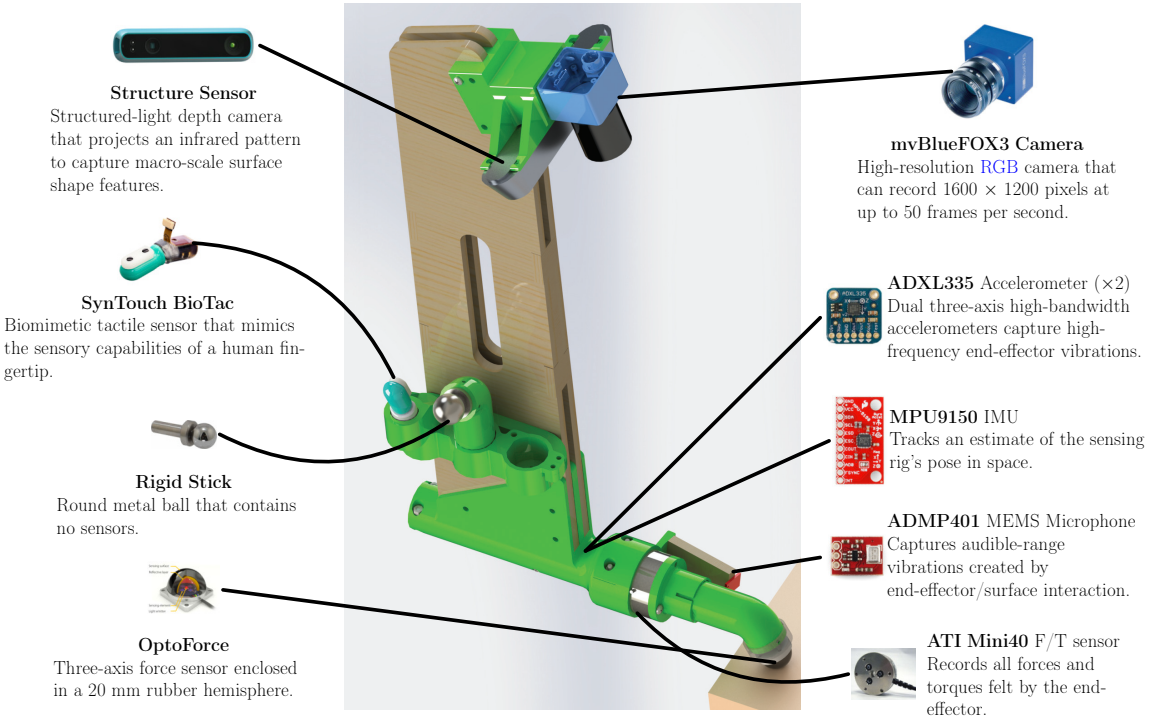


Figure 2.3: Sensors on the handheld rig.

in the middle. As shown in Fig. 2.4 on the following page, the operator typically holds the Proton with one hand on the handle and the other supporting the cylindrical base.

The cylindrical base is 3D-printed. Its two halves, which come apart to allow access to the IMU mounted securely inside, are held together by a collar in the front. This collar holds the Mini40, to which the end-effectors are attached using a quick-release clamp for easy insertion and removal.

Three finger-shaped end-effectors may be clamped into the handle, differing only in the sensing hardware at the tip. Their 3D-printed shapes were dictated by the BioTac sensor, which is intended to be held in a horizontal position, and a desire to have the point of contact of all end-effectors at the same position relative to the rest of the rig. This design removes a potential source of variation and simplifies the interpretation of the Mini40 readings.

Just below the handle is attached the "parking lot". This innovative 3D-printed part allows the operator to safely store the inactive end-effectors during field operation, enhancing

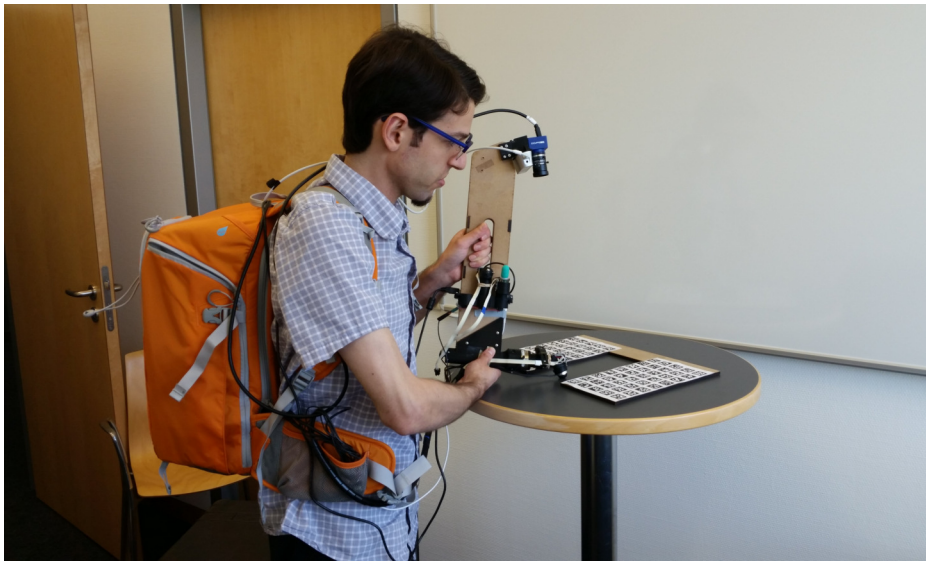


Figure 2.4: An experimenter holds the Proton, equipped with the OptoForce end-effector, in contact with a desk surface.

the Proton's portability. The parking lot contains three keyed spaces where one of the end-effectors can be placed when not mounted at the base. Internal limit switches detect the presence of each end-effector, so the backpack computer can tell which end-effector is in use and record data from the appropriate sensors.

All of the 3D-printed parts (shown in green in Fig. 2.3 on page 15) were manufactured on a Fortus 450mc FDM printer, using Nylon 12 with a 0.005 in layer resolution and solid fill.

2.3.1. Sensors

The Proton has two types of sensors: interchangeable and permanent. The three interchangeable end-effectors contain various haptic sensors. However, regardless of end-effector choice several sensors are permanently mounted in fixed positions on the handheld rig. These are the visual sensors, high-sensitivity force/torque sensor, and IMU. (High-bandwidth accelerometers are always present, but they are mounted directly on the end-effectors so as to be as close to the surface interaction as possible.)

The permanent sensors of the Proton fall into three main categories: visual, haptic, and motion tracking. Originally, for calibration purposes and ease of prototyping, we used an external motion tracking system. While such systems allow free movement within a given three-dimensional space (several cubic meters), which would make the Proton more portable than a desk-mounted system, this approach does not fulfill the design goal of a portable data collection device. Therefore, after finishing an initial proof of concept experiment, we performed additional calibration in order to perform ungrounded motion tracking using the onboard camera and fiducial markers placed in its field of view (see Sec. 2.3.1 on page 19).

Cameras

Providing the Proton with the means to replicate high-acuity vision was an important design goal. Humans can use 2D and 3D cues to infer surface properties before touching them, so it's an important sensory modality to include in our database. To this end, the Proton features two cameras: a depth camera and a red-green-blue (RGB) camera.

The Structure Sensor is a compact structured-light depth camera. It uses an infrared projector to overlay a pattern on the scene in its field of view, and an adjacent infrared camera to detect how that pattern plays over objects at different distances. The technology is similar to other depth cameras such as the Microsoft Kinect. Designed to be mounted on a mobile device with its own RGB camera, the Structure does not include its own visible-light camera.

The BlueFox is a high-end RGB camera. We use a mvBlueFOX3-1020C-2111 camera with a Computar M2518-MPW2 or M1224-MPW2 lens. The latter lens has a wider field of view (30.4° as opposed to 15°), which helps with motion tracking (see Sec. 2.3.1 on page 19). The camera records 1600×1200 pixel frames at up to 50 Hz.

When we record a surface using both of these cameras, the Proton acts as a red-green-blue-depth (RGBD) camera.

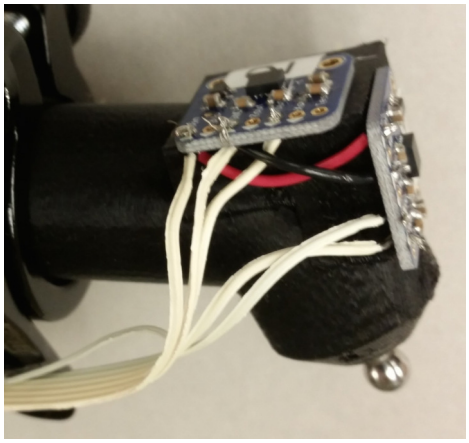


Figure 2.5: Final end-effector design with high-bandwidth accelerometers mounted orthogonally and very close to the surface contact point.

Vibration

When interpreting the haptic properties of a surface by touching it with a tool, vibration has a strong effect on how humans perceive the surface properties [38]. Tapping and dragging a tool on a surface produces characteristic vibrations at various frequencies. To capture this information, the Proton uses two high-bandwidth accelerometers (ADXL335 in the final version, though earlier we used the ADXL326) and a MEMS microphone (ADMP401). The positioning of these sensors is shown in Fig. 2.3. At first, we mounted the high-bandwidth accelerometers inside the Proton's base to make them independent of the end-effector. However, pilot experiments confirmed that putting them closer to the end-effector resulted in capturing more vibration data, so in the final version each end-effector has its own set of accelerometers (shown in Fig. 2.5). When mounting a given end-effector into the quick-release clamp, its accelerometers are connected via a six-pin connector (see Sec. 2.4.2 on page 24). The sensors themselves measure vibration on all three axes, but the Z-axis bandwidth is lower than the other two axes, so we don't want to use it. Instead, we take readings from the X and Y axes of one accelerometer, then mount another accelerometer with its X or Y axis parallel to the Z axis of the first, and take the third reading from that parallel axis.

Scan-time Parameters

With the sensors described in the previous section, we can measure with high fidelity the vibrations caused by surface interaction. However, several factors besides the character of the end-effector and surface influence these vibrations. These are characteristics of the interaction itself, which we broadly term "scan-time parameters". The most prominent parameters are normal force and scanning speed. Since the Proton, having a human operator, does not dictate these parameters exactly, we measure them.

Force To measure contact force, we use a Mini40 at the base of the Proton. (Note: the OptoForce and BioTac end-effectors can measure force in some way, as described in Sec. 2.3.2 on page 21, but we want to uniformly monitor the contact force independently of which end-effector is active.) The sensor is mounted in between the end-effector mount point and the rest of the Proton handle, as shown near the bottom of Fig. 2.3. This placement ensures that all mechanical coupling of the end-effector to the Proton (excluding flexible cables) goes through the Mini40, allowing it to capture all forces and torques. Cables are taped down to minimize their contribution to any measurements.

The force and torque readings are obtained in the sensor's local frame, which is rigidly attached to the handheld rig. We would like to decouple the tangential and normal forces to the surface, which requires transforming force vectors from the local to global frame. For this process we need to know the real-time orientation of the rig.

Motion At first, we used a straightforward yet undesirable method to track the Proton's motion through space: we attached infrared-reflective markers and used a commercially-available Vicon motion tracking system. This system allowed us to track the rig anywhere within view of the ceiling-mounted Vicon cameras, which is a three-dimensional volume at least several cubic meters in size. However, it ties data collection to the fixed motion tracking system, so we later developed an alternative approach.

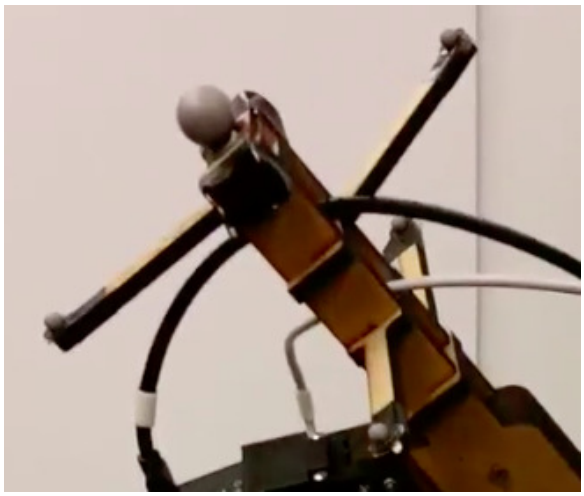


Figure 2.6: "Tree" of infrared reflective markers that makes the handheld rig visible to the Vicon motion tracking system.

To set up Vicon tracking, we attach a fixture containing small spherical infrared-reflective markers to the top of the handheld rig. Five markers in a non-coplanar, asymmetric configuration (shown in Fig. 2.6) is enough to guarantee that when the Vicon system locks on, the tracking is always pegged to the same orientation. The coordinate system attached to this fixture is drawn in the Vicon software by hand, so it is not exactly aligned with any of the other local frames. This problem is resolved by a custom calibration procedure described in Appendix A.1 on page 159.

(a) Tooling ball (b) OptoForce sensor (c) BioTac sensor

Figure 2.7: Proton end-effectors

2.3.2. End-effectors

The three end-effectors which we have used for our data collection with the Proton were chosen to represent those that might be present on a wide variety of hypothetical haptic-sensing robots. They vary in sensing modalities at the contact point, as well as hardness and friction characteristics.

Specifically, the three end-effectors end in a tooling ball, OptoForce sensor, and BioTac sensor respectively. These are shown in Fig. 2.7 and will be described more fully in the following sections.

The end-effectors were redesigned after our first proof-of-concept experiment. The improvements were validated in a second proof-of-concept [3].

Accelerometer Improvements

While the original design of the Proton called for high-bandwidth accelerometers in the middle of the handle, we found that the vibration signal was muffled (see Fig. 4.8 on page 62). Accordingly, the end-effectors were redesigned to mount the accelerometers as close to the surface contact point as possible. This is the configuration shown in Fig. 2.7.

At this time we made two other important improvements in the accelerometer setup.

First, we switched from ADXL326 to ADXL335 chips. Both are low-power MEMS accelerometers with analog outputs and X/Y axis bandwidths of $0.5 - 1600\,\text{Hz}$, the only difference being output range: $\pm 16\,\text{g}$ for the former and $\pm 3\,\text{g}$ for the latter. Since we are measuring relatively low-magnitude vibrations, the narrower output range means less quantization error when digitizing the readings.

Second, we use both the ADXL326 and ADXL335 by way of Adafruit breakout boards, which have built-in filter capacitors. The capacitors are $0.1\,\mu\text{F}$ which filters the output to a bandwidth of $50\,\text{Hz}$. This is good for typical IMU applications which need to filter out vibration noise for stable motion tracking, but it attenuates the high-frequency vibrations where we expect valuable information about the surface to be present [19, 4]. Therefore, we replaced the filter capacitors with smaller $4.7\,\text{nF}$ components, increasing the bandwidth to $1064\,\text{Hz}$, which is comparable to the $1000\,\text{Hz}$ bandwidth of the accelerometers used by Culbertson et al. [8, 39] as well as Romano and Kuchenbecker [19].

Tooling Ball

The simplest end-effector sports a steel tooling ball. Our custom 3D-printed end-effector housing is shown with the tooling ball in Fig. 2.7a. Other than the high-bandwidth accelerometers, there are no haptic sensors on the end-effector. However, it is hard and rigid, meaning all forces and vibrations are transferred to the Mini40 with a minimal amount of damping.

This end-effector is meant to represent robots that have no innate haptic sensing capabilities, but may have other common sensors (such as accelerometers) that can be repurposed as such. At $20-$21 depending on diameter,[1] the tooling ball is an inexpensive end-effector.

Tooling Ball Diameter Several tooling balls of different diameters were used in different Proton design iterations, including $0.75\,\text{in}$, $0.375\,\text{in}$, and $0.125\,\text{in}$, before settling on the

[1] Our tooling balls were sourced from McMaster-Carr: https://www.mcmaster.com/#8481a33/=16658kt

0.375 in ball. Originally, the larger diameter was selected to make the end-effector relatively heavy, guaranteeing high amplitude of forces and vibrations during surface interactions. Later, that concern about low measurements turned out to be overblown, and we switched to smaller tooling balls in order to resolve smaller surface contours. This redesign is covered in Sec. 4.5.2 on page 63.

OptoForce

As introduced above, the OptoForce is a hemispherical rubber sensor that measures force by recording the hemisphere's deformation. We designed the end-effector housing (see Fig. 2.7b) such that the surface contact point of the tooling ball and the OptoForce sensor would be similarly positioned with respect to the body of the Proton. Force readings in all three axes are generated, but only the Z axis is calibrated.

At €750-900, this sensor is mid-range in terms of price and quality.

BioTac

As discussed above, the BioTac measures pressure, shape, pressure, and temperature in a fingertip-shaped package. Our BioTac mounted in its end-effector is shown in Fig. 2.7c. Similarly to the OptoForce mounting, this end-effector is designed to position and orient the sensor so that the surface contact point is close to where the tooling ball would be, from the perspective of the Proton base.

This end-effector is the most costly (approximately $12,000 per fingertip) and high fidelity of the three, though its readings are not directly relatable to force or other commonly measured haptic quantities.

2.4. Backpack

Many sensing systems require extensive supporting hardware. It's not uncommon to see a small end-effector trailing a bundle of wires that may go to a desktop computer, bench

power supply, function generator, and more. For the Proton, we wanted to eliminate all factors that would tie the system to a lab bench, which meant finding a way for the operator to carry a power supply, a computer to interface with hardware and record data, and any additional hardware specific to the individual sensors (most notably, a signal conditioner for the Mini40 and a daughterboard to interface with the analog sensors).

To meet these needs, we designed a backpack-enclosed support system including power distribution circuitry which runs from AC or battery power, a miniature computer, as well as the aforementioned signal conditioner and daughterboard.

2.4.1. Power Distribution

A high-level schematic of the hardware carried by the operator is shown in Fig. 2.8 on the next page. Double-lined boxes denote components on the handheld rig, while the rest is contained in the backpack.

As seen in the upper right-hand corner of the figure, the entire system is powered from a 12 V battery. This may be replaced with an AC adapter producing $12 - 19$ V when the Proton is used in a stationary configuration. This main power source is connected directly to the computer and, through a voltage regulator, to the Mini40 signal conditioner, Teensy daughterboard, and a USB hub. As described in the following section, this daughterboard handles the analog sensors and all data is eventually received by the computer through the USB hub.

2.4.2. Smart Task Board

Some sensors natively communicate over USB (namely the cameras, OptoForce and BioTac), but the rest require a lower-level interface. The Mini40, high-bandwidth accelerometers and microphone output analog voltages that need to be converted to digital values. The IMU consists of digital sensors, and outputs readings using the inter-integrated circuit (I2C) protocol.

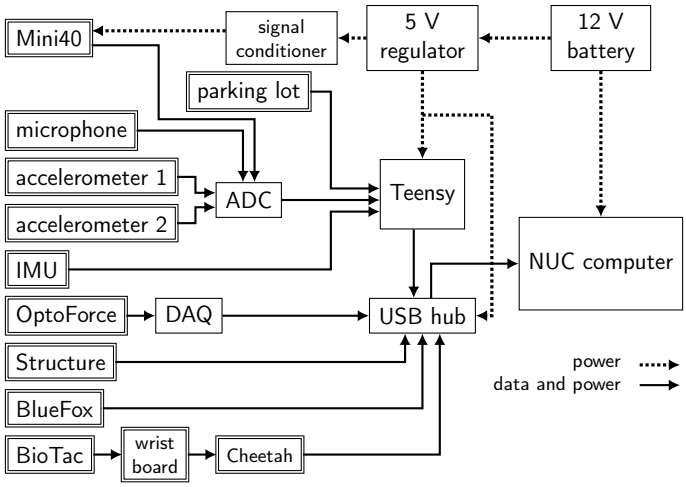


Figure 2.8: Wiring diagram of rig and backpack. Reprinted from [2] © 2016 IEEE.

To manage this assemblage of sensors, we use a modified version of the "Smart Task Board" developed by Brown et al. for an earlier project [40]. Diagrammed in Figs. 2.9 through 2.13 on pages 26–28, it features a differential amplifier necessary to condition the Mini40 readings, analog-to-digital converters (ADCs) to digitize the Mini40, accelerometer and microphone readings, and a Teensy 3.1 microprocessor to tie it all together. The Teensy's firmware is described in Sec. 3.1.2.

2.4.3. NUC Computer

We chose an Intel NUC5i3RYK as the nerve center of the Proton. This computer is quite compact and comes with a Core i3 CPU, 120 GB disk space plus 4 GB RAM, and connectivity including HDMI output, internal WiFi, Ethernet, and four USB3.0 ports. All storage is solid-state, meaning there are minimal moving parts to be damaged by vibration or impact when carrying the Proton. The connectivity allows us to read from all of the sensors in real time as well as wirelessly receive control signals. As described in the next chapter, the Proton is not normally connected to traditional input/output devices such as mouse, keyboard and monitor, but rather hosts a web server and is controlled wirelessly via smartphone.

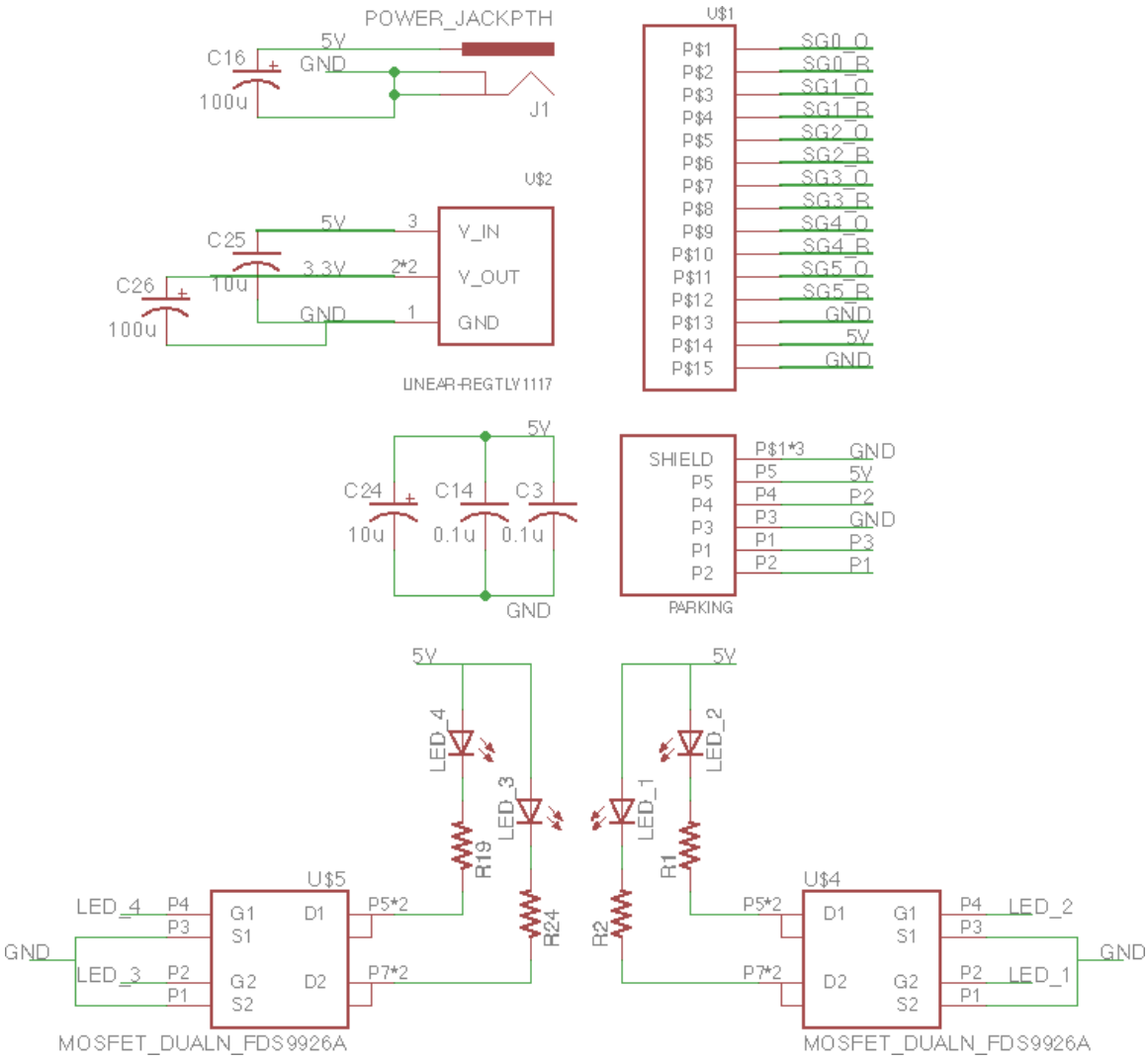


Figure 2.9: Schematic of power distribution circuit. Input power at 5 V is regulated to 3.3 V, smoothed by several filter capacitors, and distributed to various sensors. Four super-bright light-emitting diodes (LEDs) are used as status indicators. The two parallel connectors in the top right interface with the Mini40 and the parking lot (see Fig. 2.3).

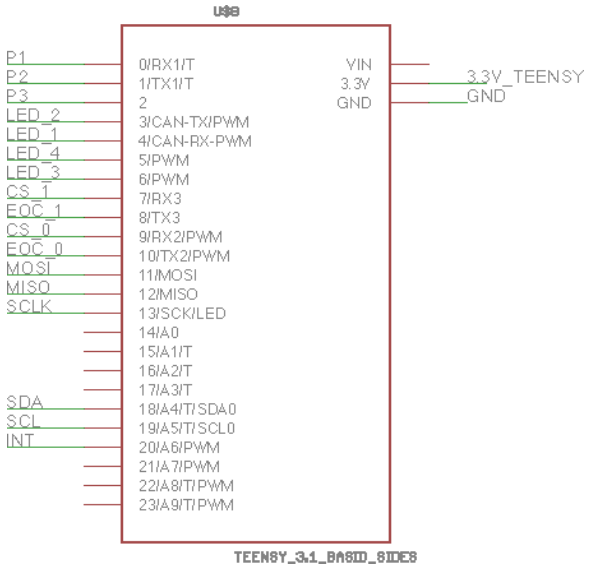


Figure 2.10: Schematic of Teensy daughterboard and peripherals. This diagram shows the power (right side) and the input/output pins (left side) of the Teensy 3.1. The Teensy controls the four status LEDs and communicates with sensors over the serial peripheral interface (SPI) and I2C protocols. While collecting sensor data, it sends data packets to the onboard computer at 3000 Hz.

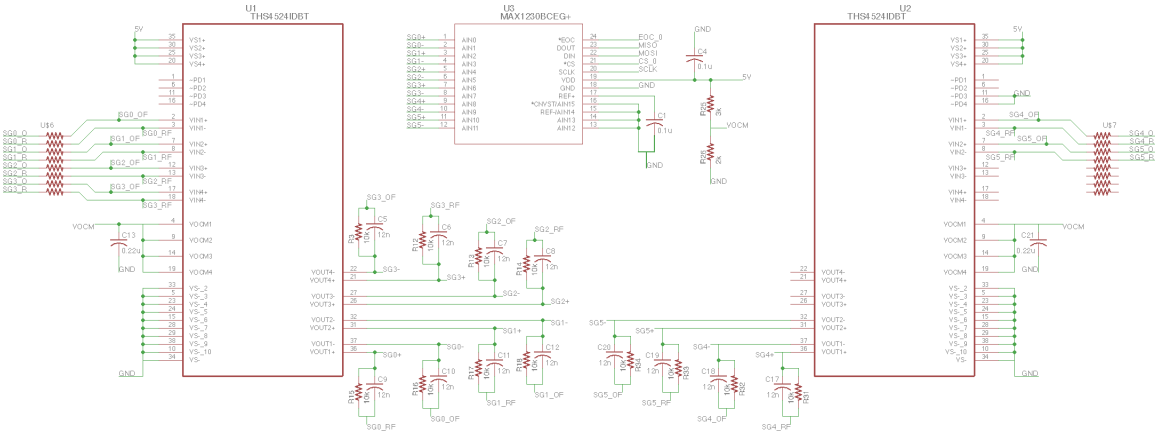


Figure 2.11: Schematic of Mini40 conversion circuit. The sensor output consists of paired electrode readings, which are subtracted and amplified by the THS4524 differential amplifiers. The resulting amplified signals are then converted by the MAX1230 ADC to digital readings, which are communicated to the Teensy over SPI.

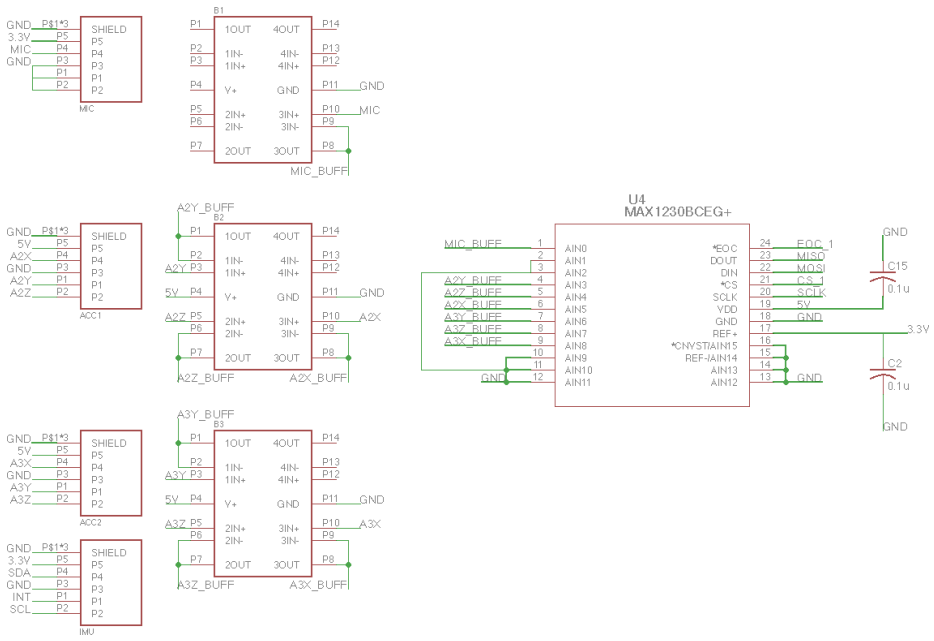


Figure 2.12: Schematic of analog sensor acquisition circuit. This part of the circuit covers the analog accelerometers and the microphone. Another MAX1230 ADC converts measurements from each accelerometer channel and the microphone. Also included in the lower left is the connector to the IMU, which does not need an ADC but sends its accelerometer, gyroscope and magnetometer readings to the Teensy over I2C.

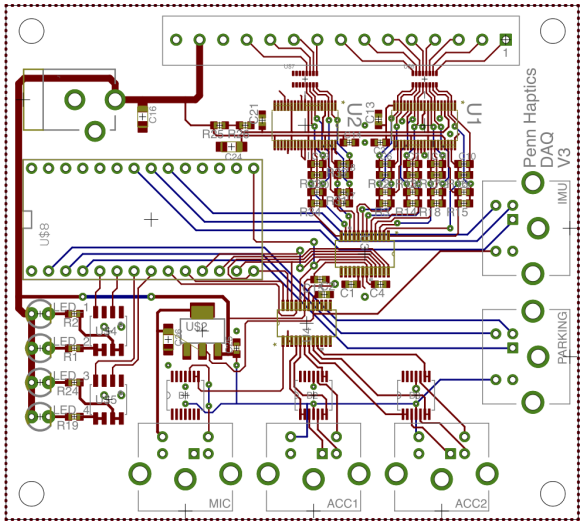


Figure 2.13: Two-layer printed circuit board (PCB) layout. This PCB includes the circuits diagrammed in Figs. 2.9 through 2.12. The Mini40 plugs into the parallel connector at the top, while the other sensors are connected to the ports arrayed around the lower right of the board. Power is supplied through a jack in the upper left. The Teensy, which controls everything, plugs into the pins in the middle left, just above the status LEDs.

2.5. Summary

This chapter presented the design and implementation of the Proton, a versatile visuo-haptic sensing system that will be used to gather a dataset we hope will be broadly useful to robotics researchers and their robotic platforms. The Proton contains sensors that span relevant parts of the haptic and visual perceptual space, designed to gather the types of data encountered by many different robots, which may have haptic sensors, visual sensors (2D or 3D), or both. Some robots may query our database in order to identify an unknown surface. Others may be trying to predict haptic sensations given visual data, or the reverse.

I also laid out the reasoning behind design decisions and tradeoffs we made, such as our choice of a human operator and the selection of visual and haptic sensors. For the latter, we tried to represent the wide range of haptic sensing capabilities that is likely to be represented by real-world robots.

Hardware is only one part of the story. The next chapter will detail the software that underpins our projects with the Proton, including the firmware that interfaces with sensors, operating software running in the backpack, user interface and data processing.

CHAPTER 3 : SOFTWARE

The hardware design detailed in the previous chapter provides the physical capability to collect data, but many layers of software were involved in the project, several of them custom-designed for the Proton Pack. This chapter details the software that runs on the Intel NUC (user interface and data recording), on the Teensy daughterboard (low-level firmware), and on other computers for post-processing and analyzing the collected data.

3.1. Data Collection

The custom supervisor software that runs in real time on the Proton computer, interfacing with the operator and recording data from the sensors, is written in Rust. Rust is a new systems programming language that puts an emphasis on safety (i.e., avoiding crashes, data corruption and undefined behavior) while running quickly and supporting data-race-free concurrency. These properties were important in writing software with many independent modules that record data simultaneously from various sensors. In normal operation the computer is operated in a headless fashion (communication only over wi-fi, without a mouse, keyboard or monitor attached), so the supervisor must not crash.

3.1.1. General Architecture

The supervisor is organized into one manager thread and a number of separate services, which each run on their own thread and communicate via message passing (see Fig. 3.1 on the following page). A service is either a driver (communicating directly with a sensor) or user interface, but the message passing interface is the same. The messages are organized as commands sent *to* services (start, stop, and data) and events bubbling up *from* services (start/stop another service, reboot computer, fatal error report, and data). Both variants

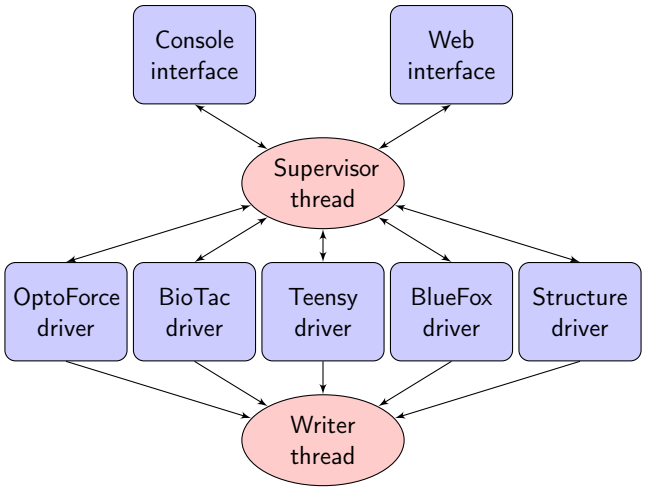


Figure 3.1: Block diagram of supervisor software. Ovals represent the controlling threads (one to supervise the sensors, and one to write data to disk), while rectangles represent the threads that interface with the sensors and the operator.

of the data message are targeted at a particular service and contain some payload that the service understands (e.g., capturing a camera frame or changing the sampling rate of an accelerometer).

The goal of each sensor driver is to operate at the speed of the sensor so as not to miss any data. Accordingly, the drivers send their data to a dedicated "scribe" thread that writes to the hard drive. It is possible to have each sensor thread write data as it becomes ready, but in this situation the threads end up contending with each other and slowing each other down. By sending the data to be written to the writer thread, writes are automatically serialized and the sensor threads do not have to wait for the disk before taking another reading. Every data packet is individually timestamped to ensure accurate temporal analysis.

Each service has a "step" function where it executes a repetitive action or responds to messages, and the manager thread repeatedly invokes this function in one of three *blocking modes*. In *immediate* mode, used for services that are IO-bound or CPU-bound but otherwise should run as fast as possible, the step is called again as soon as it finishes. In *infinite* mode, the step is not called unless there is a message to process. Lastly, a service can specify a desired period (for example, a camera driver should capture frames at a consistent frame

rate). In this case, the manager thread uses a timer to try to match this period by inserting a delay in between steps. Using a simple proportional controller, the delay is adjusted every 100 steps to compensate for the time taken by the step function itself.

3.1.2. Sensor Drivers

This section provides more details about the driver for each of the Proton's sensors. The drivers are launched in separate threads and activated only when a particular sensor is selected for data collection.

Structure

The driver for the depth camera wraps OpenNI 2 [41], which is a general purpose interface to "Natural Interaction" (i.e., Kinect-type) sensors. We read depth frames at 30 Hz at a resolution of 1280×1024 pixels in the "100 μm" format. (It is also possible to read raw infrared frames at 30 Hz, 640×480 pixels, but this capability is needed only for debugging purposes since the Structure sensor itself runs the scan-matching algorithm to calculate the depth image.) Each frame is written to disk in a raw format, while timestamps are recorded in a comma-separated value (CSV) file. On request, the driver encodes single frames to portable network graphics (PNG) and sends them to the web app; these images facilitate semi-real-time display and let the user check that the camera is working.

Bluefox

The architecture of the Bluefox driver is similar to that of the Structure driver, with some extra features. Matrix Vision provides the *mvDeviceManager* library, which is a comprehensive object-oriented C API for accessing and configuring Matrix Vision cameras. Wrapping this vendor-provided driver, we use a pull model for retrieving camera images, requesting a frame and waiting for the low-level driver to provide a buffer filled with captured pixels. This buffer must be released before the thread can request another frame, so it is copied and sent to the writer thread; this architecture keeps the throughput of the camera driver as

high as possible. Depending on processor load and USB traffic, it can sustain about 25-30 frames per second (FPS) at a resolution of 1600×1200, but to reduce interference with other sensors we restrict the frame rate to 15 Hz.

The Bluefox camera is quite sensitive to changes in ambient lighting. It does have settings available for continuous automatic adjustment of white balance, exposure, and gain, but as any computer vision researcher will tell you, these are finicky and even worse than manual adjustment. We do not have the luxury of simply calibrating the camera and leaving the settings alone, since we plan to operate the Proton for data collection in a variety of environments and times of day (the difference in ambient brightness between indoor and outdoor scenes is immense, though humans generally don't notice because our biological auto-exposure system is highly effective). Therefore, we provide an interface to adjust the settings at runtime, and we white balance the camera at the beginning of each recording. Specifically, the process involves setting the white balance mode to "Once" while activating automatic gain and exposure, waiting two seconds, then deactivating all three automatic settings to freeze the fitted values. Whenever the ambient lighting changes, the typically operator adjusts the "average grey" setting as necessary, which provides a goal for the auto-gain and auto-exposure to normalize the image brightness.

OptoForce

Our OptoForce driver uses *liboptoforce* [42], third-party software developed at ETH Zurich, which overcomes several shortcomings of the vendor-supplied driver, to communicate with the sensor over a USB (emulated serial) link. On startup it configures the sensor to run at 1 kHz, and at each step (running at 1 kHz using the aforementioned adaptive controller) it receives a data packet and writes the timestamped packet to a file. No other processing is needed, though the axes must be calibrated for offset and scale.

BioTac

Our BioTac driver is adapted from ROS drivers written for the BOLT project [43, 44]. The sensor communicates with the Proton computer through the Cheetah SPI-to-USB adapter. On startup, the driver configures the Cheetah to read from the single finger. Each step (running at 100 Hz) it receives a packet from the sensor, checks the parity, and sorts the received bytes into the appropriate data channel (absolute temperature, relative temperature, fast pressure, slow pressure, and electrode conductivity), and writes the timestamped data packet to a file. Packets with bad parity are dropped (for an unknown reason, the first two packets are always bad).

Teensy and "Parking Lot"

Several sensors lack a USB interface. In particular, the Mini40 force/torque sensor, high-bandwidth accelerometers, and microphone provide analog interfaces, while the digital IMU communicates over the I^2C protocol. To record data from these sensors, we use a custom daughterboard controlled by a Teensy (similar to the popular Arduino microcontroller series), which communicates with each sensor and sends aggregate data packets to the computer. The board and its firmware were adapted from the Smart Task Board [40].

Teensy firmware The Teensy's firmware is written in C++. The microcontroller itself is mounted on a printed circuit board with connectors for the parking lot, IMU, accelerometers, microphone, and Mini40. The Teensy firmware architecture is single-threaded with interrupts. The main loop receives commands over the serial port from the NUC, which are currently limited to the following list:

1. START: starts the flow of packets to the NUC (optionally specifying a sample rate to override the default)

2. STOP: stops the flow of packets

3. PING: writes a byte to the serial port, to confirm that the Teensy is operating

4. REQUEST_PS: checks the state of the parking lot and writes a byte to the serial port indicating which end-effectors are parked

The Teensy manages concurrent execution with two timers. The IMU timer interrupt is raised every millisecond, at which point the Teensy transfers all available accelerometer and gyroscope readings, plus one magnetometer reading, into a buffer, and raises a flag. Independently, the ADC timer interrupt is raised every 333 microseconds (configurable by the START command). This interrupt handler reads data from the Mini40 and sends a packet to the NUC. The packet format is as follows, with field lengths in bytes. A graphical representation is shown in Fig. 3.2 on the next page.

- (3) literal string "aaa" to mark the packet start
- (1) length of the rest of the packet in bytes
- (1) number of IMU accelerometer readings, a
- (1) number of IMU gyroscope readings, g
- IMU data (present only if $a + g > 0$)
  - $(6 \times a)$ IMU accelerometer readings
  - $(6 \times g)$ IMU gyroscope readings
  - (6) IMU magnetometer reading
- Analog sensor data
  - (12) Mini40 force/torque reading
  - (2) microphone reading
  - (4) zero padding

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
61	61	61	33	01	01	FA	40	FE	40	E1	40	00	A0	00	2A	FF	81	00	00	00	00	00	00
start			L	a	g	IMU acc						IMU gyro						IMU mag					

24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
0F	23	0F	28	0C	79	0F	FB	03	CE	0F	9A	06	82	00	00	00	00	06	4F	00	22	07	F8	06	9C	00	2E	00	48	01
force/torque												mic		pad				acc 1						acc 2						#

Figure 3.2: Graphical representation of Teensy packet structure. A variety of packet lengths are possible: in this example, there is one reading from each of the IMU accelerometer and gyroscope. All data values are hexadecimal.

  - (6) analog accelerometer 1 reading
  - (6) analog accelerometer 2 reading

- (1) packet number modulo 256

Since the Mini40 and accelerometers are analog, there is always a reading available. The IMU sensors (accelerometer, gyroscope, and magnetometer) run at different rates and communicate asynchronously with the Teensy microcontroller, so their readings are buffered and included in the packet when ready. Multiple readings may be queued up and transferred at once, in which case their timestamps will be interpolated (see Sec. 3.2.3 on page 45). The parking lot status is checked only while the packet stream is not running – otherwise the stray byte would cause a dropped packet. It is implemented with three reads of the parking lot limit switches, and one byte is sent back, containing three bits for the parking lot status.

Vicon

When the Vicon motion tracking system is in use, position data is broadcast to a computer running ROS [45], which filters out the desired targets to track and writes timestamped position data to a file. The Vicon service communicates with the aforementioned computer over SSH. When the service is started it starts the recording process, does nothing in its step function (setting the blocking mode to infinite) and downloads the file when it is stopped. The service does not respond to any messages.

3.1.3. Frontend

Our custom software sports dual user interfaces: a text console, for text-based control of the system, and a web server hosting the graphical user interface. The two interfaces are equivalent in power, except that the camera images can be seen only in the web interface.

The operator typically wears the backpack and holds the sensing rig in one hand, leaving only one hand free for controlling the device. Therefore, we designed a user interface that can be used with nearly any smartphone, in the form of a single-page web app. The app is served from a web server running in a separate thread of the supervisor software, so it can directly send and receive messages with the sensor drivers. The interface is shown in Fig. 3.3 on the next page. The operator can easily start and stop individual sensors for testing purposes, as well as performing some actions specific to each (for example, capturing single camera frames and adjusting the capture settings of the Bluefox camera). However, this is not convenient for data collection, since several sensors must be orchestrated to start and stop all at once. Therefore, I designed a "flow" system to make the data collection interface scriptable and reactive to the end-effector connection state.

"Flow" System

This is a somewhat general framework for building interactive experimental paradigms out of the components that we have. They are called *flows* because the specification of an experiment is isomorphic to a flow chart (such as Fig. 3.4 on the following page). Several actions are provided for building up a flow:

- Display a message to the user
- Ask the user a question with a free response
- Ask the user a question with an integer response
- Start reading data from a specific sensor

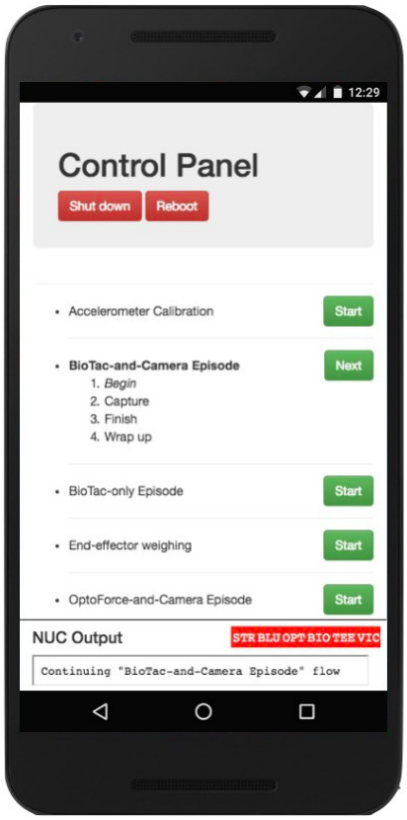


Figure 3.3: Screenshot of web application that controls Proton operation.

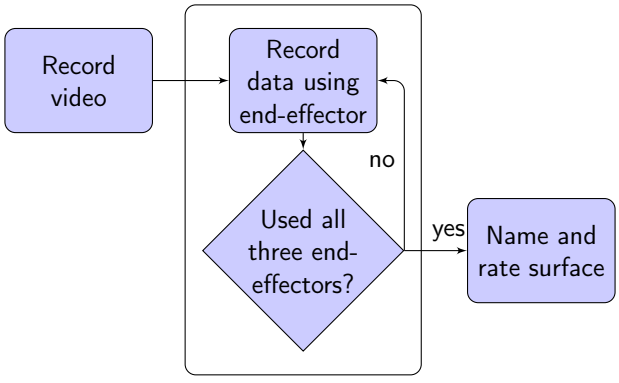


Figure 3.4: Flowchart showing steps for gathering a dataset. An episode consists of recording video, gathering data with each end-effector (boxed steps repeat three times), and human ratings.

- Stop reading data from a specific sensor
- Stop reading data from all sensors

User interaction is performed through the web interface, using WebSockets to update the displayed information without reloading. Notice that there is no explicit action for reading from the parking lot – making such reads automatic is a key feature of the flow system. Each stage of a flow (corresponding to a shaded box in Fig. 3.4 on page 38) contains a *script* of one or more actions, executed in order, and may be *triggered* by a specific end-effector. When looking for the next state to execute, the flow executor scans through the states in order, looking for a state that either has no trigger or whose trigger matches the end-effector currently missing from the parking lot. If zero, two, or three end-effectors are missing, an error is raised. This system is flexible enough to express various types of experiments that can be performed with the Proton, but it automates as much setup as possible so that the experimenter need only be concerned with the end-effectors and the surface under study.

To script the data collection procedure, I created a "flow" file format. It is designed for use both as instructions for collecting data and a log of the collection including timestamps and metadata. Therefore it is human-readable, and I wrote software to parse and transform it in multiple languages (here, Rust and MATLAB). A sample flow file is shown in Listing 3.1 on page 41; this is the flow we used to capture data using the BioTac end-effector. As shown, the first line specifies the title of the flow and the end-effector that must be present for it to be applicable. The rest of the flow then consists of a series of named stages, introduced by a line starting with a dash and followed by an indented *script* listing the actions to be taken when the operator advances to that stage. The type of action continues to be distinguished by the first character of each line:

- Lines enclosed in quotation marks are messages to be displayed to the operator. For example, line 9 displays a message to the operator once all sensors have started up.

- Lines start with an angle bracket are questions the operator must answer. A description of the requested value is given in quotation marks, and optionally a numerical range restricts the possible values (otherwise, it is a free-response question). For example, line 17 asks the operator to type in a name for the surface, while line 18 requests a warmness rating that must be an integer between 1 and 5 inclusive.

- Lines starting with colons are command messages sent to a certain sensor driver (identified by the sensor name after the colon). For example, line 6 switches the Teensy to use an internal voltage reference for its ADC.

- Finally, other lines are actions that the command processor knows how to interpret (e.g., `start`, `stop`).

When starting a flow, an episode directory is automatically created in a folder hierarchy organized by date and flow type. Then the operator advances to each stage by pushing a button. As the flow interpreter runs the script, it records a timestamp for every action, as well as the answers to any operator questions. Then, the flow file is rewritten with this extra information. An example is shown in Listing 3.2 on the next page. Note that every line now has a timestamp (enclosed in square brackets, formatted as a Unix timestamp with nanosecond precision). Additionally, the operator question actions (lines starting with angle brackets) have the answer to the question enclosed in square brackets (for example, the answer for surface name, line 19, was "FixCrete" and the warmness rating, line 20, was 1 out of 5).

```
BioTac-and-Camera Episode: BioTac
- Begin
    "Starting new episode!"
- Capture
    start teensy biotac
    : teensy ref int
    : bluefox disk start
    : structure disk start
    "Now recording!"
- Finish
    : structure disk stop
    : bluefox disk stop
    "Writing to disk, please wait..."
    stop biotac teensy
    "Done!"
- Wrap up
    > "surface name"
    > "cool/warm" (1..5)
    > "soft/hard" (1..5)
    > "smooth/rough" (1..5)
    > "slippery/sticky" (1..5)
```

Listing 3.1: Sample flow file, for a data collection episode using the SynTouch BioTac.

```
BioTac-and-Camera Episode [1504712297.290607214]
- Begin [1504712297.291033506]
    "Starting new episode!" [1504712297.291046381]
- Capture [1504712299.313479662]
    start teensy [1504712299.313504219]
    start biotac [1504712301.313683271]
    : teensy ref int [1504712303.313887596]
    : bluefox disk start [1504712303.313925266]
    : structure disk start [1504712303.313957453]
    "Now recording!" [1504712303.313979149]
- Finish [1504712381.180655003]
    : structure disk stop [1504712381.180758476]
    : bluefox disk stop [1504712381.180809498]
    "Writing to disk, please wait..." [1504712381.180828810]
    stop biotac [1504712381.180950403]
    stop teensy [1504712381.181131124]
    "Done!" [1504712381.181189299]
- Wrap up [1504712383.941423416]
    > "surface name" ["FixCrete"] [1504712383.941451788]
    > "cool/warm" (1..5) [1] [1504712401.482496977]
    > "soft/hard" (1..5) [5] [1504712413.376324892]
    > "smooth/rough" (1..5) [2] [1504712417.176620245]
    > "slippery/sticky" (1..5) [4] [1504712427.793419123]
```

Listing 3.2: Sample flow file, for a data collection episode using the SynTouch BioTac, with information from an episode filled in. The surface under investigation was FixCrete, a fiberglass-infused concrete sample in the Material ConneXion database.

3.2. Postprocessing

After data collection is finished, the goal shifts from writing data to disk to making the data human-readable and interpretable. The first step is *reprocessing* to minimally transform the raw data into a format we can include in the published dataset. This section also describes the software we use for managing stored data, processing common to all machine learning studies that we performed, and data visualization.

3.2.1. Data Reprocessing

During real-time data collection, the onboard computer is constrained by both input-output bandwidth and processing power, so the goal is to record sensor data to disk as fast as possible. Accordingly, the in-memory representations of timestamped data packets are directly serialized to disk, without regard for encoding. Clearly, for analysis and dissemination of data, this representation should be transformed into commonly used and human-readable formats.

The first postprocessing step consists of programs that load these data dumps back into memory and re-encode them for further processing:

- The Structure and Bluefox camera images are encoded in the lossless PNG format. A side table is generated for each camera giving the timestamp of each frame (see Table 3.1 on the following page).

  Additionally, the April Tag tracking algorithm (see Sec. 2.3.1 on page 19) is run over the Bluefox frames so that further motion tracking analysis does not require costly image processing. This step results in a CSV file whose format is detailed in Table 3.2 on the following page.

- The data packets from the Teensy daughterboard contain data from several sensors, so re-encoding results in four CSV files:

Column name	Description
`Frame number`	integer index of the frame captured by the camera
`Filename`	image filename
`Unix timestamp`	time that the frame was captured

Table 3.1: Format of `structure_times.csv` and `bluefox_times.csv` files

Column name	Description
`Frame number`	integer index of the frame captured by the camera
`Tag IDs`	semicolon-separated string listing the integer indices of the April tags found in the image
`Tag centers`	semicolon-separated list of floating-point pairs indicating the pixel location of the center of each tag found
`Tag P1s`	semicolon-separated list of floating-point pairs indicating the pixel location of the upper left corner of each tag found
`Tag P2s`	semicolon-separated list of floating-point pairs indicating the pixel location of the lower left corner of each tag found
`Tag P3s`	semicolon-separated list of floating-point pairs indicating the pixel location of the lower right corner of each tag found
`Tag P4s`	semicolon-separated list of floating-point pairs indicating the pixel location of the upper right corner of each tag found

Table 3.2: Format of `april.csv` files

  - Analog sensors, including Mini40 force/torque data and high-bandwidth accelerometer data (see Table 3.3 on the next page)
  - IMU accelerometer (see Table 3.4 on the following page)
  - IMU gyroscope (see Table 3.5 on the following page)
  - IMU magnetometer (see Table 3.6 on the following page)

- When running, the OptoForce and BioTac sensors have individual CSV files (see Table 3.7 on the next page and Table 3.8 on page 45)

Our custom software scans through a directory of data collection episodes and runs all applicable reprocessing steps. Once the reprocessing is finished, the original raw data files can be deleted.

Column name	Description
`Timestamp`	time that the packet was received
`Teensy dt`	debugging information for helping to detect missing packets
`Packet number`	
`FT0-11`	Mini40 readings
`FT12-13`	Microphone reading
`FT14-17`	zero
`FT18-23`	Accelerometer 1 readings
`FT24-29`	Accelerometer 2 readings

Table 3.3: Format of `teensy.ft.csv` files

Column name	Description
`Timestamp`	time that the packet was received
`FIFO Position`	reading location in queue (see Sec. 3.2.3 on the next page)
`Acc X`	X axis acceleration
`Acc Y`	Y axis acceleration
`Acc Z`	Z axis acceleration

Table 3.4: Format of `teensy.acc.csv` files

Column name	Description
`Timestamp`	time that the packet was received
`FIFO Position`	reading location in queue (see Sec. 3.2.3 on the next page)
`Gyro X`	X axis angular velocity
`Gyro Y`	Y axis angular velocity
`Gyro Z`	Z axis angular velocity

Table 3.5: Format of `teensy.gyro.csv` files

Column name	Description
`Timestamp`	time that the packet was received
`Mag X`	X axis field strength
`Mag Y`	Y axis field strength
`Mag Z`	Z axis field strength

Table 3.6: Format of `teensy.mag.csv` files

Column name	Description
`Timestamp`	time that the packet was received
`Force X`	X axis force
`Force Y`	Y axis force
`Force Z`	Z axis force

Table 3.7: Format of `optoforce.csv` files

Column name	Description
`Timestamp`	time that the packet was received
`PDC`	100 Hz DC pressure reading
`PAC #0-21`	2200 Hz AC pressure readings
`TDC`	Absolute temperature reading
`TAC`	Delta temperature reading
`Electrode #0-18`	Electrode readings

Table 3.8: Format of `biotac.csv` files

3.2.2. Data Management

Our data collection procedure generated a large amount of data: approximately 1 GB per end-effector/surface combination (after the reprocessing detailed in the previous section). We managed this according to the maxim, "if your data isn't backed up, you don't have any data." Accordingly, we developed supporting software to track episodes across multiple hard drives, collating them by treating the (collection date, end-effector, episode number) triple as a unique identifier. It also checks for the presence of raw data files and runs the reprocessing if necessary. By inspecting the flow files (see Sec. 3.1.3 on page 37), it is possible to emit a list of scanned surfaces and identify any missing end-effector/surface combinations.

3.2.3. MATLAB Postprocessing

A majority of the data analysis for the investigations presented in this thesis was done in MATLAB R2014b and R2017b. Accordingly, I developed a suite of MATLAB functions for loading, manipulating, and presenting the data. A portion of this code will be distributed with the dataset as sample code.

Loading

Presented with a directory containing data recorded during a surface interaction, the first order of business is to load it, decode the sensor readings and apply calibration offsets.

Timestamp Interpolation The digital sensors are mostly straightforward to read, since the readings can be loaded directly from the CSV files detailed in Tables 3.4 through 3.8 on

page 44 and on page 45. In the case of the IMU accelerometer and gyroscope, recall that several readings may be batched together in a single data packet, since the sensors run at different rates. Therefore, there will be several readings with the same timestamp, namely, the time at which the composite packet was received from the Teensy daughterboard. To remedy this, we assume that the sensors themselves run at a consistent rate, and linearly interpolate the timestamps between packets.

ADC Decoding and Scaling For the analog sensors, the values in the CSV files are still somewhat raw. The Teensy daughterboard has two MAX1230BCEG+ ADCs, which are both 12 bits wide. Therefore, during postprocessing we implement conversions described by the following equations. First, to decode the values in the `FT0-FT29` columns of Table 3.3 on page 44, we combine adjacent columns by bit-shifting (3.1). For Mini40 readings, the sensor outputs pairs of voltages and we configure the ADC for bipolar differential conversion, so we need to perform a further sign conversion (3.2). Meanwhile, the accelerometer and microphone readings are biased to 2.5 V, which reads as 2048, so we only need to subtract that from the decoded value.

$$u_i = 2^8 r_{2i} + r_{2i+1} \tag{3.1}$$

$$s_i = \begin{cases} u_i & \text{if } u_i < 2048 \\ u_i - 4096 & \text{if } u_i \geq 2048 \end{cases} \tag{3.2}$$

$$m_i = \frac{s_i}{500} \tag{3.3}$$

$$a_i = 29.43 \frac{u_i - 2048}{4096} \tag{3.4}$$

Furthermore, each analog sensor has a scale factor that we must apply. For the Mini40, we simply divide by 500. To convert the accelerometer readings to g-force, we subtract the reference voltage and multiply by a constant. The constant is derived from the formula $\frac{3g}{4096}$, where $g = 9.81\,{}^{m}/_{s^2}$ and the accelerometers have a range of $\pm 3g$.

Mini40 Calibration The Mini40 has an internal calibration matrix which is measured during sensor construction and provided by the vendor as a 6×6 transformation matrix $\mathbf{C}$. Furthermore, we measure the force and torque bias $(\vec{b}_F, \vec{b}_\tau)$ of our sensor using the procedure described in Appendix A.2 on page 161. These two calibrations are applied as shown below.

$$\begin{bmatrix} \vec{F} \\ \hline \vec{\tau} \end{bmatrix} = \mathbf{C}\vec{m} - \begin{bmatrix} \vec{b}_F \\ \hline \vec{b}_\tau \end{bmatrix} \tag{3.5}$$

Processing

Converting the sensor readings into physical units is just the first step. In order to make sense of the readings, we perform several postprocessing steps to align the various sensor data streams.

Upsampling and Filtering The first order of business is to have all data on the same time axis. Every data point is timestamped according to the same reference point (namely, the Unix epoch), but each sensor stream starts a slightly different time and runs at a different rate. We use the time axis of the analog sensors as a baseline and upsample all other data to that.

Coordinate Frame Transformation The Proton is a rigid structure, yet it is at times convenient to use various points as the origin of a local coordinate system. We refer to four local frames:

- the ATI Mini40 six-axis force/torque sensor measures forces and torques in its own local frame $\mathcal{F}$;
- the Vicon marker frame $\mathcal{M}$ is tracked by the Vicon system;
- the end-effector frame $\mathcal{E}$ is anchored at the center of the tooling ball end-effector;

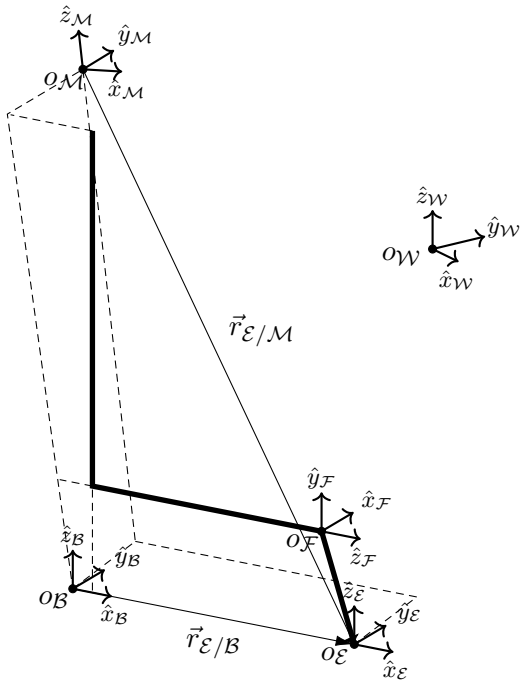


Figure 3.5: Coordinate frames. The thick line shows an outline of the Proton (not to scale). The world frame $\mathcal{W}$ is stationary, and the other four frames are rigidly attached to the rig. The body frame $\mathcal{B}$ and the end-effector frame $\mathcal{E}$ are aligned and have collinear x-axes. The origin of $\mathcal{B}$ is in the yz-plane of the Vicon marker frame $\mathcal{M}$. Reprinted from [2] © 2016 IEEE.

- and the body frame $\mathcal{B}$ is in between: aligned with the end-effector frame but with its origin in the Vicon marker frame's yz plane.

As shown in Fig. 3.5, the body frame is defined with its xy-plane parallel to that of the IMU, its $\hat{z}$-axis pointing up along the Proton handle, and its $\hat{x}$-axis pointing out toward the end-effector. The world frame is stationary, and we typically assume that the surface under investigation is stationary in the world frame, whereas the other four frames are rigidly attached to the rig, and all move together. The Vicon tracking system reports the time-varying position and orientation of the marker frame with respect to the world frame, $H^{\mathcal{W}}_{\mathcal{M}}(t)$. The Mini40 measures forces and torques referenced to its frame $\mathcal{F}$, which is oriented at a known 90-degree angle to the body frame. Therefore, we will transform the Vicon position measurements to find the pose of the body frame. We use two calibration

```
% Marker frame pose is stored in v, a matrix where the first
    column is a timestamp and the rest are a pose vector
% Transformed pose will be stored in vend
vend = v;
for i=1:size(v,1)
    tf = xfconv(v(i,2:7)) * Hvb * Hib;
    vend(i,2:4) = tf(1:3,4);
    vend(i,5:7) = xfconv(tf(1:3,1:3));
end
```

Listing 3.3: The Vicon system measures $H_{\mathcal{M}}^{\mathcal{W}}(t)$, the pose of the marker frame relative to the world frame. This code transforms that to $H_{\mathcal{E}}^{\mathcal{W}}(t)$, the pose of the end-effector.

```
% Force/torque readings are in FT, a matrix where the first
    column is a timestamp, the next three are force, and the
    final three are torque
% Transformed force/torque will be stored in FTworld, and
    finally gravity-compensated force/torque in FTworldcomp
FTworld = FT;
FTworldcomp = FT;
fg = mass * [0; 0; -9.81];
for i=1:size(FTbody,1)
    tf = xfconv(vend(i,5:7));
    FTworld(i,2:4) = tf * Hmb(1:3,1:3) * FT(i,2:4)';
    FTworld(i,5:7) = tf * Hmb(1:3,1:3) * FT(i,5:7)';
    FTworldcomp(i,2:4) = FTworld(i,2:4) - fg';
    FTworldcomp(i,5:7) = FTworld(i,5:7) - cross(tf*com, fg)';
end
```

Listing 3.4: The Mini40 produces readings in its local frame. We transform them to reference the world frame.

procedures to recover different parameters: one with the end-effector fixed, and another with the end-effector free.

We obtain transformation matrices between these frames from a combination of computer-aided design (CAD) modeling and the empirical calibration procedures described in the next chapter. These are used in the postprocessing steps to reinterpret force, torque, and position measurements such that they are all referenced to the world frame. Listings 3.3 and 3.4 shows this code.

3.3. Summary

This chapter detailed several layers of software involved in the Proton's real-time operation, offline processing and analysis of data. Our custom visualization software is covered in Chapter 5. The software, much of it custom-written, will be made available online. These details should serve as a manual for future researchers working with the published dataset or modifying the Proton itself.

CHAPTER 4 : SURFACE DISCRIMINATION WITH ENGINEERED FEATURES

The eventual goal of our project, and the main reason for designing and building the Proton Pack, is to give robots the ability to make haptic inferences about their environment. This task does not necessarily require *a priori* identification of objects in the environment. A top-down approach is certainly possible – for example, a surface that looks like wood is likely hard, metallic surfaces are generally cool to the touch, and a surface attached to an object identified as a sofa will be soft. However, this approach can be fooled, and it is equally possible to imagine criteria for inferring these properties that do not involve object identity: roughness can correspond to spatial frequency or reflectance, while shape may give clues as to hardness.

Neither approach is infallible, but in our machine learning work we took the second approach, opting to infer haptic properties directly from visual input, bypassing surface classification. Despite this choice, data collected with the Proton can of course be used for surface classification. Such data was used in several experiments to validate the effectiveness of data collection and fine-tune the data collection procedure. Classification into categories is a common task in the literature, and it is easier to implement and test than regression to continuously varying properties. By classifying small sets of surfaces using haptic data collected with our device, we can compare accuracy results with other researchers and between different design iterations of the Proton.

Portions of this chapter are adapted with permission from papers that were published in the proceedings of MFI [2], ICRA [3], and WHC [6] (© 2016, 2017, 2017 IEEE respectively).

4.1. Background

Recently, haptic sensing technology has been advancing faster than our strategies for interpreting the resulting data. Romano and Kuchenbecker [19] performed classification among a closed set of fifteen surfaces by training an ensemble of one-class ν-SVMs (one for each surface) on features that included normal force, end-effector speed, and frequency-binned vibration power. Yang et al. [46] applied deep learning over similar data to improve upon those results. Strese et al. achieved even better results using features inspired from traditional audio processing such as cepstral analysis [4].

4.2. Types of Features

When classifying surfaces, we start with several data streams (e.g., position, orientation, force, and acceleration vs. time) and we want to end up with a surface label (a single number identifying one member of the predefined set of surfaces). For each sample the data streams will have a different length. Any number of machine learning tools can be brought to bear for classification, such as support vector machine (SVM), logistic regression, and more, but they can't deal with a variable number of inputs, and besides, feeding in thousands of data points would overwhelm nearly any algorithm with irrelevant data. To solve this problem, machine learning pipelines typically calculate "features" over the input data. These are properties that are characteristic of the sample, and ideally highly correlated with the labels. For example, one feature might be the average vibration power in the 100-200 Hz frequency range. Another could be the average ratio of normal force to tangential force while the end-effector was in contact with the surface and the movement speed was nonzero – i.e., the coefficient of kinetic friction.

These two example features differ in an important way – the latter depends on knowing the orientation and movement speed of the end-effector, in order to decompose the force measurement into normal and tangential components as well as identifying periods of active motion. The Proton expends significant hardware and software resources to measure these

properties, which we call *scan-time parameters* because they are not inherent properties of a surface, but important properties of a particular data sample. We call features that depend on these parameters *scan-dependent features*. Other features are *scan-free*. Scan-free features can be used on data where the scan-time parameters are not available (for instance, because they were not measured, in the case of a device such as the Texplorer [4]), and could be more robust to experimental variation since they are deliberately crafted to compensate for the effects of the scan-time parameters (see Sec. 4.6 on page 72).

4.3. Scan-Dependent Feature Pipeline

These features are based on the vibration generated from the end-effector/surface contact, though we also need force and motion tracking in order to do gravity compensation. The original sensor data is in several streams: force and accelerometer readings referenced to the body frame, plus end-effector pose measurements referenced to the world frame.

There were several processing steps before the handoff to machine learning. First, the sensor streams were synchronized (this includes upsampling the Vicon data to match the 3000 Hz sample rate of the ATI Mini40 six-axis force/torque sensor and accelerometers, as described in Sec. 3.2.3 on page 47) and automatically trimmed. Next, referring to Fig. 4.1 on the following page, the end-effector pose measurements $H_{\mathcal{M}}^{\mathcal{W}}(t)$ were transformed to give the pose of the body frame $H_{\mathcal{B}}^{\mathcal{W}}(t)$.

$$\begin{bmatrix} \mathbf{R}_{\mathcal{B}}^{\mathcal{W}}(t) & \vec{x}_{\mathcal{B}}^{\mathcal{W}}(t) \\ \vec{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathcal{M}}^{\mathcal{W}}(t) & \vec{x}_{\mathcal{M}}^{\mathcal{W}}(t) \\ \vec{0} & 1 \end{bmatrix} \mathbf{H}_{\mathcal{B}}^{\mathcal{M}} \tag{4.1}$$

The force and acceleration readings, measured with respect to the body frame, were rotated to reference the world frame. Since the directionality of vibration is not important to human

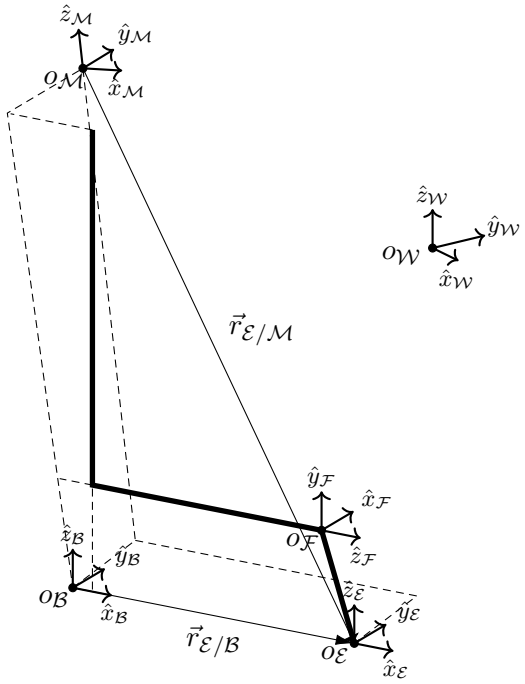


Figure 4.1: Coordinate frames (repeated from Fig. 3.5 for reference)

haptic perception, the acceleration readings are combined using the DFT321 algorithm [47].

$$\vec{F}^{\,\mathcal{W}}(t) = R_{\mathcal{B}}^{\mathcal{W}}(t)\ \vec{F}^{\,\mathcal{B}}(t) \tag{4.2}$$

$$\vec{a}^{\,\mathcal{W}}(t) = R_{\mathcal{B}}^{\mathcal{W}}(t)\ \vec{a}^{\,\mathcal{B}}(t) \tag{4.3}$$

$$\widetilde{a}^{\,\mathcal{W}}(t) = \mathrm{dft321}(\vec{a}^{\,\mathcal{W}}(t)) \tag{4.4}$$

The world-frame superscripts will now be dropped to reduce clutter.

Once in the world frame, where the direction of gravity is known, we can perform gravity compensation on the force measurements to remove the contribution from the end-effector's own weight.

$$\vec{F}_c(t) = \vec{F}(t) + m\vec{g} \tag{4.5}$$

$\vec{F}_c(t)$ can be decomposed into normal force (the vertical component, since each sample was found or placed on a level surface), and tangential force (the horizontal component). To isolate friction from any side-to-side acceleration of the end-effector caused by surface contours, the tangential force is projected onto the opposite of the tangential velocity vector. The velocity was obtained by numerically differentiating the in-plane position data and adding a small correction due to angular velocity $\vec{\omega}$ of the end-effector (modeled as a sphere of radius r).

$$x_N(t) = \vec{x}(t) \cdot \hat{z} \tag{4.6}$$

$$\vec{x}_T(t) = \vec{x}(t) - x_N(t)\hat{z} \tag{4.7}$$

$$\vec{v}_T(t) = \frac{d}{dt}\vec{x}_T(t) + \vec{\omega} \times (-r\hat{z}) \tag{4.8}$$

$$F_{c,N}(t) = \vec{F}_c(t) \cdot \hat{z} \tag{4.9}$$

$$F_{c,T}(t) = (\vec{F}_c(t) - F_{c,N}(t)\hat{z}) \cdot (-\vec{v}_T(t)) \tag{4.10}$$

These transformations result in decomposed force, tangential velocity, and combined vibration measurements across time, which are further used to calculate the scan-dependent features.

4.4. Initial Proof-of-Concept Experiment

After completing the calibration process (see previous chapter and Appendix A), we conducted a proof-of-concept surface discrimination experiment to test the quality of the data collected using the Proton. This section details the experimental design and results.

We configured the Proton with the tooling ball end-effector and gathered contact force, vibration and end-effector pose data from the following five surfaces: acrylonitrile butadiene styrene (ABS) plastic, paper plate, folder (thin cardboard), MDF, and canvas. The surface samples are shown in Fig. 4.2 on the next page and come from the dataset in [48]. We

Figure 4.2: Surface samples used in the 2016 proof-of-concept experiment: "ABS Plastic", "Paper Plate 1", "Folder", "MDF", and "Canvas 2" from the Penn Haptic Texture Toolkit [48]. All materials are mounted on a hard plate.

purposely chose some surfaces that feel distinct and others that feel similar to one another. Motion tracking was done using the Vicon system as it was the only one implemented at the time, and it allowed us to focus on validating the fidelity of the other sensors while generating ground-truth data for the subsequent work on onboard motion tracking (see Appendix A.1 on page 159).

During data collection, the Proton end-effector was dragged in straight lines back and forth over the surface, varying the speed and applied normal force. There were five trials for each surface, each ten seconds long. The surface was placed on a horizontal table, so it was known to be parallel to the world xy-plane. Since the Vicon tracker's output stream is not synchronized with the other sensor streams, each trial started and ended with two taps. These taps can be clearly identified in both the force and position data, so the streams can be manually aligned in time (for an example of data aligned using this method, see Fig. 4.3 on the next page).

Each example for training or testing the machine learning algorithm was a 50 ms window. To avoid training the classifier on periods of time where the operator was changing direction, or while the end-effector was not in constant contact with the surface, these windows were selected from periods where the end-effector was moving with at least 20 mm/s of speed and contacting the surface with at least 3 N of normal force. From 250 s of data, 103.75 s met these thresholds, which yielded 2075 time windows. These were randomly partitioned into a training set of 1689 examples and a test set of 386 examples.

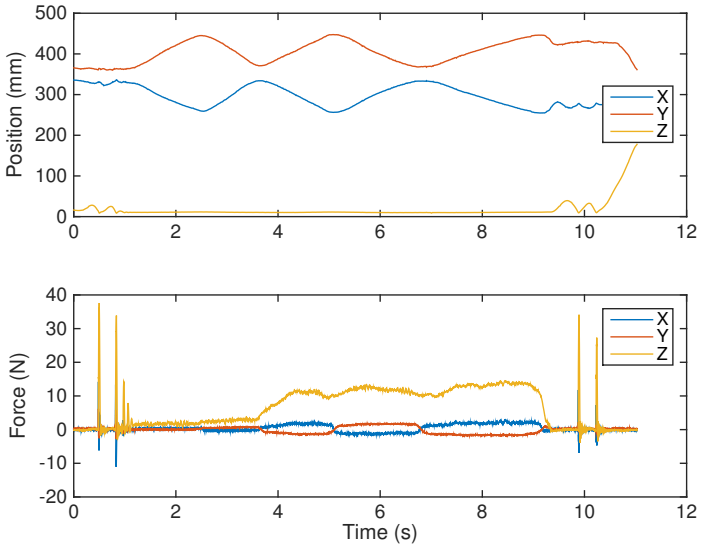


Figure 4.3: Typical position (top) and force (bottom) data generated by dragging the Proton's tooling ball end-effector across a folder texture sample. Taps, evident at the beginning and end of both plots, were used for temporally aligning the data streams. Reprinted from [2] © 2016 IEEE.

The learning was done with a multi-class ν-SVM [49] using a radial basis function (RBF) kernel, parametrized by ν (allowed misclassifications) and γ (inverse variance of the kernel). For each example we calculated binned acceleration frequencies, following the "perceptual binning" scheme of [19], and summary statistics (mean and standard deviation) of the normal force, F_N, tangential force, F_T, and speed, v. This feature extraction was parametrized by n (number of frequency bins) and α (width of the frequency bins, described in [19]). To choose optimal values for ν, γ, n, and α, we performed grid search with five-fold cross-validation.

Fig. 4.4 on the following page shows a visualization of the normalized features in the training set, compared between two surfaces with different haptic properties. Features are normalized by subtracting the mean and dividing by the range (calculated from the training set). The first five features are the frequency-binned vibration power, followed by mean and standard deviation of normal force, tangential force, and speed respectively. It appears that each surface has a distinctive vibration fingerprint, in terms of average amplitude as well as dependence on force and speed.

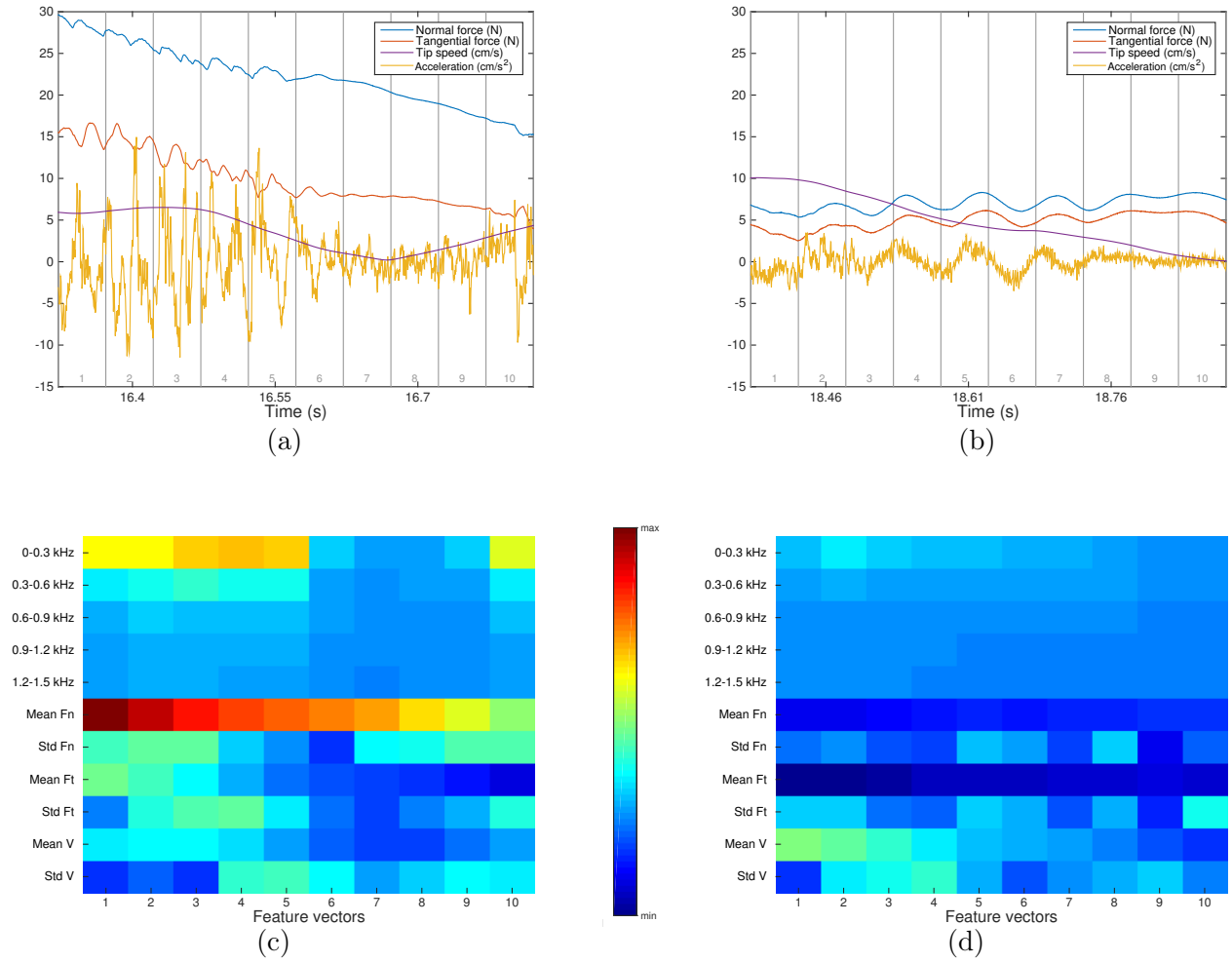


Figure 4.4: Visualization of the feature extraction process for 0.5 s snippets of data from ABS plastic (left) and vinyl (right), using the best-performing parameters from Table 4.4 on page 65. Top row: time-domain force, vibration and speed data, with boundaries of the 50 ms chunks shown in gray. Bottom row: feature vectors, including frequency bins and summary statistics of force and speed. Each chunk in (a) and (b) corresponds to one row in (c) and (d). The of feature vectors are scaled by subtracting the mean and dividing by the range. For display (though not for the machine learning) the frequency bin features are scaled together, so their magnitudes may be directly compared. Reprinted from [3] © 2017 IEEE.

Parameter	Minimum	Maximum	Increment	Best value
ν	0.05	0.4	0.05	0.25
γ	1	20	2	7
n	1	10	2	3
α	0.1	0.5	0.05	0.15

Table 4.1: Best learning parameters, as decided by cross-validation. Reprinted from [2] © 2016 IEEE.

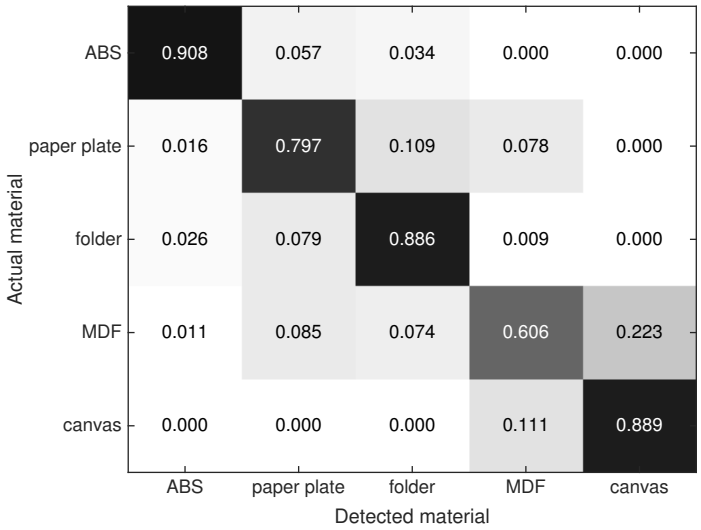


Figure 4.5: Confusion matrix. Entries are normalized by the total number of samples of each material. Reprinted from [2] © 2016 IEEE.

The grid search and cross-validation revealed the best choices of the parameters, shown in Table 4.1, which resulted in 85.2% average accuracy on the validation set and 82.5% accuracy on the test set. Numerical measures of the model performance on all five materials, slightly inflated due to treating the five-class SVM as a one-vs-all classifier, are shown in Fig. 4.5 and Table 4.2 on this page and on the next page. Performance is significantly higher than chance (20% accuracy), although we see some confusion among surfaces, especially the paper plate and folder. Compared to [19] (72.4% accuracy among fifteen surfaces) and [4] (95% accuracy among 69 textures), our results were less impressive – we achieved higher absolute accuracy than [19], but with a smaller set of surfaces. We believe the large tooling ball evoked less interesting signals than the smaller tool tips used in these previous studies. As seen in the next section, much better accuracy was achieved after improving hardware design issues.

Surface	Precision	Recall	F_1 score
ABS	0.940	0.908	0.924
paper plate	0.699	0.797	0.745
folder	0.856	0.886	0.871
MDF	0.740	0.606	0.667
canvas	0.842	0.889	0.865

Table 4.2: Test set performance of the best model from cross validation. Reprinted from [2] © 2016 IEEE.

4.5. Design Improvements

Owing to the unsatisfying performance observed in our first proof-of-concept experiment, we went back to the drawing board on several aspects of the Proton. After fixes to the accelerometer hardware and end-effector size (see Secs. 4.5.1 through 4.5.2 on page 63), we repeated the experiment with five different surfaces. These improvements were combined with the original plan to switch from external Vicon motion tracking to onboard tracking using the Bluefox camera (Sec. 4.5.3 on page 66).

Using a procedure similar to the one outlined in [2], we recorded 30-second interactions between each of two end-effectors and the five surfaces, for a total of 10 interactions and 300 seconds of data. The end-effectors were tooling balls of different diameters. Each surface was affixed to a horizontal table. The end-effector was attached to the Proton 2 and dragged across the surface by a human experimenter, deliberately varying speed, direction and normal force over time. During post-processing, the Vicon motion tracking output was used to estimate the end-effector speed, and to transform the force readings into the world frame. The three-axis acceleration readings were merged into a single dimension using the DFT321 algorithm [47], and the mean was subtracted to reveal pure vibration of the end-effector. For machine learning, the 30 s recordings were chopped into 50 ms chunks, each of which was treated as a labeled example of the surface from which it was recorded. These chunks were randomly divided into a training set, consisting of 80% of the data, and a held-out testing set, consisting of 20%. Finally, feature vectors were formed from the mean and standard deviation of the end-effector speed, normal force and tangential force,

Figure 4.6: Interim and final designs of 6.35 mm tooling ball end-effector with external accelerometers.
Left: bottom view. Middle: side view. Right: Final design with reprinted plastic fixture. Reprinted from [3] © 2017 IEEE.

plus a frequency histogram of the vibration power. They were normalized by calculating the training set mean and range by column, subtracting the mean and dividing by the range, constraining all feature values to the range $(-1, 1)$. Using these vectors, we trained a multi-class support vector machine (ν-SVM [49]). This processing and machine learning were repeated as a benchmark for the various improvements that we tested in work published in the proceedings of ICRA 2017 [3].

One important feature used to model surfaces is the vibration power at various frequencies during interaction. We previously followed the machine learning pipeline of Romano and Kuchenbecker [19], including their perceptual frequency binning strategy, in which the power values are combined via sums weighted by Gaussian coefficients. Though it led to a slight benefit in classification accuracy for that work, on further investigation we found that the more complex binning strategy sometimes provides no benefit, so in this work we also evaluated a simpler linear binning strategy, in which vibration powers are divided evenly and summed as in a histogram. However, we found little difference or even a slight advantage to retaining the perceptual binning.

Figure 4.7: Surface samples used in the 2017 validation experiment: "ABS Plastic", "Glitter Paper", "Silk 2", "Vinyl 2", and "Stained Wood" from the Penn Haptic Texture Toolkit [48]. All materials are mounted on a hard plate. Reprinted from [3] © 2017 IEEE.

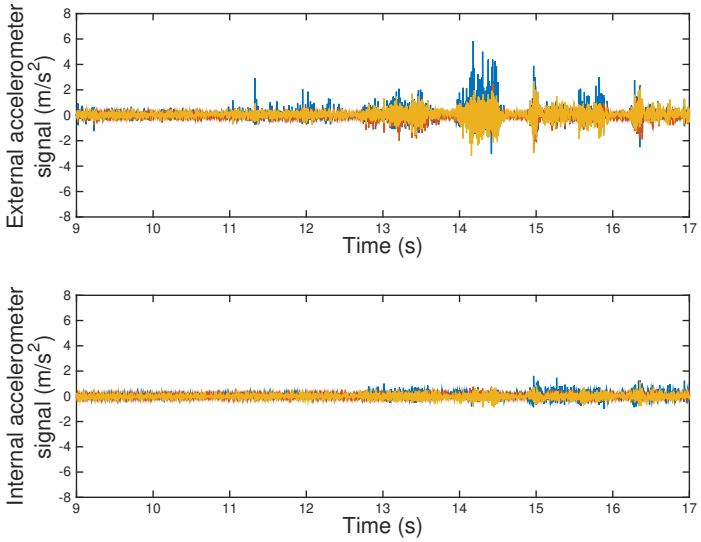


Figure 4.8: Top: typical readings from externally mounted accelerometer. Bottom: typical readings from internally mounted accelerometer. These recordings were taken at the same time, showing clearly that internal accelerometers receive muffled vibrations. Reprinted from [3] © 2017 IEEE.

4.5.1. Accelerometer Placement and Filtering

The performance of the Proton in the first proof-of-concept may have been degraded by the low-pass filtering applied to the accelerometer signals by the breakout boards to which they are attached (Adafruit product ID 1018). We expect valuable information about the surface to be present in the high-frequency vibration data [19, 4]. Accordingly, the output capacitors were changed from their original value of 0.1 uF to 4.7 nF, increasing the acceleration bandwidth from 50 Hz to 1064 Hz, which is comparable to the bandwidths of the accelerometers used by Culbertson et al. [8, 39] and Romano and Kuchenbecker [19].

Even after this change, we found that the acceleration signals were muffled (were low magnitude) when the accelerometers were mounted inside the plastic handle. We conjecture that this muffling stems from several factors. Considering the handheld rig as a rigid body, the center of rotation is likely closer to the handle than to the end-effector, so the acceleration felt at the handle will be lower. Furthermore, the vibration signal must propagate from the point of contact through several centimeters of plastic and a few material interfaces to reach the internal accelerometers, which incurs attenuation. To fix this problem, we moved the accelerometers to the outside of the end-effector, very close to the point of contact. Fig. 4.6 on page 61 shows the modified end-effector. Plotted in Fig. 4.8 on page 62 is a section of data gathered while dragging the end-effector on ABS plastic, with one accelerometer in its original position and the other mounted externally. The difference in sensitivity is striking.

4.5.2. Tooling Ball Size

Based on the range of texture probes used in the literature, we elected to equip the Proton with smaller tooling ball end-effectors to capture contact vibrations with higher fidelity.

When a human or robot contacts a surface through a tool, the geometry of the tool affects the resulting sensations [50]. In particular, considering a spherical or hemispherical tip, the tip size strongly affects the size of the surface features that can be resolved. Klatzky and Lederman studied roughness perception through thimbles and rigid sticks [51]. That

Device	End-effector diameter
WHaT [30]	1.98 mm
Haptic Camera [8, 39, 19]	1.6 mm, 3.175 mm, 3.25 mm
Haptic stylus [4]	5 mm
Proton 1 [2]	19.05 mm
Proton 2	6.35 mm, 9.525 mm

Table 4.3: End-effector diameter of various haptic sensing systems. Value for WHaT was estimated from Fig. 1 of Pai and Rizun [30]. Reprinted from [3] © 2017 IEEE.

study used probes with contact areas 2 mm and 4 mm wide, and surfaces with inter-element spacing of $0.5 - 3.5$ mm. They found higher roughness sensitivity with the larger probe, although subjects gave higher roughness ratings to surfaces with low inter-element spacing when using the smaller probe (presumably due to larger vibration amplitude). Klatzky et al. further explored the relationship between tip size, inter-element spacing and roughness perception [38], including a condition where the applied force and speed were controlled by the apparatus. They confirmed a significant effect of probe diameter as well as end-effector speed. More recently, Lawrence et al. conducted similar experiments [52] with a tip diameter of 9 mm and gratings with groove widths of $0.125 - 8.5$ mm. Although only one tip diameter was studied, the ratio of groove width to tip diameter can be considered. They similarly found higher perceived roughness at higher groove widths, which would correspond to a smaller probe tip if spatial frequency were held constant.

For direct comparison, we consider the tip diameters of extant haptic surface probes. Table 4.3 compares end-effector diameter between the three pen-shaped haptic sensors that inspired this research, and the two versions of the Proton. The Proton 1 end-effector had a diameter more than ten times larger than those of prior work; the new end-effector diameters are within an order of magnitude. We did not evaluate even smaller diameters because we found they tend to damage the surfaces when attached to the heavy Proton.

We prepared two interchangeable end-effector modules for the Proton. They attach rigidly to the force/torque sensor and hold tooling balls with diameters of 6.35 mm and 9.525 mm. Interactions were recorded with a set of five sample surfaces selected from the Penn Haptic Texture Toolkit [48]: ABS plastic, glitter paper, silk, vinyl, and wood (shown in Fig. 4.7 on

Diameter	ν	γ	n	α	Accuracy Train	Accuracy Test
6.35 mm	0.1	10	50	0.1	84.4%	84.5%
9.525 mm	0.1	30	60	0.2	88.2%	91.5%

Table 4.4: Classification accuracy and optimal hyperparameters for each end-effector diameter using Vicon motion tracking. Reprinted from [3] © 2017 IEEE.

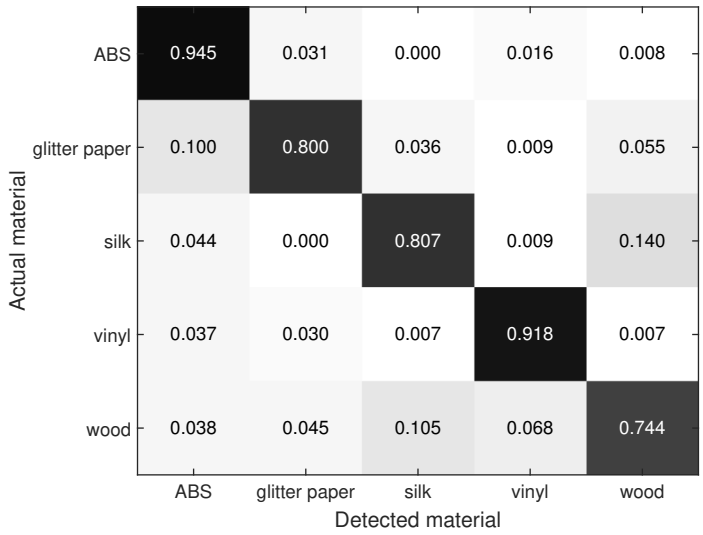


Figure 4.9: Confusion matrix for the 6.35 mm model from Table 4.4 using Vicon motion tracking. Entries are normalized by the total number of samples of each material. Reprinted from [3] © 2017 IEEE.

page 62). This set was selected with the goal of using flat surfaces that would not damage or be damaged by our steel end-effectors, and that represent distinct values of roughness and softness. The ABS plastic and glitter paper are quite rough to the touch, while the wood is smoother. The silk is very slippery, while the vinyl has high friction.

The experiment described at the beginning of this section was performed for each end-effector size using Vicon motion tracking. Table 4.4 shows the training and test set accuracies achieved by the optimal models for each end-effector, as well as the optimal hyperparameter values. Perceptual binning gave equal or better performance in all cases. Fig. 4.9 visualizes the confusion matrix for the 9.525 mm end-effector, quantifying the classification mistakes, and Table 4.5 on the following page shows numerical measures of the same end-effector's performance.

Surface	Precision	Recall	F_1 score
ABS	0.822	0.945	0.879
glitter paper	0.863	0.800	0.830
silk	0.829	0.807	0.818
vinyl	0.904	0.918	0.911
wood	0.805	0.744	0.773
mean	0.845	0.843	0.842

Table 4.5: Test set performance of the 6.35 mm model from Table 4.4 on page 65 using Vicon motion tracking. Reprinted from [3] © 2017 IEEE.

Overall, the classification accuracy was high, with most of the mistakes involving confusion between the wood and silk or the ABS plastic and glitter paper surfaces. Both pairs are quite similar in roughness and friction characteristics. Though the former pair seem like quite different surfaces due to their difference in hardness, this property was not explicitly considered in this experiment.

Among these results, the larger of the two tested end-effectors produces somewhat higher classification accuracies on the training and test sets. In theory, a smaller end-effector should be able to better resolve surface details, but these details may be irrelevant for our classification task. Another factor is the human operator, who tended to exert slightly more force and move slightly slower when using the 9.525 mm end-effector; such differences in exploratory motion can change the difficulty of the task.

4.5.3. Onboard Motion Tracking

Tracking the motion of an object in a camera view, or (equivalently) tracking the motion of a camera with respect to its surroundings, is a common computer vision task. The typical solutions involve identifying distinctive "features" (a different sense of the word from the first part of this chapter!) in the camera view that are likely to still be visible after small motions. Tracking many of these features over time allows the extrapolation of a motion track.

Unfortunately, by design, the Proton's camera will often be pointed at flat, homogeneous surfaces where easy-to-track features may not be available. In order to perform highly accurate motion tracking in a wide variety of scenes, we therefore introduce known fiducial

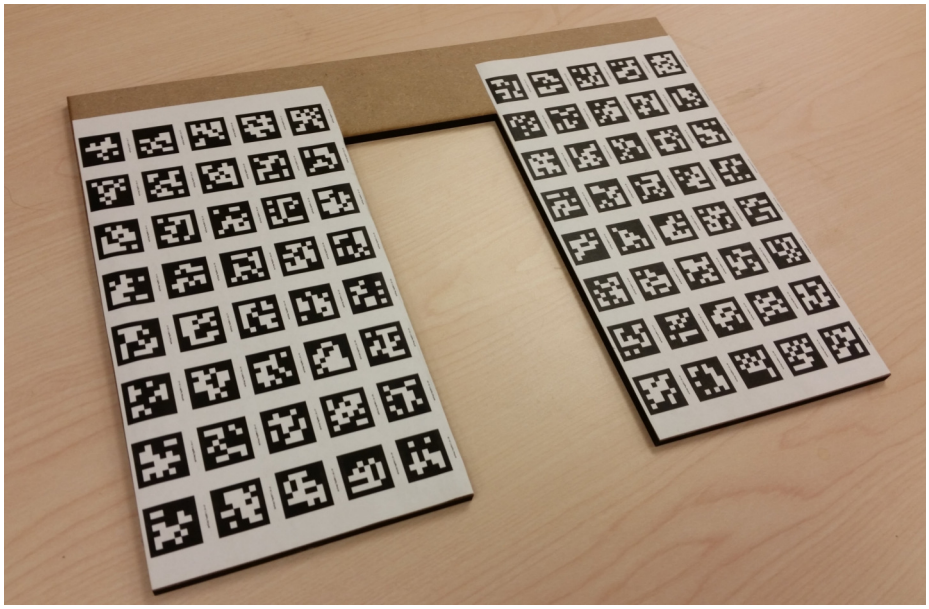


Figure 4.10: Fiducial frame on top of a desk surface. Reprinted from [3] © 2017 IEEE.

markers (AprilTags [53]) into the environment. These markers can be clearly identified with the mvBlueFOX3 camera in a variety of lighting conditions, and being of known size and shape, their position and orientation with respect to the camera can be determined from a single image without a depth sensor. Accordingly, we fabricated a U-shaped frame covered with an array of AprilTags, as pictured in Fig. 4.10. During data collection, this frame is placed over the observed surface so that both the tags and the surface are within the camera's field of view. In this way the mvBlueFOX3 camera is used both for motion tracking and to record the visual appearance of the surface.

Motion tracking is more accurate when given the chance to track more spatially distributed features. Therefore, we use a Computar M1224-MPW2 lens, which has a wider field of view than the Computar M2518-MPW2 used in [2] (30.4° as opposed to 15°), trading off some image detail for better tracking performance.

Optical Motion Tracking Algorithms

The use of AprilTags makes the extraction of stable, rigid tracking points easy. However, it remains to choose an algorithm for turning those tracked points into a camera pose. Four motion tracking algorithms were compared:

PNP The goal of the Perspective-n-Point (PNP) problem [54] is to recover the camera pose given point correspondences between locations in the world frame and the pixel coordinates on the camera's image plane. Fiducial markers provide four points per tag (the tag corners), which are located at known world-frame coordinates (referenced to an origin placed arbitrarily at the corner of the fiducial frame). This algorithm uses a closed-form solution [55] to compute the camera pose at every frame, using all visible tags.

PNP+IMU The fiducial frame contains over 70 AprilTags, so there are usually many visible in every camera image. However, quick motion can induce blur, or the Proton may be tilted at an extreme angle, so that few tags are captured and the PNP solver may output an inaccurate pose or none at all. For this reason, the Proton has an IMU containing a three-axis accelerometer and gyroscope. We integrate the acceleration and angular velocity and fuse with the PNP estimate using an extended Kalman filter (EKF).

OF The PNP solver requires the fiducial markers to be anchored to a world frame. Alternative algorithms can start with an initial estimate of the camera pose and track it from there, using any visible features. We again use the AprilTag corners to find robust features to track, but then we use the Structure depth sensor to find their 3D coordinates. The optical flow (OF) algorithm uses point correspondences to align successive frames and extract the rigid transformation representing the change in camera pose [56]. These incremental transformations are applied cumulatively to the initial pose estimate, making that estimate very important.

Algorithm	PNP	PNP+IMU	OF	SLAM	
Initialization	N/A	N/A	IMU	IMU	PNP
X (mm)	**2.765**	2.899	9.405	20.979	21.823
Y (mm)	**2.682**	2.711	15.453	15.433	16.570
Z (mm)	**1.889**	2.112	16.641	21.465	19.495
Roll (rad)	**0.010**	0.011	0.082	0.048	0.053
Pitch (rad)	**0.012**	0.013	0.081	0.092	0.123
Yaw (rad)	**0.004**	0.006	0.041	0.125	0.103

Table 4.6: End-effector pose errors (average root-mean-square (RMS) error across three runs) on test datasets using each motion tracking algorithm. Reprinted from [3] © 2017 IEEE.

SLAM To jointly track the pose of the camera and the features, a simultaneous localization and mapping (SLAM) algorithm can be used. Similar to the optical flow algorithm, the features do not need to be registered to the world frame. Our implementation uses an error-state Kalman filter (ESKF) tracking the Proton's position, orientation, and velocity, plus the world-frame coordinates of the AprilTag corners [57].

The OF and SLAM methods both require an initial orientation estimate, which can be found by using the IMU to measure the direction of the gravity vector, or by running the PNP solver on the first image frame.

We compared the output of the four motion tracking algorithms with ground truth from the Vicon system during motions that are typical of data collection, with the Proton mostly level and the end-effector moving in straight lines across a flat surface, as well as motion exhibiting more extreme orientations in order to test the limits of the tracking.

We then tested the best of these algorithms in the classification test described in the previous section. Motion tracking was done with both the Vicon system and the onboard camera. For the second experiment, the same machine learning pipeline was run again, using the pose estimates from camera tracking in place of Vicon data.

Results from the motion tracking comparison (see Table 4.6) show that the PNP algorithm performs the best, with RMS tip position errors of about 2 mm and RMS angle errors of about 0.01 rad. Surprisingly, IMU fusion did not help in our testing. Nevertheless, it is

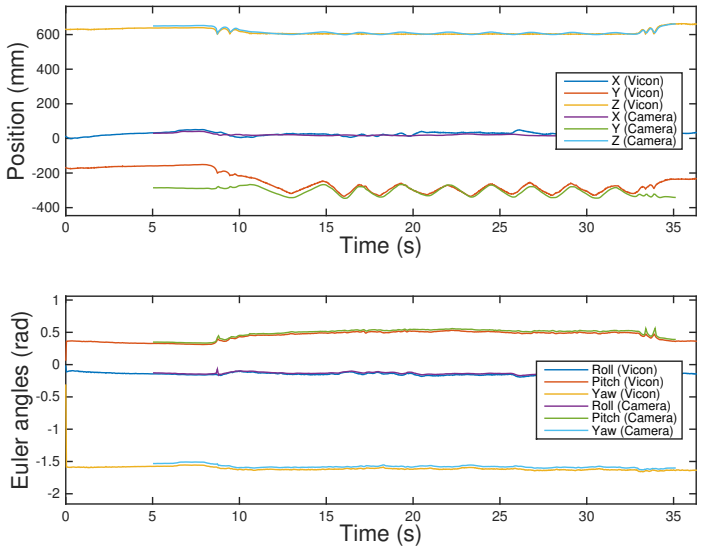


Figure 4.11: Graphical comparison of motion tracking using the Vicon system and onboard camera, during the interaction shown in Fig. 4.8 on page 62. Plots show the pose of the coordinate frame assigned to the tracked Vicon markers. Reprinted from [3] © 2017 IEEE.

possible that better filtering could make it more beneficial, or that varying environmental conditions may make the fusion necessary (e.g., there were no prolonged periods of missing or blurry fiducials in the test data, which may occur during practical data collection). The higher error rates of the OF and SLAM algorithms may be explained by the fact that they operate on less information, since they are not given the world-frame AprilTag locations; also, they persist estimates from frame to frame, compounding any errors, while the PNP estimate is recalculated.

Fig. 4.11 shows the camera pose over time according to the Vicon system and the winning PNP algorithm, during a representative test. Since the two tracking algorithms use different world frames, they have been manually brought into alignment here for ease of visual comparison. Qualitatively, the tracks are nearly identical, confirming the results above.

Having decided on a motion tracking algorithm, we repeated the machine learning analysis again. Motion tracking is quite far removed from the eventual result of a classification score, but it feeds into many of the features used for classification, so we hoped that using the entire pipeline as a benchmark would provide a good validation that onboard motion tracking was

Tracking	Diameter	ν	γ	n	α	Accuracy Train	Accuracy Test
Vicon	6.35 mm	0.1	10	50	0.1	84.4%	84.5%
Vicon	9.525 mm	0.1	30	60	0.2	88.2%	91.5%
Camera	6.35 mm	0.1	10	60	0.1	83.0%	84.7%
Camera	9.525 mm	0.1	90	30	0.15	87.4%	87.7%

Table 4.7: Classification accuracy and optimal hyperparameters vs. end-effector diameter and tracking method. Reprinted from [3] © 2017 IEEE.

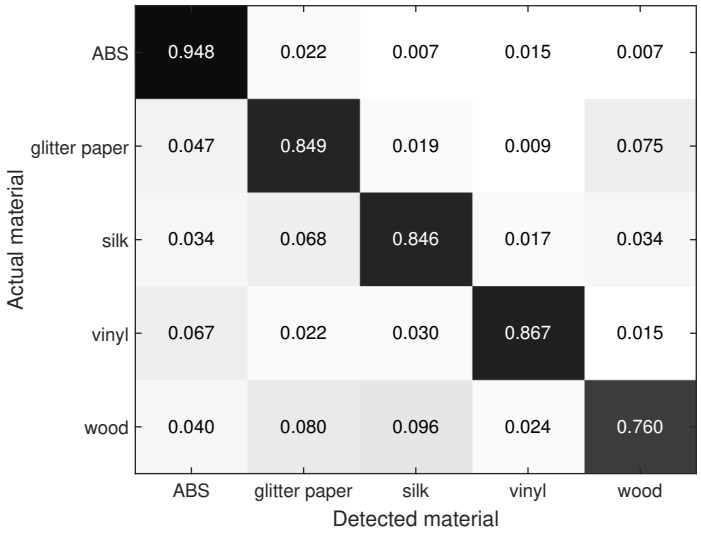


Figure 4.12: Confusion matrix for the 9.525 mm model from Table 4.7 using onboard motion tracking. Entries are normalized by the total number of samples of each material. Reprinted from [3] © 2017 IEEE.

Surface	Precision	Recall	F_1 score
ABS	0.847	0.948	0.894
glitter paper	0.789	0.849	0.818
silk	0.839	0.846	0.843
vinyl	0.936	0.867	0.900
wood	0.864	0.760	0.809
mean	0.855	0.854	0.853

Table 4.8: Test set performance of the 9.525 mm model from Table 4.7 using onboard motion tracking. Reprinted from [3] © 2017 IEEE.

viable. Table 4.7 on page 71 repeats the results from Table 4.4 on page 65 (first two rows), and shows the results of the same experiment with the Vicon motion tracking replaced with onboard camera-based tracking, using the PNP algorithm. A visualization of the confusion matrix for the larger end-effector is shown in Fig. 4.12 on page 71 and analyzed in Table 4.8 on page 71. For the 6.35 mm end-effector, peak test set accuracy increases slightly from 84.5% to 84.7%, while for the 9.525 mm end-effector, the test set accuracy decreases from 91.5% to 87.7%. We consider these results to be comparable. The optimal hyperparameter values are similar, but not identical to their previous values. As before, the perceptual frequency bins are the better choice.

This validation experiment confirmed the effectiveness of onboard motion tracking, and further research did not use the Vicon system.

4.6. Comparison with Scan-Free Features

To advance the state of the art in research, it is beneficial to periodically question the conventional wisdom and cross-pollinate ideas between different research groups. In our lab, previous surface classification experiments [19, 58] used similar hand-engineered features to classify surfaces from haptic data. As described previously, our feature pipeline recognizes that the scan-time parameters have a large effect, and copes with that by measuring them and including them in the feature vector. However, other researchers in the same field are engaged in other feature engineering strategies. For example, Strese et al. [4, 5] engineered a series of features which purport to be independent of scan-time parameters. We tried to do a rigorous comparison of the two approaches, first replicating that lab's classification experiment (using their published data) to verify our implementation of the scan-free features, then comparing classification using each feature type, calculated at several window lengths, on new data collected with the Proton. Finally we verified the claimed property – that scan-free features are actually scan-free. As described in this section, we found that the scan-free features generally live up to their name and are more effective for classification than the scan-dependent approach.

4.6.1. Scan-Free Feature Definitions

Strese et al. [5] identify features by acronyms with a two-letter prefix indicating the sensory modality (A for acceleration, F for friction, and S for sound) and the exploratory procedure (I for impact/tapping, and M for movement/dragging). We implemented all of the given acceleration, friction and sound features, but we excluded the image features to focus on haptic perception. A brief overview of the acceleration feature calculations is given below, largely summarized and reproduced from [4, 5] with a few clarifications and corrections [59], marked with a star ($\star$).

Relevant signals are a_l (acceleration leading up to tapping impact), a_i (acceleration during impact), a_m (acceleration during movement phase), and f_m (friction measurement during movement phase), all sampled at 10 kHz. The acceleration signals were reduced to a single axis using the DFT321 algorithm [47]. Many features depend on a sliding time window, denoted as $\text{win}(k) = a_k ... b_k$ where k starts at 1 and the kth window ranges from a_k to b_k. Additionally, several mathematical functions are repeatedly used: $\sigma(\text{signal})$, the standard deviation of a signal; $\text{lpf}(\text{signal}, f)$, which signifies a low-pass filter with cutoff frequency f; $\text{dct}(\text{signal}, m)$, the discrete cosine transform at a length of m samples; and $\text{sma}(\text{signal}, n)$, a simple moving average of width n samples.

Modified Melfrequency Cepstral Coefficients (AMCC) $\star$

Strese et al. [5] define overlapping sliding windows of width 25 ms and separation 15 ms. The HTK-MFCC library [60] is used to extract coefficients from each window, and the average values of coefficients 2 through 14 are used as the first 13 features.

$$\begin{aligned} \text{win}(k) &= (15\,\text{ms})(k-1) \ ... \ (15\,\text{ms})(k-1) + 25\,\text{ms} \\ \vec{c}^{\,k} &= \text{mfcc}(a_m[\text{win}(k)]) \\ \text{AMCC} &= \frac{1}{N_w} \sum_{k=1}^{N_w} \left[\ c_2^k \mid ... \mid c_{14}^k \ \right] \end{aligned} \tag{4.11}$$

Impact Hardness (AIH)

Strese et al. [5] locate the three highest peaks of the acceleration during tapping impact and average their time indices n_1, n_2, n_3, measured from the time of impact. The hardness feature is the value of the highest peak divided by that average index, and additionally normalized by the sum of the smoothed acceleration during the lead-up to impact (which approximates the impact velocity).

$$\begin{aligned} \overline{n} &= \frac{n_1 + n_2 + n_3}{3} \\ a_{l,filt} &= \mathrm{lpf}(a_l, 12\,\mathrm{Hz}) \\ \mathrm{AIH} &= \frac{\max|a_i|}{\overline{n}} \frac{1}{\sum a_{l,filt}} \end{aligned} \tag{4.12}$$

Impact Spectral Centroid (AISC)

Strese et al. [5] define spectral centroid to be the average of the squared discrete cosine transform (DCT) of the acceleration during impact, weighted by the DCT frequencies.

$$\begin{aligned} I &= \mathrm{dct}(a_i, 4096) \\ \mathrm{AISC} &= \frac{\sum_{k=1}^{\frac{m}{2}} I(f_k)^2 f_k}{\sum_{k=1}^{\frac{m}{2}} I(f_k)^2} \end{aligned} \tag{4.13}$$

Temporal Roughness (AMTR)

Strese et al. [5] measure roughness in the time domain using the Coiflet3 wavelet transform of acceleration. d_1 and d_5 are the reconstructed wavelet coefficients at detail levels 1 and 5. The log mean difference between d_1 and a rescaled version of d_5 aims to capture microscopic

surface features.

$$
\begin{aligned}
d_1 \ldots d_5 &= \text{coiflet3}(a_m) \\
\text{AMTR} &= \log_{10}\left(\overline{d_1 - \frac{\bar{d}_1}{\bar{d}_5} d_5}\right)
\end{aligned}
\tag{4.14}
$$

Spectral Roughness (AMSR) ⋆

Strese et al. [5] evaluate roughness in the frequency domain by comparing changes in the windowed DCT over time.

$$
\begin{aligned}
\text{win}(n) &= (0.5\,\text{s})(n-1) \ldots (0.5\,\text{s})n \\
X_n &= \text{dct}(a_m[\text{win}(n)], 4096) \\
D_k &= X_n - X_{n+100} \\
\text{AMSR} &= \log_{10}\left(\sum_{k=1}^{K} D_k^2\right)
\end{aligned}
\tag{4.15}
$$

Waviness (AMWV) ⋆

Strese et al. [5] use the standard deviation of the difference between two differently filtered versions of the same signal to measure the "deviation of the low-frequency signal slopes".

$$
\begin{aligned}
\text{win}(k) &= (20\,\text{ms})(k-1) \ldots (20\,\text{ms})k \\
a_{m,filt} &= \text{lpf}(a_m, 100\,\text{Hz}) \\
m_k &= \overline{|a_{m,filt}[\text{win}(k)]|} \\
\text{AMWV} &= 1 + \log_{10}\left(\sigma(m - \text{sma}(m, 100))\right)
\end{aligned}
\tag{4.16}
$$

Spikiness (AMSP)

Strese et al. [5] detect spikes using a very wide moving average to obtain an indicator of surfaces with spatially discontinuous features.

$$\begin{aligned} a_{m,filt} &= \mathrm{lpf}(a_m, 100\,\mathrm{Hz}) \\ x_{sma} &= \mathrm{sma}(a_{m,filt}, 5000) \\ x_{th} &= 2\sigma(a_{m,filt}) + \overline{a_{m,filt}} - \overline{x}_{sma} \\ \mathrm{AMSP} &= \log_{10}\left(\overline{x - x_{th}}\right) \end{aligned} \tag{4.17}$$

Fineness (AMF) ⋆

Strese et al. [5] repeat the spectral centroid calculation on the dragging portion of the acceleration signal, averaged over one-second windows.

$$\begin{aligned} \mathrm{win}(q) &= (1\,\mathrm{s})(q-1)\ \ldots\ (1\,\mathrm{s})q \\ I_q &= \mathrm{dct}(a_m[\mathrm{win}(q)], 4096) \\ f_q &= \frac{\sum_{k=1}^{\frac{m}{2}} I_q(f_k)^2 f_k}{\sum_{k=1}^{\frac{m}{2}} I_q(f_k)^2} \\ \mathrm{AMF} &= \frac{1}{Q}\sum_{q=1}^{Q} f_q \end{aligned} \tag{4.18}$$

Regularity (AMRG)

Strese et al. [5] state that autocorrelation of the acceleration signal can reveal repetitive patterns on the surface.

$$\begin{aligned} \hat{x} &= \frac{a_m}{\max a_m - \min a_m} \\ r_k &= \text{xcorr}(\hat{x}) \\ \text{AMRG} &= \overline{\Delta r_k} \end{aligned} \tag{4.19}$$

Unnormalized Friction (Fr)

This feature, used only in the earlier paper by Strese et al. [4], is simply an average of the tangential force.

$$\text{Fr} = \overline{f_m} \tag{4.20}$$

Normalized Friction (FM)

This feature, which Strese et al. introduced in their latter paper [5], improves upon Fr by putting the average acceleration in the denominator, to try to correct for the dependence between normal and tangential force.

$$\text{FM} = \frac{\overline{|f_m|} + \sigma(\Delta|f_m|)}{\overline{|a_m|}} \tag{4.21}$$

4.6.2. Replication Study

We downloaded the Lehrstuhl für Medientechnik Texture (LMT) database [61] and preprocessed it using Strese et al.'s instructions [4, 5]. This dataset includes acceleration, friction and sound data recorded when experimenters tapped and dragged a rigid metal tool on

69 surfaces, divided into training and testing sets each containing ten scans per surface. Tangential force was estimated by measuring the force exerted on the tool handle by the experimenter with a differential pair of force-sensing resistors; this signal may be used to derive approximate friction characteristics, although the scan-time parameters of normal force and tip speed were not measured.

The provided training set contains ten 25 s recordings of acceleration and sound for each surface (including tapping and dragging), plus 100 s of friction data (including only dragging, recorded separately). We divided the friction data into 10 s chunks and associated one with each of the acceleration/sound trials, even though it was not recorded simultaneously. All of the above features were calculated and then standardized by extracting the mean and range of each feature over the training set, then subtracting that mean from both the training and validation features, and dividing by the range.

We trained support vector machines for classification using ten-fold cross-validation (in each fold, nine recordings were used for training, and the remaining one for validation). We observed 87.4% average classification accuracy on the training set. The confusion matrix representing the performance of this classifier is shown in Fig. 4.13 on the following page. Although somewhat lower than the 95% accuracy value reached by Strese et al. [4], this value is comparable. Some discrepancy here is expected and explained by the presumably different cross-validation processes, as well as the differences in the details of feature selection and machine learning algorithms employed. This successful replication validates our implementation of the scan-free features.

4.6.3. Classification

We built on our previous surface classification work [2, 3] by adding the scan-free features invented by Strese et al. [4, 5] to our feature vector.

A dataset was collected using the Proton, containing approximately 20 s of pose, force, and acceleration data while a human operator dragged the tooling ball end-effector back and

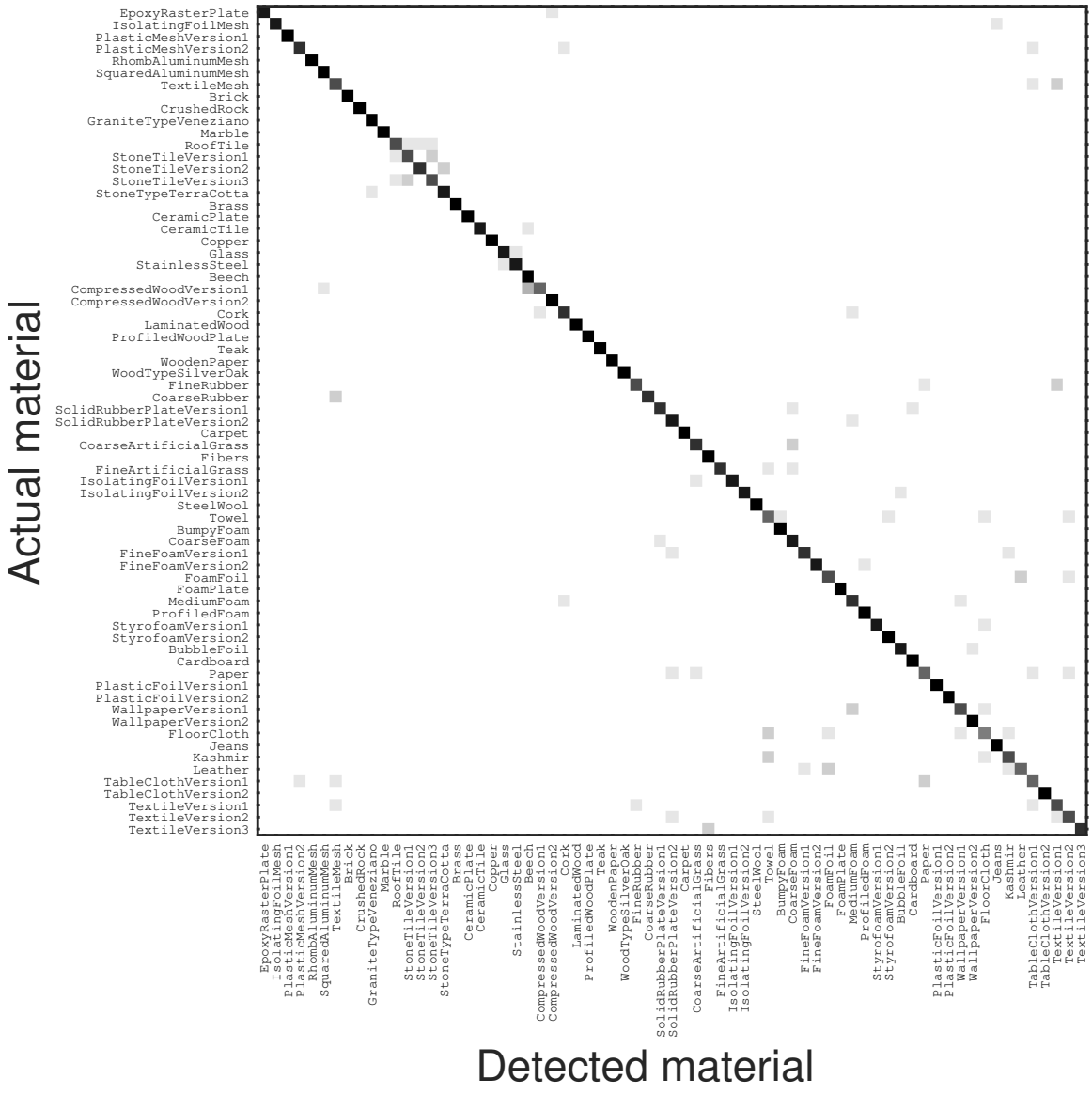


Figure 4.13: Confusion matrix for SVM classifier on the LMT texture database using scan-free features. The trained model, which obtains 87.4% overall accuracy, is a ν-SVM with a RBF kernel, $\nu = 0.1, \gamma = 0.5$. Reprinted from [6] © 2017 IEEE.

forth across each surface at various speeds (up to about 20 cm/s) and normal forces (up to about 25 N). The surfaces are 28 10-cm-square surface patches from the Penn Haptic Texture Toolkit [48]. Following the process in our previous publications [2, 3], we divided the dragging portions of these surface recordings into windows and calculated a feature vector for each window. We tried three window lengths: 0.05 s, 0.25 s, and 1.25 s. The dragging portion was automatically segmented by locating the region of the recording in which the vertical position of the contact point was within 6 mm above its median position.

For this experiment we used a feature vector consisting of the power of the acceleration in regularly spaced frequency bins; mean and standard deviation of normal force, tangential force and tip speed; and the AMCC, AMTR, AMSR, AMWV, AMSP, AMF, AMRG, Fr, and FM scan-free features, defined in Sec. 4.6.1 on page 73. These scan-free features were selected because they operate on the same data as the extant scan-dependent features: acceleration and force data during the dragging phase (tapping was not considered in this experiment). At each window size, we divided the data equally into five folds. Holding out each fold as a test set in turn, we trained a support vector machine (SVM) with a RBF kernel on the remaining four folds. Grid search over the hyperparameters of the ν-SVM (learning parameter ν and kernel width γ) and the scan-dependent features (number, shape, and width of acceleration frequency bins), evaluated using three-fold cross-validation, was used to find the optimal configuration. Table 4.9 on the following page shows the results achieved by the SVM for each window width and set of features. Fig. 4.14 on page 82 shows confusion matrices for classification using a 0.25 s window and, respectively, scan-dependent and scan-free features. Though we have a relatively small set of surfaces, it is clear that the scan-free features are much more effective at classifying the surfaces. In particular, the scan-dependent classifier is weak in distinguishing between certain types of floor tiles and paper-like surfaces, in addition to having more misclassifications overall. Interestingly, the scan-free classifier's confusion between the flannel and acrylic surfaces is striking, but this may indicate that the classifier "sees through" the thin flannel to the rigid plastic backing.

Window	Scan-dependent (%)	Scan-free (%)	All
0.05 s	49.01 ($\sigma = 1.3$)	56.99 ($\sigma = 1.5$)	63.79 ($\sigma = 1.2$)
0.25 s	59.92 ($\sigma = 2.6$)	92.30 ($\sigma = 1.2$)	92.91 ($\sigma = 0.9$)
1.25 s	38.95 ($\sigma = 5.0$)	97.37 ($\sigma = 1.9$)	96.58 ($\sigma = 0.7$)

Table 4.9: Classification accuracies with different feature sets. Chance performance would be 3.6% for 28 surfaces. Reprinted from [6] © 2017 IEEE.

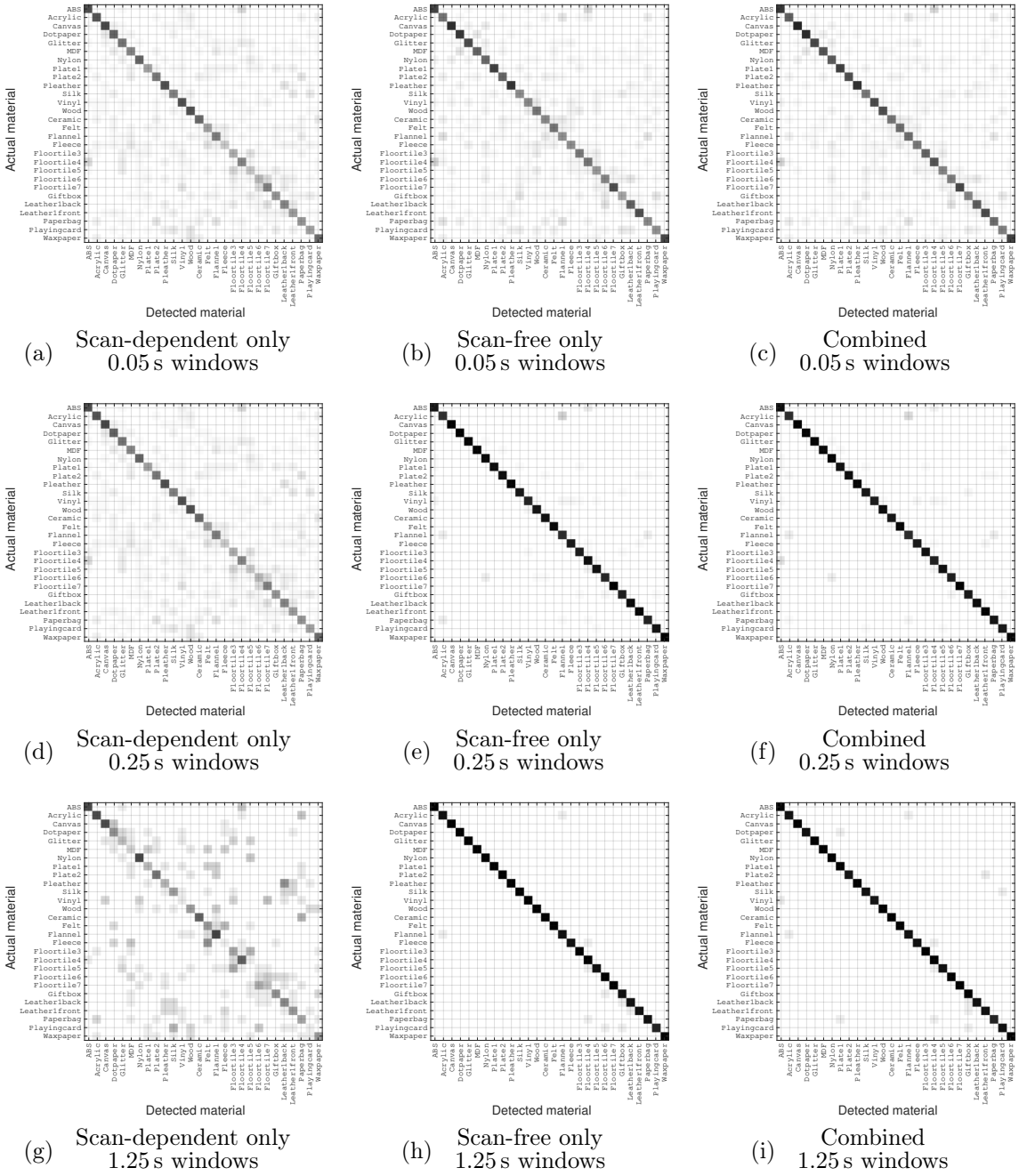
These results show that of the features considered here, the scan-free features of Strese et al. [4, 5] are clearly the better choice for haptic surface classification. However, the scan-free features are about two orders of magnitude slower to calculate in our implementation, so computational resources may be a relevant concern.

4.6.4. Parameter Sensitivity

The development of scan-free features [4, 5] is predicated on the assumption that the engineered features do not depend on normal force or tip speed. However, previous work did not directly measure these parameters, so the assumption is unchecked. In our data, we do have those measurements, so we can check the assumption. Splitting the data into the same 0.05 s, 0.25 s, and 1.25 s windows, we recalculated the scan-dependent and scan-free features on each window.

For each surface and window, we can calculate the average normal force and scanning speed, as well as a value for each feature. We then attempt to fit a bilinear model relating each feature value to these average scan-time parameters. The fit coefficients are not important for our purposes, but the R^2 goodness-of-fit measure will be high when a feature strongly depends on scan-time parameters. The visualizations in Fig. 4.15 on page 84 show the correlation between the scan-time parameters and the two types of features for each surface and window length, as well as average correlations across surfaces.

As expected, the scan-free features are much less dependent on the scan-time parameters in general. However, there are exceptions. In particular, the two friction features Fr and FM are relatively highly scan-dependent, which is consistent with the simplistic two-parameter



(a) Scan-dependent only 0.05 s windows

(b) Scan-free only 0.05 s windows

(c) Combined 0.05 s windows

(d) Scan-dependent only 0.25 s windows

(e) Scan-free only 0.25 s windows

(f) Combined 0.25 s windows

(g) Scan-dependent only 1.25 s windows

(h) Scan-free only 1.25 s windows

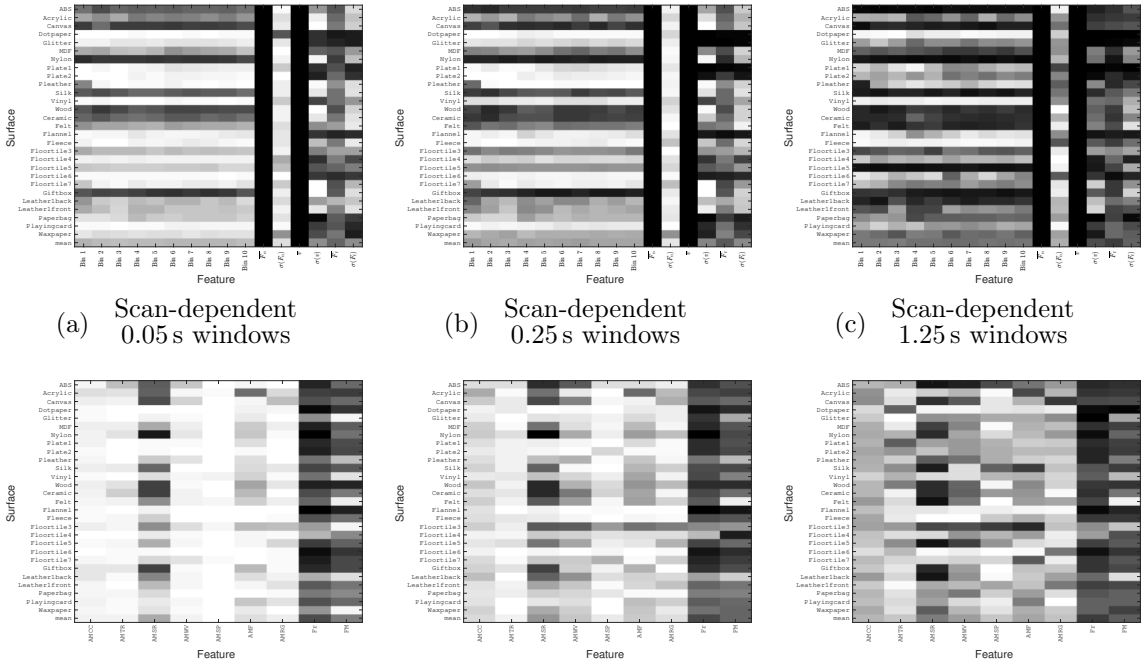
(i) Combined 1.25 s windows

Figure 4.14: Confusion matrices for classifiers using scan-dependent and scan-free features separately and combined, at $0.05 - 1.25$ s window lengths. Reprinted from [6] © 2017 IEEE.

model of friction as a tangential force linearly proportional in magnitude to normal force and tangential speed; adding a normalization based on vibration as in FM reduces but does not eliminate the correlation, most likely because vibration magnitude highly depends on tip speed as well [62]. Also, some purportedly scan-free features are more scan-free than others: using a 0.25 s window, the average R^2 values imply that "spikiness" or AMSP is the least correlated with scan-time parameters, while "spectral roughness" or AMSR is the most (excluding the friction features). However, this pattern varies widely by surface, which underscores the importance of considering a representative training set of diverse surfaces.

Curiously, all features (scan-free and scan-dependent) appear to be slightly more correlated with the scan-time parameters at the longer window length, despite the fact that classification is more effective. A possible explanation is that shorter windows are more sensitive to random sensor noise, making features appear to be less correlated with scan-time parameters than they really are – but that random noise makes classification harder with short windows. Indeed, a purely random feature would be perfectly scan-free but would not enable accurate classification.

The scan-free classification pipeline was highly effective in our surface classification experiment (Sec. 4.6.3 on page 78). Strese et al.'s friction features, which cannot truly measure friction without taking normal force into account, are fairly highly correlated with the scan-time parameters, so it may seem that including them allows information about the scan-time parameters to leak into the scan-free classification pipeline. However, this is not a problem: if the friction features are omitted entirely, the accuracies in Table 4.9 on page 81 remain virtually unchanged. This stability implies that the scan-free acceleration features are most important to the superior classification performance. We speculate that this dominance stems from the highly nonlinear nature of the features, which reflects the nonlinearity of the physical interactions that generate haptic sensations during dragging, as well as the fact that using features inspired by audio engineering draws upon that discipline's decades of experience in identifying salient properties and useful transformations.



(a) Scan-dependent 0.05 s windows

(b) Scan-dependent 0.25 s windows

(c) Scan-dependent 1.25 s windows

(d) Scan-free 0.05 s windows

(e) Scan-free 0.25 s windows

(f) Scan-free 1.25 s windows

Figure 4.15: Feature sensitivity to scan-time parameters. Sensitivities are shown as the R^2 value mapped onto a scale from white (no correlation) to black (100% correlation). For AMCC, the average sensitivity of the cepstral coefficients is used. Note: Figs. 4.15a through 4.15c include the scan-time parameters ($\overline{F_n}$ and $\overline{v}$ columns), so those correlations are always 100%. Reprinted from [6] © 2017 IEEE.

4.7. Summary

This chapter presented the proof-of-concept experiments that established the Proton to be capable of the goal we set for it in the original design: capturing haptic data that effectively characterizes the properties of a surface, without the need of any stationary or grounded hardware.

Moving the accelerometers close to the contact point, removing the low-pass filter, and reducing the end-effector diameter improved surface classification performance from one classification experiment to the next. Onboard motion tracking using the RGB camera and PNP solver achieved comparable performance in an end-to-end proof-of-concept surface classification task, fulfilling the planned goal of breaking the dependence on stationary motion tracking systems.

We also demonstrated that the scan-free features invented by Strese et al. [4, 5] are effective on the provided data and on data collected by the Proton. Furthermore, most of these features are, indeed, not highly correlated with scan-time parameters, which suggests that they can be used effectively in situations where the scan-time parameters are unknown.

Further studies in the area of scan-free features should consider other possible feature sources, including surface appearance and shape, as well as scan-time parameters not considered here that could affect the data, e.g., probe angle.

CHAPTER 5 : DATASET

One of the main contributions of this thesis is a large dataset of matched multimodal surface interaction data. We hope that this dataset, and future datasets like it, will become a basis for researchers to build and test visuo-haptic perception algorithms, and furthermore that this paradigm of dataset sharing and federation, of which the Proton Pack dataset is merely one of the first steps, will be beneficial for the field of haptics as a whole – as it has been for other related research areas.

Computer vision is one such field which has a culture of publishing datasets and organizing research around them. Images and videos are comparatively easy to gather and disseminate. For example, the National Institutes of Standards and Technology published the MNIST Digits Dataset [63], consisting of many scans of handwritten digits 0-9, and invited computer vision researchers to use it to design better OCR software for automated mail processing systems. They did, but beyond that, the digits dataset survives and is often used as an exemplar of an easy classification task, to benchmark a new machine learning algorithm. More recently, through the Internet, researchers can access much more audiovisual data than they could hope to personally collect, owing to websites that host user content, such as YouTube, Flickr, Instagram – though commonly regarded as platforms for sharing content between humans, from the backend these services can be reinterpreted as searchable databases of labeled input (which must be suitably anonymized before analysis). A prominent example is the ImageNet database [64], consisting of 1000 object categories with 1000 images of each. There is an annual competition for classification of this dataset, and this was the setting for the stunning return of artificial neural networks to the forefront of machine learning, with the publication of AlexNET [65] in 2012. The existence of these shared datasets means that researchers in different groups can compare their algorithms against

the same benchmarks. It is hard to imagine the same advancements that we have seen in recent years if computer vision researchers were limited to gathering their own datasets with one-off equipment designed in-house.

That sounds silly, yet this is the situation where we find ourselves in the comparatively young field of haptics. At the moment haptic sensors are more expensive, harder to design, and much more varied than cameras. Accordingly, each research group tends to design its own sensor and then use it to gather a corpus of data for analysis. Sometimes this data is published, such as the LMT database [61], but it is harder to interpret in isolation than visual data. Just as images of the same scene will differ based on ambient brightness and camera settings, haptic data varies due to environmental characteristics (e.g., ambient temperature and humidity) as well as details of the sensor used (this depends on the type of sensor – for example, saturation in the case of a force sensor, or the range and bandwidth of an accelerometer). Qualitatively, haptic data is also harder to share because humans are not used to interpreting static representations of time-varying data (just as one cannot transcribe speech by looking at an audio waveform, it is a tall order to identify a surface by looking at a plot of time-varying contact force – though recent work, such as that of Culbertson et al. [39] makes playback possible with certain kinds of data). Essentially, touch is an active sense, not passive, so it is much less straightforward to create a static encoding of haptic properties. Katz recognized this key difference a century ago:

> The tactual properties of our surroundings do not chatter at us like their colors; they remain mute until we make them speak... Eye movements do not create color the way finger movements create touch. [66]

Even in computer vision, the need for datasets that go beyond static, single-vantage-point images and videos has been recognized [67] – but the need is greater in fields such as haptics. One possible remedy is to share the objects themselves, which prevents the problem of translating data between different haptic sensors. There is considerable cost and logistics involved, but a recent entry in this space is the Yale-Columbia-Berkeley (YCB) object

set [68], which has been largely used for studies on grasp and manipulation planning (but see Regoli et al.'s work on haptic in-hand object recognition [69] and Hollis et al.'s effort to design tactile robotic skin using simulated contact force [70]).

The Proton dataset is one of the first published datasets we know of that contains *matched* (i.e., continuous and aligned over time) haptic, visual, and motion data. This chapter will describe the available data, how it was collected and how to understand the contents. The data and a version of this companion material will also be submitted to an archival journal.

5.1. Description of Data

Our surface database and hardware to create it were designed with two end goals in mind: first, to have a dataset spanning many common material categories and ranges of haptic properties, so that robots of varying haptic awareness could have a wealth of vicarious experience to use in evaluating haptic percepts from their environment; and second, to help develop and test machine learning algorithms to explore the relationship between multiple surface perception modalities. To fulfill these goals, the database will have to contain high-fidelity representations of its surfaces, as well as be easily accessible to researchers so they can integrate it with other systems or federate it with additional datasets. These criteria drove our apparatus design and surface search, and continue to motivate the presentation in this chapter of the data along with ways to parse and visualize it.

Not every mobile robot possesses a dedicated haptic sensor. Of those that do, there are many different types. Some robots may have force sensors as part of grippers, or over a region of "robotic skin," while others could use flexible sensors like the SynTouch BioTac or the newer GelSight. Still others have only visual sensors, or other modalities that can be repurposed to provide some level of haptic sensing (e.g., a microphone or an accelerometer as part of an IMU). This diversity of sensors makes sharing of haptic data between different research groups less valuable than sharing the computer vision data discussed above. In envisioning our visuo-haptic database, we incorporated visual sensors and various types of

haptic sensors in order to collect diverse (yet matched and time-synchronized) data about each surface.

The surface recordings are episodic. For each surface, we continuously recorded data for about one minute with each end-effector, while several sensors were always on regardless of end-effector (see Sec. 5.1.2 on the next page for more details). All data points are timestamped against the same clock, which means all the sensor readings can be played back in correct temporal relation to each other (see [71] for some discussion of the importance of timestamps, though we avoid needing to use their TICSync algorithm by using a single clock). Thus, in every episode we have a baseline measurement of the surface, matched to varying haptic end-effectors. In this chapter I will present the precise procedure we used to collect the data, describe the storage format, and provide sample code for researchers wishing to use the data.

5.1.1. Surface Criteria

A useful database will have a wide variety of surfaces included, measured both by material type (e.g., wood, plastic, metal) and variance across values of relevant haptic properties (such as hardness, roughness, friction coefficient, and thermal conductivity). In addition to this criterion of variety, we constrained our search space in several ways in order to keep the dataset coherent and for practical considerations. We sought flat, texturally homogeneous surfaces which would not damage, or be damaged by, our end-effectors. Surfaces that would not damage the end-effectors were dry and not so rough or sticky that they would abrade or tear off the skin of the BioTac (this being the most delicate of the three end-effectors). On the other hand, we avoided granular materials, like sand, which would change shape after being touched by an end-effector. A few surfaces were not scanned with all three end-effectors because they were either so delicate that the tooling ball would leave tracks, or too abrasive to scan with anything but the tooling ball.

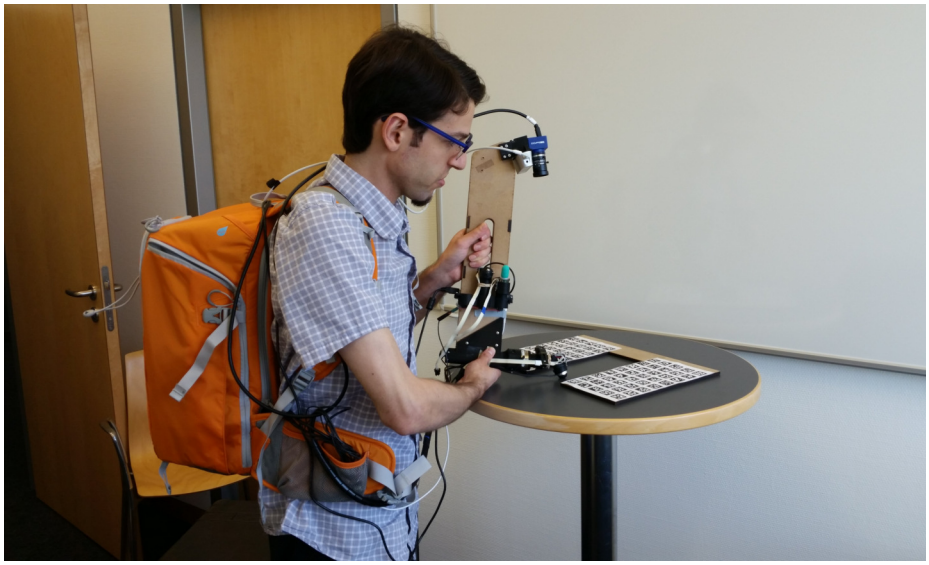


Figure 5.1: An experimenter holds the Proton, equipped with the OptoForce end-effector, in contact with a desk surface (repeated from Fig. 2.4).

Some of the surfaces we scanned were fixed to the environment, such as floors and tables. Others were movable objects or fabric samples. We clamped down movable surfaces during data collection. In the case of thin fabrics or paper-like materials, securing them to a surface means that there was a hard backing, transferring some haptic properties through the surface. This backing may affect the relevance of our measurements of those materials when they are encountered in other contexts (e.g., clothing on a human body or an empty cardboard box).

5.1.2. Collection Procedure

Our database consists of recordings made by several human operators, all with the same instrument, the Proton. The surface episodes were recorded in batches: we performed a series of recordings with one end-effector, and then we went back and scanned each surface again (not always in the same order) with the other end-effectors. We made an effort to ensure that surfaces were observed under adequate lighting in a relatively quiet environment, but given that the surfaces were scanned in different places, inside and outside, over a period

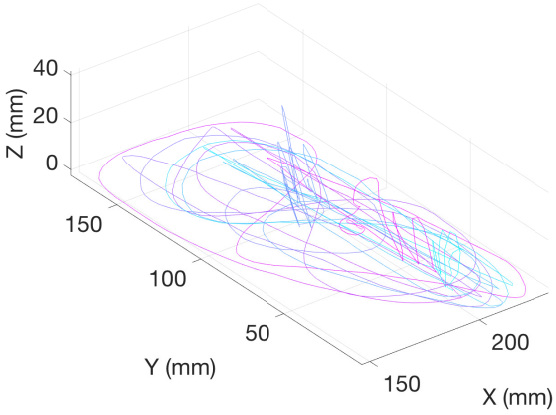
of many months, environmental conditions such as light levels, temperature and humidity were not held constant.

In a typical data collection episode, the operator holds the Proton with two hands, one holding the base and the other holding the handle halfway between the base and the cameras, as shown in Fig. 5.1 on page 90. The fiducial marker frame (see Fig. 4.10 on page 67) is placed on the surface under study (sometimes clamped or manually held down if the surface is not fixed in place). The end-effector tip is brought into contact with the surface and moved back and forth inside the marker frame, in both straight lines and curves, deliberately varying the speed and normal force. The operator will also tap the end-effector on the surface several times with varying impact speeds. Fig. 5.2 on the following page shows the traced 3D position of the end-effector during a typical episode. Some of the sensor data streams from that same episode are shown in Fig. 5.3 on page 93.

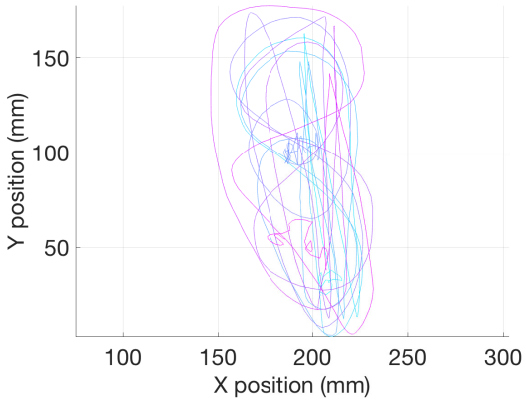
To give an idea of the parameters of data collection, Figs. 5.4 and 5.5 on page 94 show the distribution of normal forces, tangential speeds, and impact forces across the entire dataset. They are separated by end-effector, which is the most significant source of variation (the three end-effectors are made of different materials and therefore have quite different friction characteristics when interfacing with most surfaces). In particular, it is clear that lower impact forces were used with the BioTac end-effector, which is unsurprising as it is a soft, delicate and expensive instrument. Of course, these are averages and we expect the force and speed distributions to vary somewhat between operators, and surfaces (for example, operators will instinctively use slower speeds and lower forces on high-friction surfaces). However, since we measure these parameters, we can use them in analysis, either directly as features or as a normalization factor.

5.1.3. Surface Sources and Locations

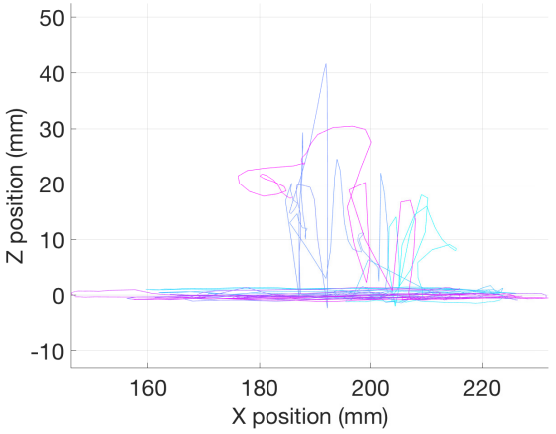
We designed the Proton to be portable and self-contained so that it would be possible to collect data in the field from a variety of sources. Some previous surface datasets are



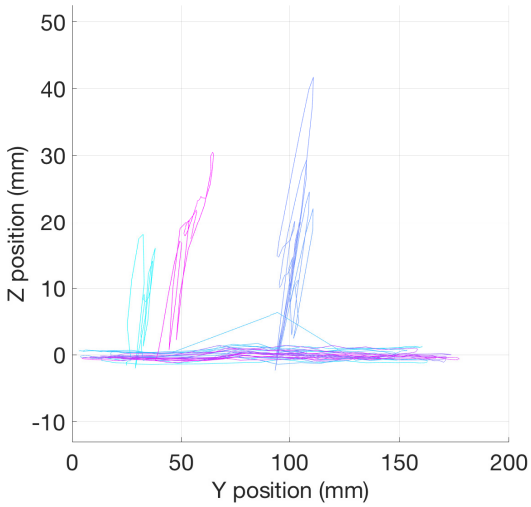
(a) 3D view



(b) Top view

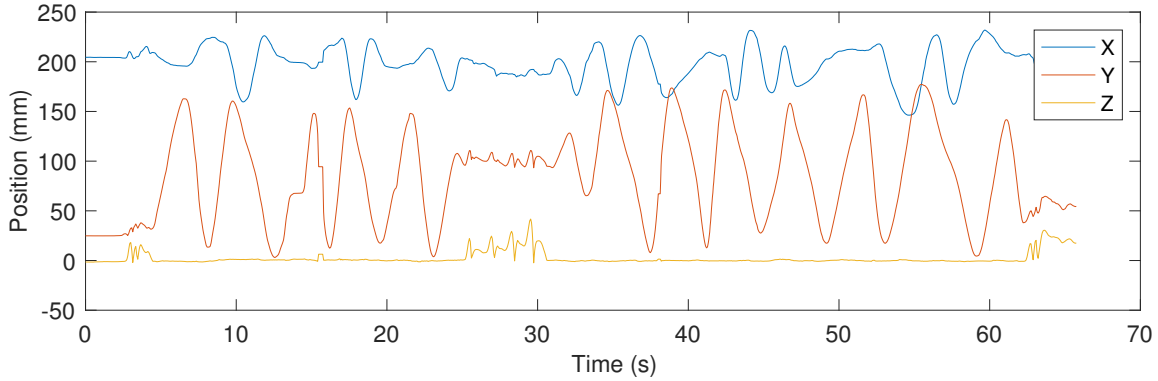


(c) Side view

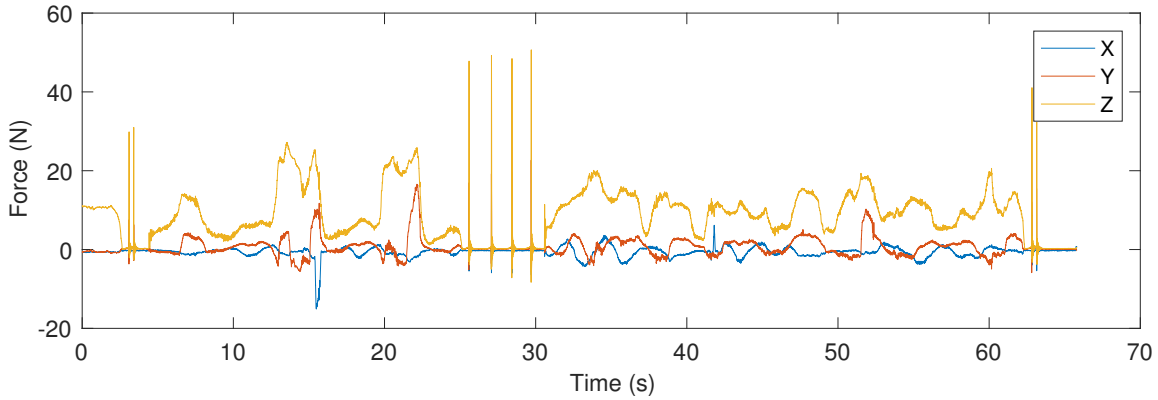


(d) Side view

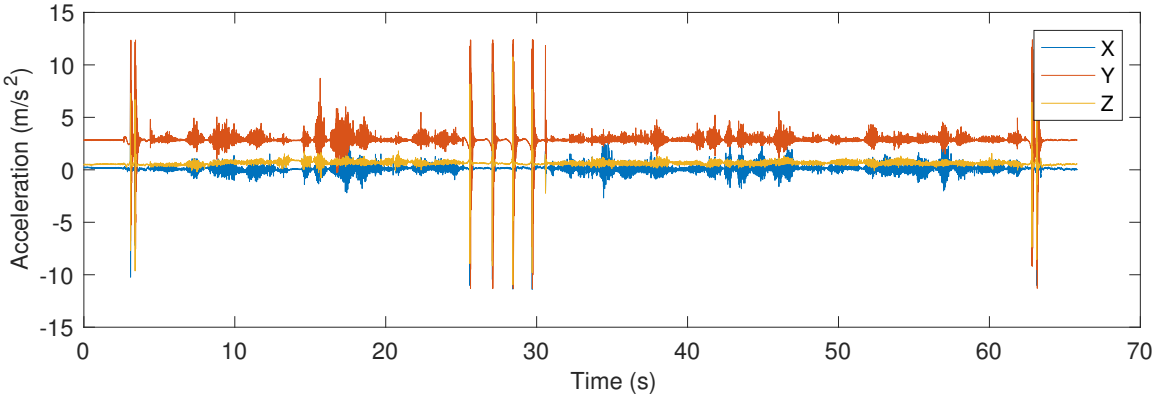
Figure 5.2: Trace of end-effector position during a typical surface recording. The majority of the data is planar dragging, with several tapping motions above the surface. The color gradient shows the progression of time, from start of data collection (cyan) to end (pink).



(a) End-effector position

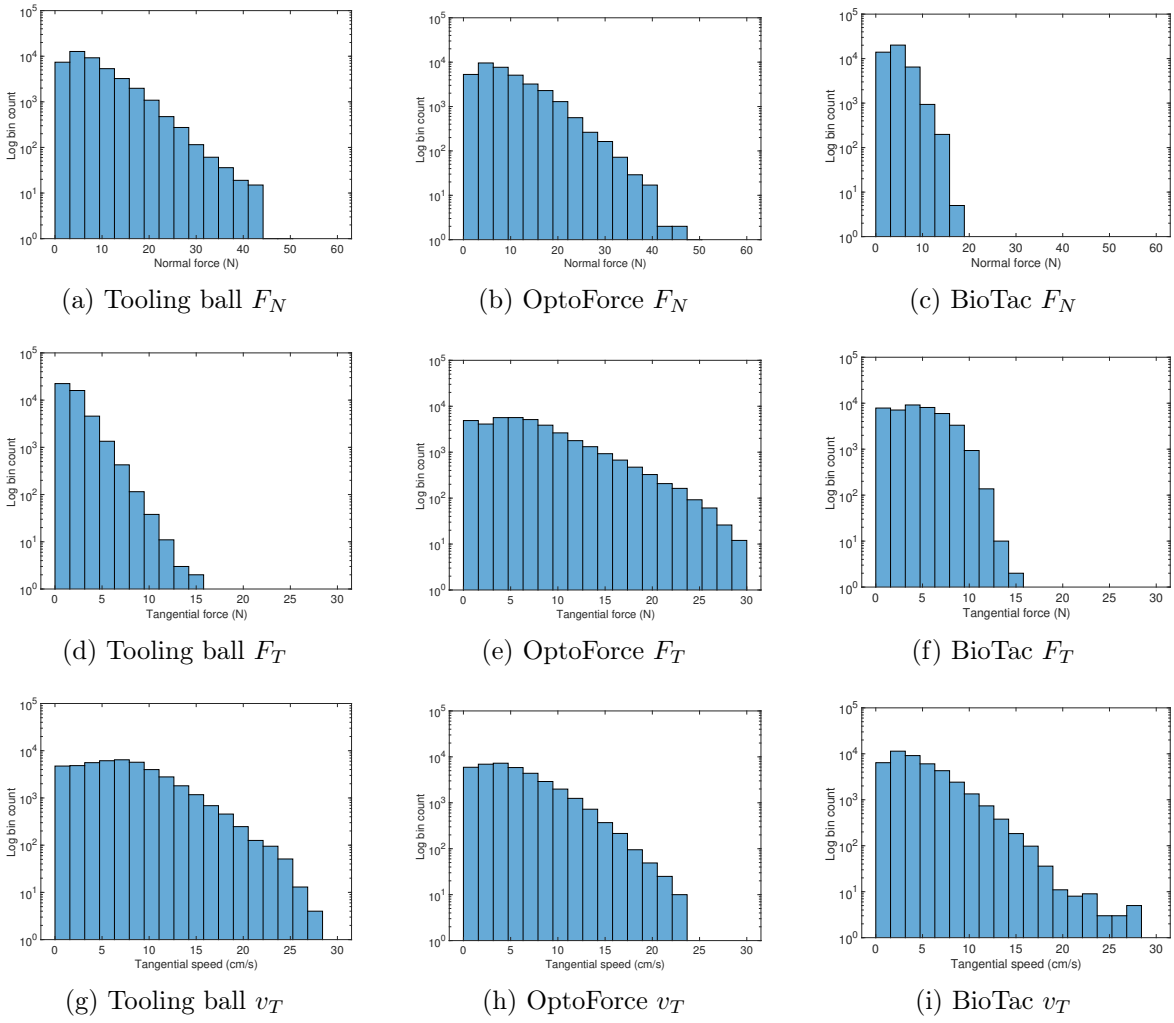


(b) End-effector force



(c) End-effector vibration

Figure 5.3: Data gathered by the (a) onboard motion tracking, (b) ATI Mini40 six-axis force/torque sensor, and (c) high-bandwidth accelerometers during the same typical surface recording shown in Fig. 5.2 on page 92.

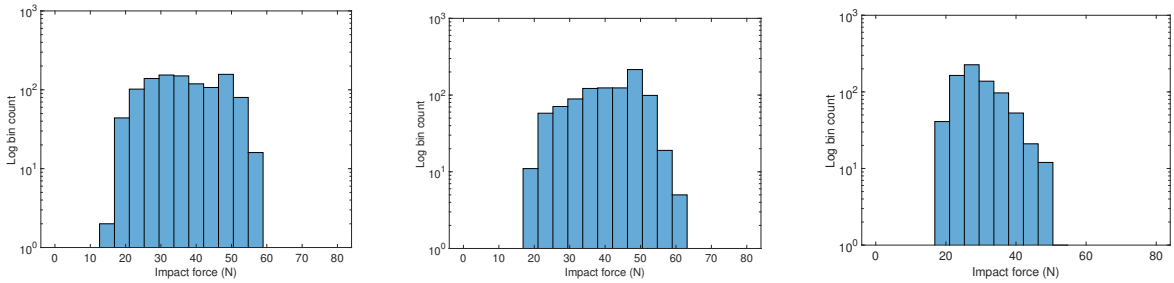


(a) Tooling ball F_N (b) OptoForce F_N (c) BioTac F_N

(d) Tooling ball F_T (e) OptoForce F_T (f) BioTac F_T

(g) Tooling ball v_T (h) OptoForce v_T (i) BioTac v_T

Figure 5.4: Histograms of end-effector force and speed, decomposed into normal and tangential components, over the entire dataset. The first and second rows show the distribution of normal and tangential forces exerted throughout data collection, separated by end-effector, while the second row shows the tangential speed while dragging.



(a) Tooling ball F_I (b) OptoForce F_I (c) BioTac F_I

Figure 5.5: Histograms of end-effector impact force over the entire dataset.

associated with a set of material samples stored in one place, but this can limit the generality of its contents. For maximum diversity of surfaces and data collection conditions, we aim to collect data from surfaces in their natural environment. To be sure, many of the surfaces in our database are portable, but not all. In particular, we have drawn upon the materials in the Penn Haptic Texture Toolkit [48], the LMT material collection [61],[1] the installation of materials from Material ConneXion [72] at the Princeton University engineering library,[2] and a few flat-surfaced objects from the YCB object set [68]. Several other portable materials were purchased from the bookstore at the University of Pennsylvania (Philadelphia, PA, USA) or collected[3] at the Max Planck Institute for Intelligent Systems (Stuttgart, Germany). Finally, substantial data was collected from surfaces fixed in place at the University of Pennsylvania as well as several private homes in the greater Philadelphia area. Further details of the precise locations are given in Appendix B on page 167.

5.1.4. Limitations

There are several limitations inherent in this dataset, mainly related to the amount of time required for data collection, as well as the difficulty of finding a wide variety of surfaces and the difficulty of controlling environmental factors while collecting data with a large number of sensors.

For about half of the surfaces, we do not have data with all three end-effectors. In some cases, the omission was deliberate because the surface was too rough to use with the comparatively delicate OptoForce and BioTac sensors, or conversely, so soft that the steel tooling ball would have damaged it. In other cases, this lack of coverage results from omitting data that was low quality due to poor lighting or sensor malfunction.

We further have some surfaces that are included in the dataset, but missing data from certain sensors because they were not working at various times during data collection: namely the microphone and the Structure Sensor. In particular, the manufacturer-provided Structure

[1]The authors acknowledge Matti Strese for providing on-location access and an extended loan of materials.
[2]The authors also acknowledge Willow Dressel for facilitating access to this collection.
[3]Finally, the authors acknowledge Michaela Wieland for collecting the materials.

Sensor driver was unreliable, such that it often stopped reading frames in the middle of data collection, and interfered with the OptoForce sensor.

5.2. Episode Length Experiment

In order to collect a useful dataset, we sought to pick a suitable interaction duration for each surface, noting the tradeoff between data collection resources and completeness of data. We performed a small-scale pilot experiment, presented in this section, which helped us settle on a duration of approximately one minute per end-effector per surface.

One approach to the duration question frames the data collection process as an online learning problem, building an incremental surface model and using that model to decide when there is enough data. In this experiment we examined how to do such online surface modeling for the initial problem of learning a kinetic friction model. Many different learning objectives are possible, and it's hard to know which will be best representative of the data quality. But kinetic friction is a basic haptic property, and is easy to calculate. With approximately two minutes of data consisting of force, vibration, and speed recorded while continuously dragging the tooling ball end-effector across a flat vinyl surface, we found a good stopping point at 55.4 s.

5.2.1. Friction Modeling

Data collected by the Proton can be used to extract various visual and mechanical surface properties. In this experiment we focus on friction, a basic mechanical property that has been the subject of much modeling effort.

Armstrong-Hélouvry, Dupont and De Wit survey a wide range of friction models [73]. Their review culminates in a combined seven-parameter model, including Coulomb friction, viscous friction, the Stribeck effect, rising static friction, frictional memory, and presliding displacement.

Bilinear model

In this experiment we used a simple model for friction, including Coulomb friction and viscous friction, but neglecting second-order effects for simplicity. The model is given by the following equation:

$$\vec{F}_T(t) = -\Big(\alpha||\vec{F}_N(t)|| + \beta||\vec{v}_T(t)||\Big)\frac{\vec{v}_T(t)}{||\vec{v}_T(t)||} \tag{5.1}$$

The time-varying friction force $\vec{F}_T$ acts in opposition to the tangential velocity $\vec{v}_T$, with a magnitude linearly related to that of the normal force $\vec{F}_N$ and the magnitude of the tangential velocity, where α and β are coefficients fit from experimental data.

5.2.2. Online Learning

Machine learning typically runs over a complete dataset with few constraints on efficiency or processing power required, since the goal is only to learn a model that represents the data well. Typically the model itself is designed to run quickly, but the training process is not (e.g., evaluating vs. training a neural network). However, substantial literature exists in online learning, where a model is updated continuously, often in real time, while the full dataset is not known.

A key question is at what point the learned model can be considered done, and the learning process can stop incorporating new data. One answer is a technique called Stabilizing Predictions [74]. In that work, which considers the problem of word sense disambiguation (a discrete classification task), a small unsupervised test set is reserved from the input data, and learning is considered done when the (unchecked) predictions on the test set have stabilized. In this work, we adapt this technique to a continuous-time domain, fitting a first-order model

to data and inspecting the stability of the model coefficients, which completely determine the model's predictions on unseen data.

5.2.3. Methods

We collected 120 seconds of data while rubbing the steel tooling ball end-effector on a vinyl surface at varying speeds and normal forces. Modalities recorded were force data at 3000 Hz and pose measurements at 15 Hz. Using the processes detailed in previous chapters, we rotated the force measurements into the world frame (relative to the fiducial markers), compensated for contributions from the force of gravity, and decomposed the force and velocity measurements into their respective tangential and normal components, yielding tangential force opposing the direction of motion $F_T(t)$, normal force $F_N(t)$, and end-effector tip speed $v_T(t)$. These data streams were smoothed using a low-pass filter with a cutoff frequency of 15 Hz, then segmented into 100 ms windows (300 samples each). To avoid attempting to fit a kinetic friction model to data reflecting the effect of static friction, windows with an average tip speed of less than 25 mm/s were not considered.

We would like to find the point at which the data encompasses a representative sample of interactions with the vinyl surface, which is a good stopping point for data collection. To analyze progress toward this ideal during the vinyl interaction, we simulated an observation by adding each window in turn to running sets of input data (normal force and tip speed) and output data (tangential force). At each iteration a least-squares fit was used to find the best coefficients of the bilinear model for the friction process (see (5.1)) based on all data observed up to that point. This model was then used to predict the tangential force in the next window.

In theory, the prediction error should decrease to near zero as the model increases in accuracy. However, random noise disrupts the data and makes the simple bilinear model inaccurate on short time scales. Therefore, instead of tracking the incremental prediction error, we looked at the variation of the model coefficients α and β at each window k. This approach

mimics the Stabilizing Predictions method reviewed above [74], which tracks the stability of unsupervised predictions on a test set.

To find a suitable stopping point, we calculated the relative differences between each successive pair of model coefficients to get $\Delta\alpha$ and $\Delta\beta$ (see (5.2) and (5.3)) and smoothed these differences using a moving average filter (window size 10), resulting in $\Delta\alpha^s$ and $\Delta\beta^s$. We set the stopping point to the first iteration at which the average of $\Delta\alpha^s(k)$ and $\Delta\beta^s(k)$ drops below a tunable threshold of 0.001.

$$\Delta\alpha(k) = \frac{|\alpha(k) - \alpha(k-1)|}{\alpha(k-1)} \tag{5.2}$$

$$\Delta\beta(k) = \frac{|\beta(k) - \beta(k-1)|}{\beta(k-1)} \tag{5.3}$$

5.2.4. Results and Discussion

Fig. 5.6 on the following page shows the recorded contact force and speed, which formed the inputs and output to the online learning process. The coefficients of each incremental model are shown in Fig. 5.7 on page 101. We measured the RMS error of the tangential force predictions $\hat{F}_T(t)$ against the ground truth $F_T(t)$, both for a single window w starting at time t_0^w, and for the entire dataset of N samples:

$$e_{inc}(w) = \sqrt{\frac{1}{300}\sum_{i=1}^{300}\left|\hat{F}_T\left(t_0^w + \tfrac{i}{3000}\right) - F_T\left(t_0^w + \tfrac{i}{3000}\right)\right|} \tag{5.4}$$

$$e_{tot} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|\hat{F}_T\left(\tfrac{i}{3000}\right) - F_T\left(\tfrac{i}{3000}\right)\right|} \tag{5.5}$$

The two error metrics are plotted in Fig. 5.8 on page 102. The incremental prediction error (top) represents the performance of each model on the next time window that was not used in its training, while the overall error (bottom) shows each model evaluated against

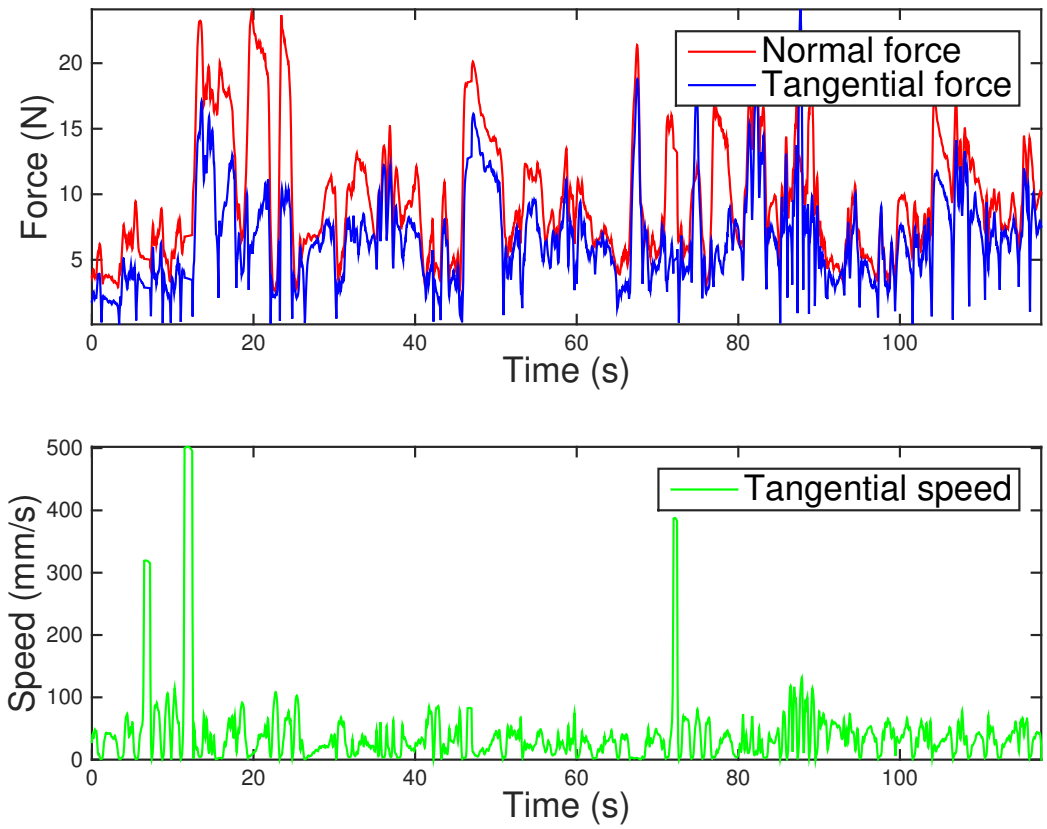


Figure 5.6: Inputs and outputs of the friction model. Top: normal and tangential components of contact force. Bottom: Tangential speed of the end-effector. The friction model takes normal force and tangential speed as inputs and predicts tangential force.

the entire dataset. Application of the coefficient stabilization criterion, as described in the previous section, implies a stopping point after 257 windows, or 55.4 seconds of data. As can be seen in Fig. 5.9 on page 103, the predictions generated by the model at the stopping point are nearly indistinguishable from those generated by the final model.

Strangely, the overall error shown in Fig. 5.8 on page 102 (bottom) trends downward but does not drop significantly during the learning process. This lack of reduction may imply that the data is too noisy for our simple model of friction to account for all effects, or that the choice of 100 ms windows focuses too narrowly, so that the evaluation of model predictions on a single window reflects random noise rather than model accuracy. However, the overall error, which tests the predictions of each incremental model over the entire 120

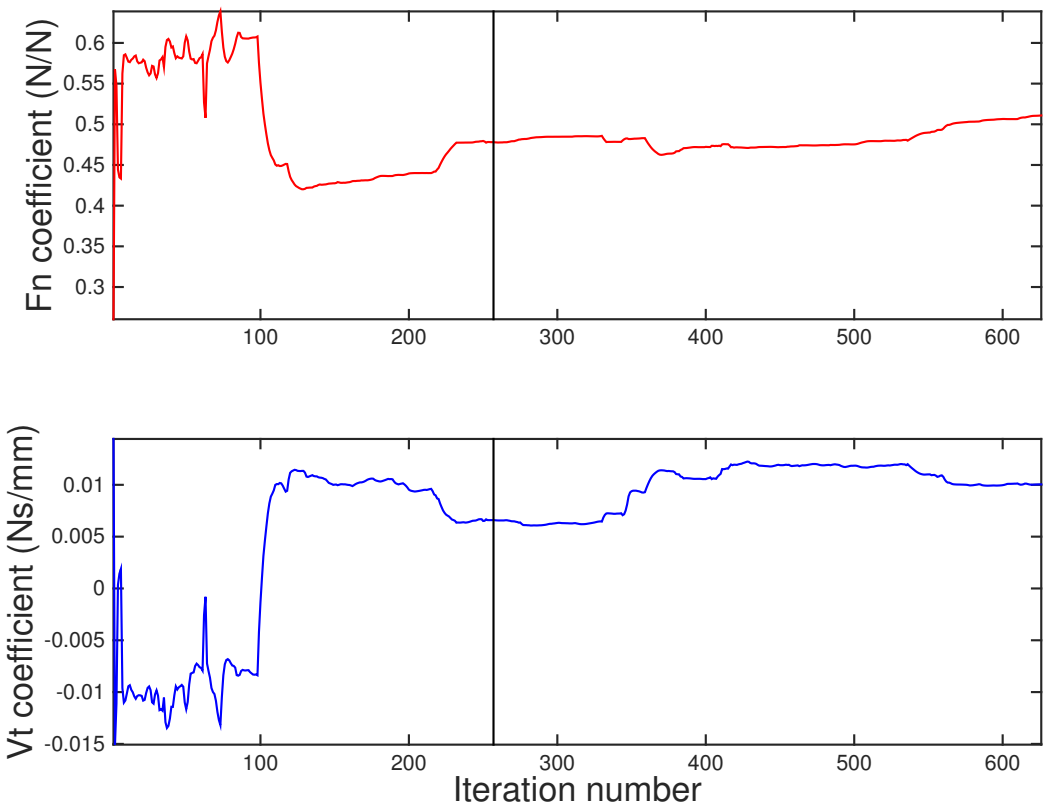


Figure 5.7: Coefficients of each incremental friction model (see (5.1)). Top: α, the normal force coefficient. Bottom: β, the tangential speed coefficient. Vertical lines mark the chosen stopping point.

second dataset (thereby including data that the model was not given for learning), does stabilize at about 1.5 N RMS after initial fluctuations. The point at which the overall error stabilizes is close to the point at which the model coefficients stabilize. This correlation implies that coefficient stabilization can be used as a heuristic, indicating that the overall error (which cannot be evaluated in an online setting) is stabilizing as well.

There is clearly a danger, when using a threshold, of stopping too early (perhaps the incremental prediction error drops very low after a particularly uneventful stretch of time), stopping too late (because the chosen threshold is too conservative), or getting stuck in a local minimum, where the learned model is good enough for the data collected so far but misses possible generalization. Indeed, Fig. 5.7 clearly shows that the best fit value of

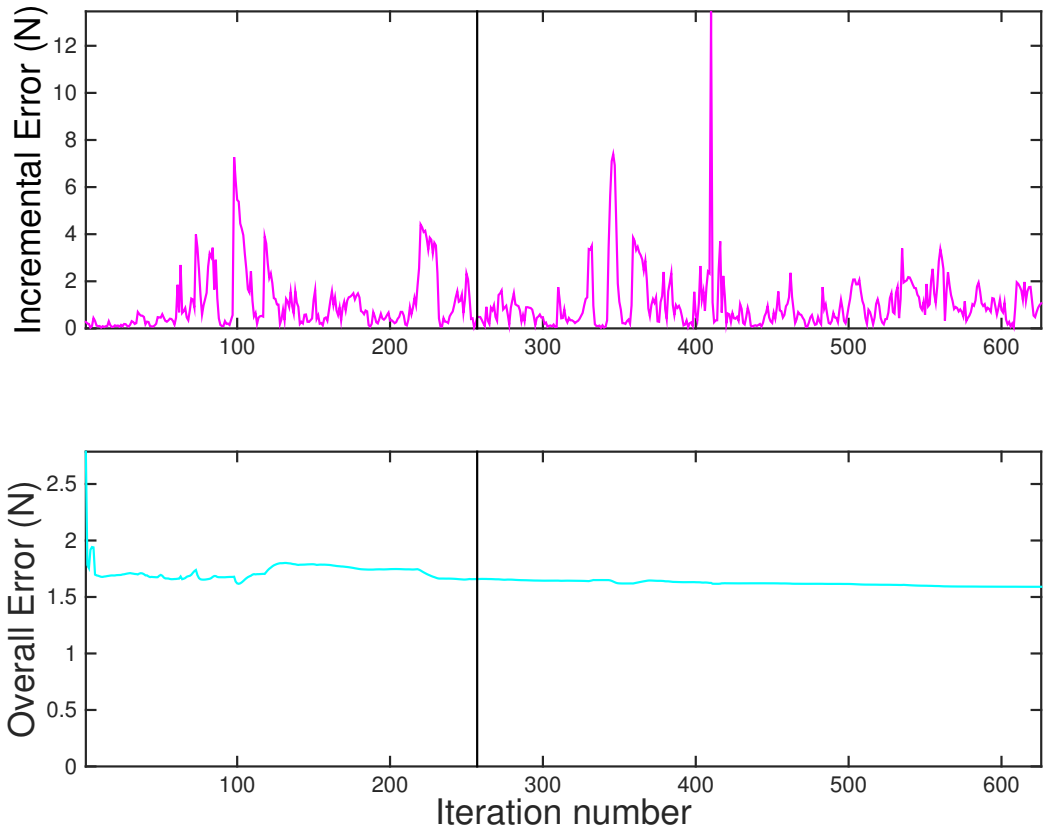


Figure 5.8: Model error during the online learning process. Top: Incremental performance of each model on the next unseen time window. Bottom: Overall performance of each model on the entire dataset. Vertical lines mark the chosen stopping point.

the tangential speed coefficient changed significantly after the stopping point (though the predictions in Fig. 5.9 on the following page appear largely unaffected).

Although the chosen stopping point reflects stabilization of the model coefficients, it is also informative to analyze the comprehensiveness of the distribution of normal forces and speeds. Figs. 5.10 and 5.11 on page 104 show visualizations of the distribution of normal forces and tangential speeds, respectively, after 24 seconds, 60 seconds, and the full 120 seconds. Clearly, the distribution at 60 seconds is of the same shape as the final distribution, mainly different in overall magnitude, but the distribution at 24 seconds has noticeable regions of missing data. If it is possible to guide the data collection process in order to achieve such a distribution earlier in time, we expect the model coefficients to stabilize sooner as well.

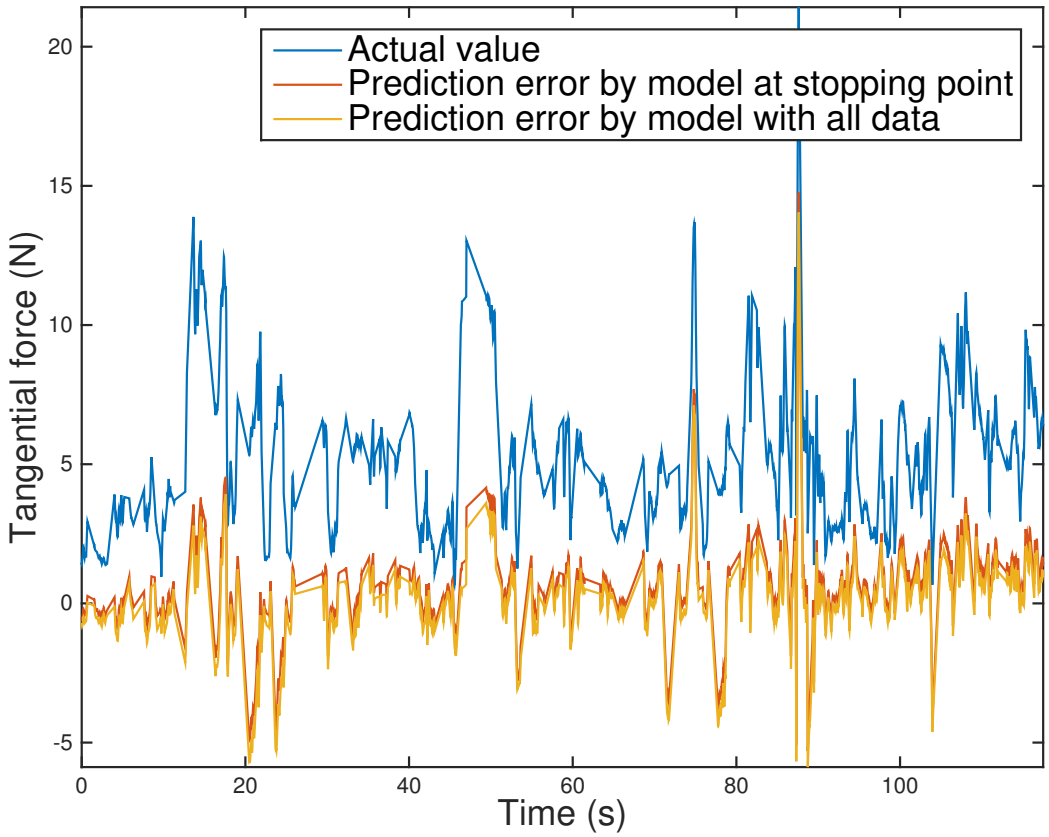


Figure 5.9: Model comparison. The ground truth tangential force is plotted along with the prediction error for a model trained on data up to the chosen stopping point (root-mean-square error = 1.67 N), and one trained on the entire dataset (root-mean-square error = 1.61 N).

Future directions could include hinting the human operator so as to encourage a uniform distribution of applied normal forces and tangential speeds. For the present, this experiment validated our policy of spending approximately one minute (per end-effector) dragging and tapping on each surface.

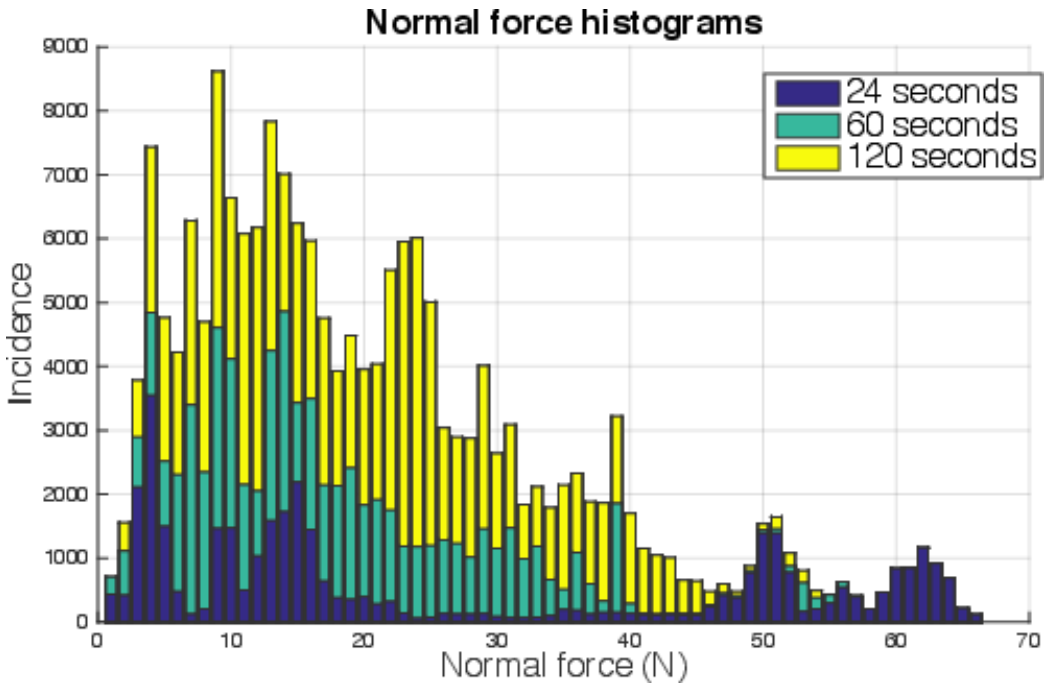


Figure 5.10: Histograms of the distribution of normal force at three points during dataset collection.

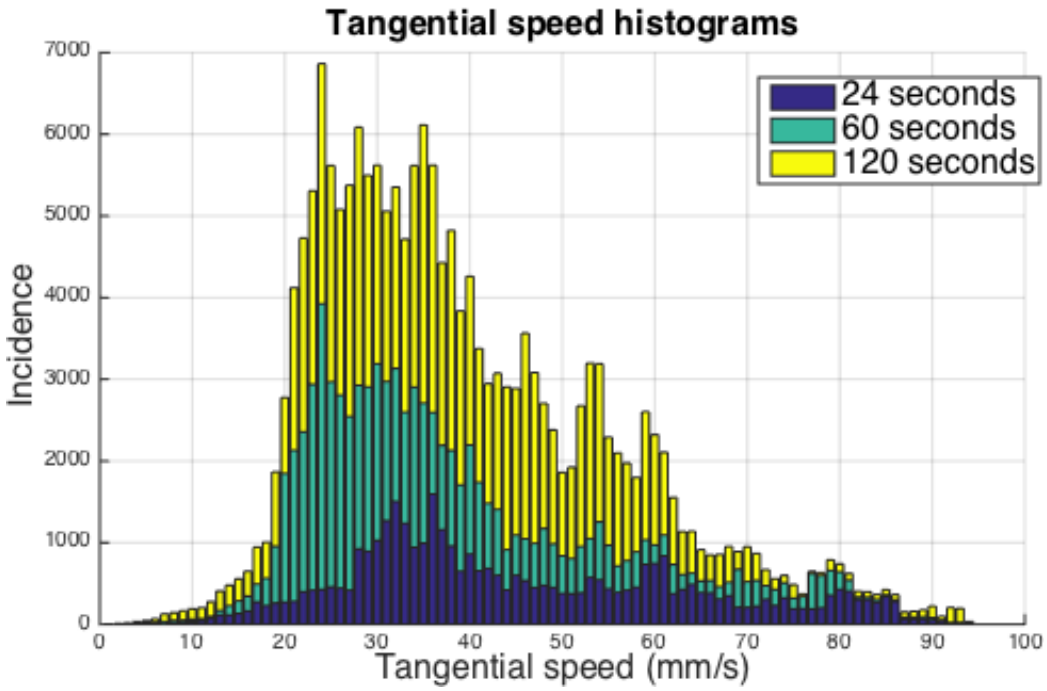


Figure 5.11: Histograms of the distribution of tangential speed at three points during dataset collection.

5.3. Data Format

In the published dataset, the data is separated into "episodes", where an episode is a recording of an interaction between one end-effector and a particular surface. These are organized into subfolders by date of collection and then by end-effector. Thus, a three-tuple (date, end-effector, and numerical index) uniquely identifies an episode.

In each episode directory there are CSV files containing the processed output of each activated sensor, as well as folders containing PNG image frames from the Bluefox and Structure (if activated[4]) cameras. Additionally, a folder containing cropped surface images, selected using the procedure described in Sec. 6.2 on page 113, is provided. Finally, there is a "flow" file which provides timestamps for each step of the data collection procedure, as well as the experimenter's answers to questions. These questions include the surface name and perceived stickiness (for all end-effectors), plus the perceived hardness, roughness, and temperature for the BioTac end-effector.

The formats of the individual CSV files have been detailed in Sec. 3.1 on page 30.

5.3.1. Visualization

Visualizing results is important both during and after the data collection process. We provide two main visualization tools. The first is a MATLAB graphical interface for navigating the multiple sensor streams simultaneously, while the second synthesizes video with motion and vibration data.

Episode Visualizer GUI

We will provide a MATLAB GUI visualizer with the surface interaction dataset. Sarah Allen made significant contributions to the code. The GUI, shown in Fig. 5.12 on the following page, displays the video recorded by the Bluefox camera as well as haptic data streams (force, acceleration, and OptoForce or BioTac data if available) in synchronized

[4]During some of our data collection, the Structure camera was turned off due to driver issues.

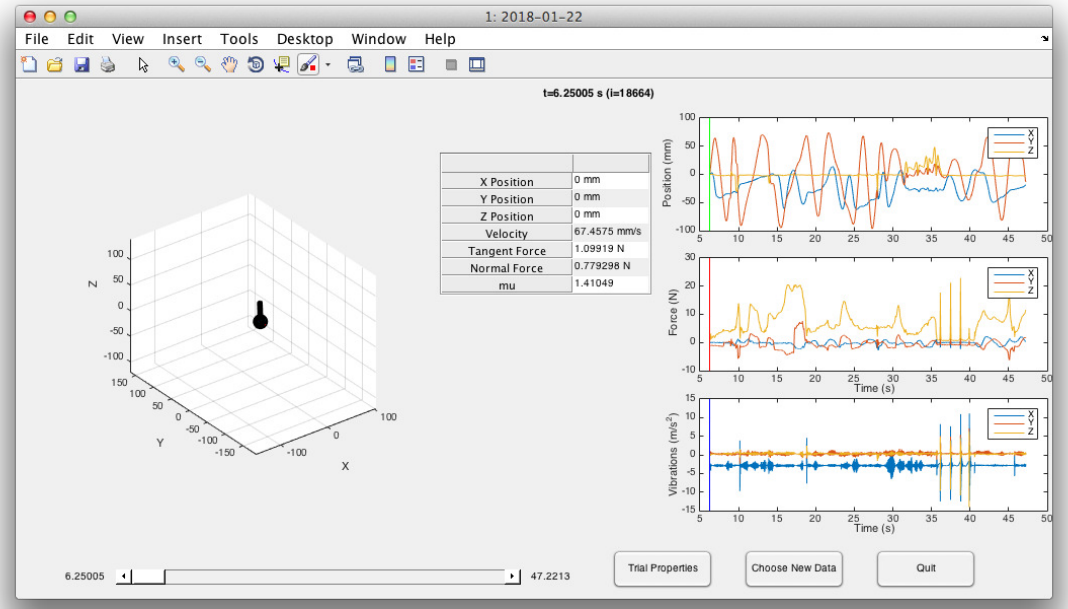


Figure 5.12: MATLAB visualizer analyzing a dataset episode in which the tooling ball end-effector was tapped and dragged on a velvet surface.

plots. Vertical cursors (respectively green, red, and blue vertical lines in the figure) show the time position in the data plots. The user can advance the display at various playback speeds or seek to specific points using a scrollbar.

Video Renderer

Because the surfaces in our database are flat, it is possible to use the motion data to retrace the path of the end-effector. To turn a surface interaction episode into a movie, we first project the end-effector position into the plane of the surface and plot that point in every frame, as well as the preceding end-effector positions to trace a path. An example frame is shown in Fig. 5.13. We further add an audio track derived from the vibration measurements (mean-centered and combined to a single dimension using the DFT321 algorithm [47]).

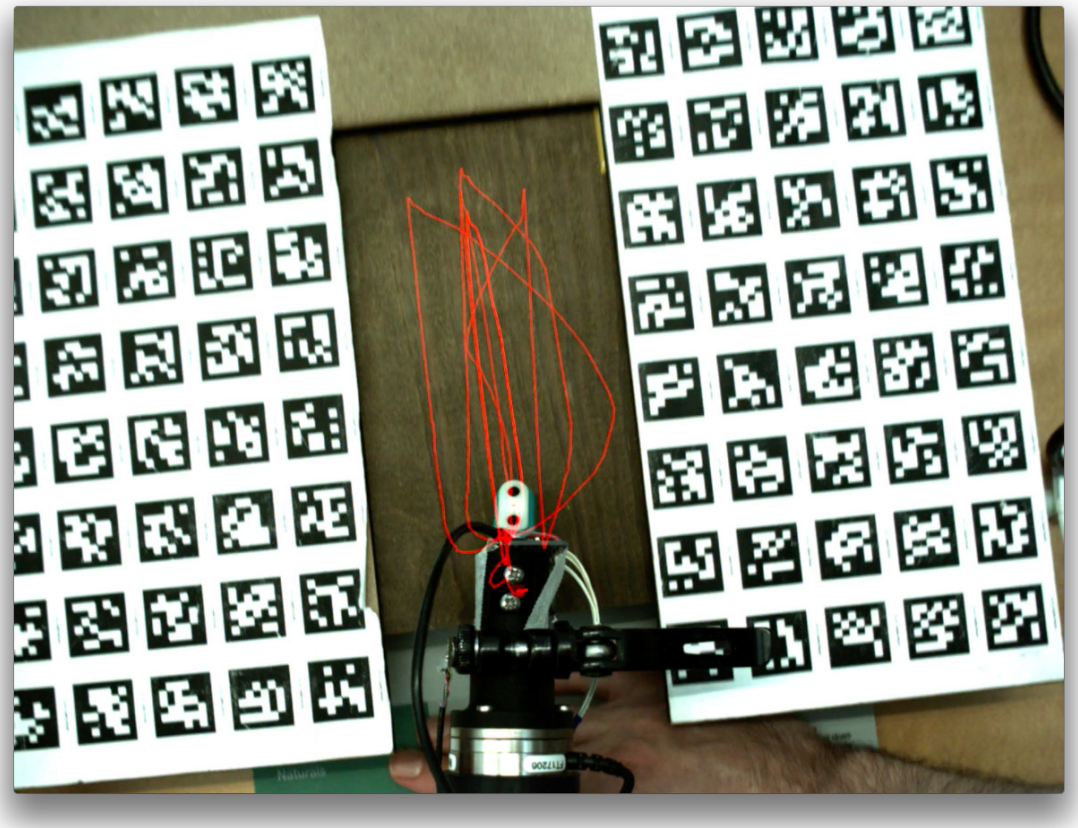


Figure 5.13: Example frame of a video processed from interaction between the BioTac end-effector and a pinewood surface.

5.3.2. Usage Instructions

In our work, all processing and visualization of the data has been done using MATLAB versions R2014b and R2017b. Several loading and processing scripts are provided with the data as examples. The code shown in Listing 5.1 on the next page loads all episodes collected with the OptoForce end-effector on April 6, 2018, applies the appropriate weight calibration and rotates all data into the world frame. Then it shows the cropped pictures of the surface, as well as plotting the data from each sensor. See the documentation of each MATLAB function for more information.

```
[data, materials, calib] = proton_load_and_process(...
    'proton_data', ... % data dir
    '20180406',    ... % collection date
    'optoforce');      % end-effector
proton_plots(data(materials{1}), ...
             {'crops', ... % cropped surface images
              'pose', 'force', 'accel', 'sound', ...
                  % always-on sensors
              'opto'});
                  % sensor specific to this end-effector
```

Listing 5.1: Example code for loading episodes and plotting some of the data, in MATLAB.

5.4. Summary

We believe that the field of haptics needs to join other research fields in the use of shared datasets. These provide the means for measuring new advances against a common benchmark, as well as supplying researchers with more comprehensive and versatile datasets than they could gather alone with idiosyncratic instruments.

In this chapter we presented one of the major contributions of this thesis, a database of matched visuo-haptic recordings of several hundred surfaces. The database will be made publicly accessible so that it may be used for many different applications and federated with other, similar databases. The next chapter will describe our first application of the dataset, a cross-modal machine learning algorithm.

CHAPTER 6 : VISUO-HAPTIC MACHINE LEARNING

Prediction of haptic properties is useful in tasks such as navigation and object manipulation. When walking or driving on different materials, the surface's shape and friction characteristics constrain the parameter space of viable speeds, accelerations, and turning radii. When transitioning from, e.g., asphalt to ice at driving speed it is highly advantageous to anticipate these properties before making physical contact. Similarly, a robot with a task that involves handling objects of different materials, such as cooking a meal, will need to adjust grip force, motion speed, and other control parameters to account for properties such as weight, hardness, and slipperiness.

Humans tend to have an intuition that visuo-haptic property estimation should be easy, since most of us do it every day, overcoming the many difficulties inherent in interpreting images (e.g., occlusion, variable lighting, visual patterns obfuscating material texture). We can generally recognize material classes despite varying viewing conditions and misleading color cues. For example, plastic is often decorated to look like wood, but specular reflections can give it away.

This chapter presents a human-subject study to test this intuition and provide a benchmark for the task difficulty. We then describe progress toward solving the problem with a machine learning approach. We use cropped images from our dataset to predict the quantitative haptic properties studied in Sec. 4.6 on page 72, using a model that is first trained to classify the images into broad material types, then cross-trained to perform regression.

Figure 6.1: An illustration of "Moravec's paradox": often, tasks that seem easy and intuitive to humans (such as perception) are quite difficult for computers to learn, even though they excel at reasoning that challenges humans (such as chess and Go [75]). Comic from XKCD [76], licensed under CC BY-NC 2.5.

6.1. Background

In 1966, an MIT CSAIL researcher named Seymour Papert (not Marvin Minsky, as is popularly believed [76, 77]) organized a group of students to work on simple computer vision problems for a summer [78], including figure/ground segmentation and geometric object recognition. There have been many advances since then, though perhaps not quite as fast as those early AI researchers anticipated (see Fig. 6.1).

Computer vision is used to solve many problems that are relevant to robotics, such as *visual servoing*, which refers to controlling actuators or motion based on visual input, *localization*, using visual clues to determine a robot's position on a map which is known or under con-

struction (in the latter case, the problem is referred to as SLAM), and *object recognition*, identifying and describing the elements of a scene.

A common technique underlying many of these tasks is *feature engineering*: defining small recognizable regions of the input images, and ways to describe them, that generalize across different images of the same object, or different objects of similar types, in order to allow tracking and identification over a large dataset. A feature might be a small image (such as a letter) that is convolved over the input to find similar regions, or a distinctive corner (there are many popular corner detection algorithms, such as FAST [79]), or something else chosen by a heuristic algorithm such as SIFT [80] or SURF [81]. In localization, a technique called structure from motion (SfM) tracks feature correspondences from one image to the next, and a 3D transformation can be inferred from their relative motion.

For object classification or description, feature engineering is also an integral part of the traditional approach. Given a large dataset of input images and desired outputs, features common to the different categories of input can be identified. In one approach, known as *bag of words* because it was adapted from the field of natural language processing (NLP), the relative number of different types of features present in an image, regardless of their geometric arrangement, can serve as the signature of an object class. The field of computer vision is too large to adequately survey here. Modern advances can classify thousands of objects, driven by annual competitions and crowd-sourced datasets such as ImageNet [64] (which, incidentally, also owes its structure to the field of NLP).

However, as tasks proliferate and datasets grow, feature engineering starts to lose its appeal. How can we be sure that the features we design are general enough to cover realistic image datasets, yet specific enough to be germane to the task at hand? Human intuition is not easily dissected – one cannot generally explain the instantaneous reasoning that leads to recognizing an image of, say, a cat – so it will always be a difficult engineering problem to come up with such rules for a computer. Enter automatic representation learning, specifically deep neural networks.

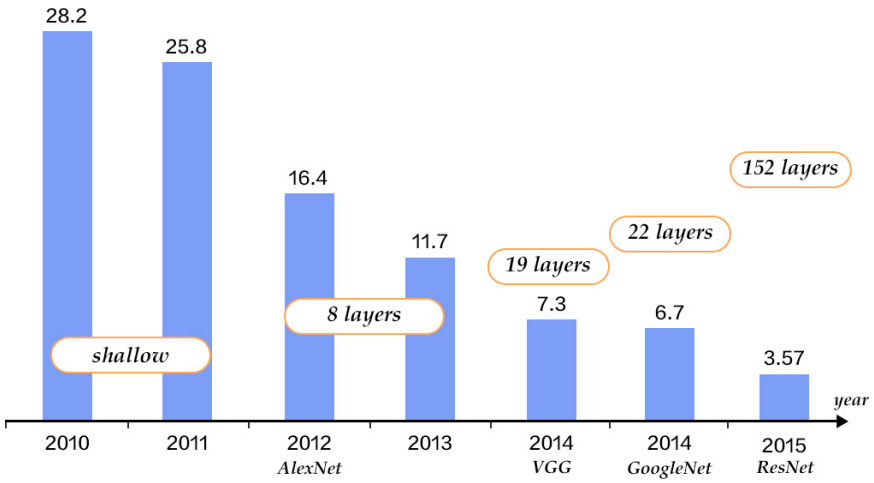


Figure 6.2: Top-5 error rates on ILSVRC image classification, showing the 2012 deep net breakthrough. For reference, human performance on the classification task is in between GoogLeNet and ResNet. Figure reproduced from [82], licensed under CC BY-SA 4.0.

In 2012, Krizhevsky et al. [65] shocked the computer vision world with their submission to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which revived the dormant technology of deep convolutional neural networks (CNNs) and catapulted it to the forefront of machine learning (where it has remained ever since). Fig. 6.2 shows the progress since the competition was started in 2010, as well as the depth of neural network solutions that were submitted. AlexNet is generally considered a turning point in computer vision research. Since then, neural networks of increasing complexity and varying structure have dominated the cutting edge of computer vision research – taking a substantial portion of the graphics processing unit (GPU) market away from video games.

Neural network learning has been applied to surface image datasets. Bell et al. [83] collected a dataset of images of various materials, both crowd-sourced from Flickr and hand-labeled from an interior design company. That dataset was mainly used for material classification using retrained versions of AlexNet [65], VGG [84], and GoogLeNet [85]. Gao et al. [46] extended the BOLT project's haptic adjective classification work [43] by fine-tuning the Materials In Context (MINC) networks using images of the sixty BOLT objects and then added haptic data as an input, achieving higher classification accuracy. More recently, Yuan

et al. [86] trained CNNs jointly on visual and haptic data in order to classify images of fabrics based on haptic data, and vice versa.

To our knowledge, this thesis is one of the first to focus on a regression task for material property perception, rather than classification. Though the most common application for networks is classification, in this work we adapt a network designed for general image classification to be able to estimate haptic properties from visual input. After training the network to classify materials into categories, we extract layer activations as features and use them to train SVMs.

6.2. Task Definition

Our main goal is to infer haptic properties from visual data. Therefore, the input is a close-up image, or several images, of a surface, and the output is one of a set of quantitative haptic properties or a numerical rating on an adjective scale. In this work, we used the same scan-free features studied in Sec. 4.6.3, since they are a convenient set of haptically relevant properties which can describe recordings of various lengths, abstracting over the scan-time parameters of individual recordings. For the human-subject study, we asked participants to look at images and provide ratings on numerical (1-5) scales of hardness, roughness, stickiness, warmness. These are the same ratings that the Proton Pack operator made while collecting the data. We also tried to correlate those human ratings with the haptic properties. In our subsequent machine learning work, we treat the haptic properties and numerical ratings as separate outputs.

Following Secs. 4.3 and 4.6.3, we use both scan-dependent features (binned vibration frequency and friction coefficient) and scan-free features (melfrequency cepstral coefficients, hardness, spectral centroid, temporal roughness, waviness, spikiness, fineness, regularity, friction, and normalized friction) as regression targets in our machine learning work. It is unclear how closely these scan-free features are connected to qualitative surface descriptions such as “hard” or “slippery” – see the next section for some investigation.

We used similar procedures to prepare data for the human-subject and machine learning studies. For each surface, we select five freeze-frames from the recorded video with a clear view of the surface, and cropped the view to emphasize the surface. There are several steps to this extraction process. First, the motion track of the end-effector through the entire episode is calculated, and its convex hull is used as a conservative estimate of the extent of the surface under study. Next, since the end-effector partially blocks the camera view, the overlap of the end-effector (which is always at a fixed place in the camera frame) and the convex hull is calculated. We choose the five frames with the smallest overlap (and therefore the clearest view of the surface) with the constraint that selected frames should be at least one second apart. Finally the image is cropped to include the convex hull plus a small buffer radius.

6.3. Human-Subject Baseline Study

Despite its ubiquity, we are not aware of a quantitative benchmark for visuo-haptic inference, which is desirable when designing a machine learning solution. This section details work toward quantitatively assessing human performance at inferring haptic properties from images. We presented human users with images of materials such as those shown in Fig. 6.3 on the next page. We then compared their haptic ratings with those made by the Proton operator and with physical properties calculated from haptic sensor data.

6.3.1. Study Design

We ran a human-subject study, approved by the Penn Institutional Review Board (IRB) under protocol #828525. The survey was conducted through Amazon Mechanical Turk, an online survey platform. After completing a consent form, participants were presented with one characteristic surface image at a time, and asked to rate it on several dimensions that were chosen to represent the main dimensions of tactile perception [87]. The interface is shown in Fig. 6.4 on the following page. The first question, about the shape of the image, is purely an attention check – responses with the wrong answer to this question were

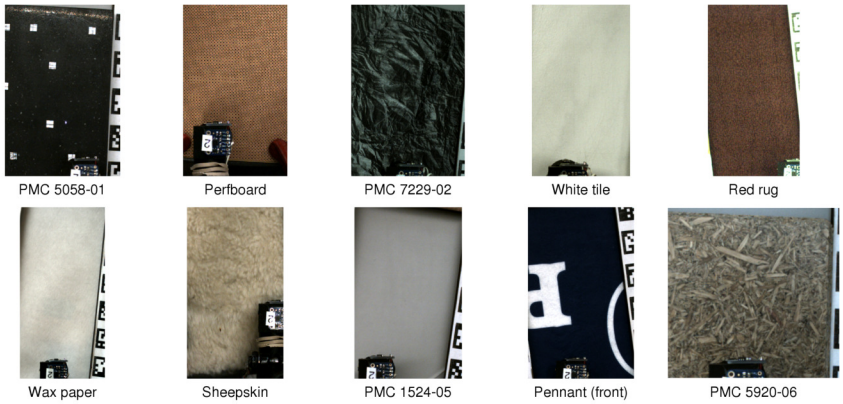


Figure 6.3: Some surface images in the human-subject study dataset.

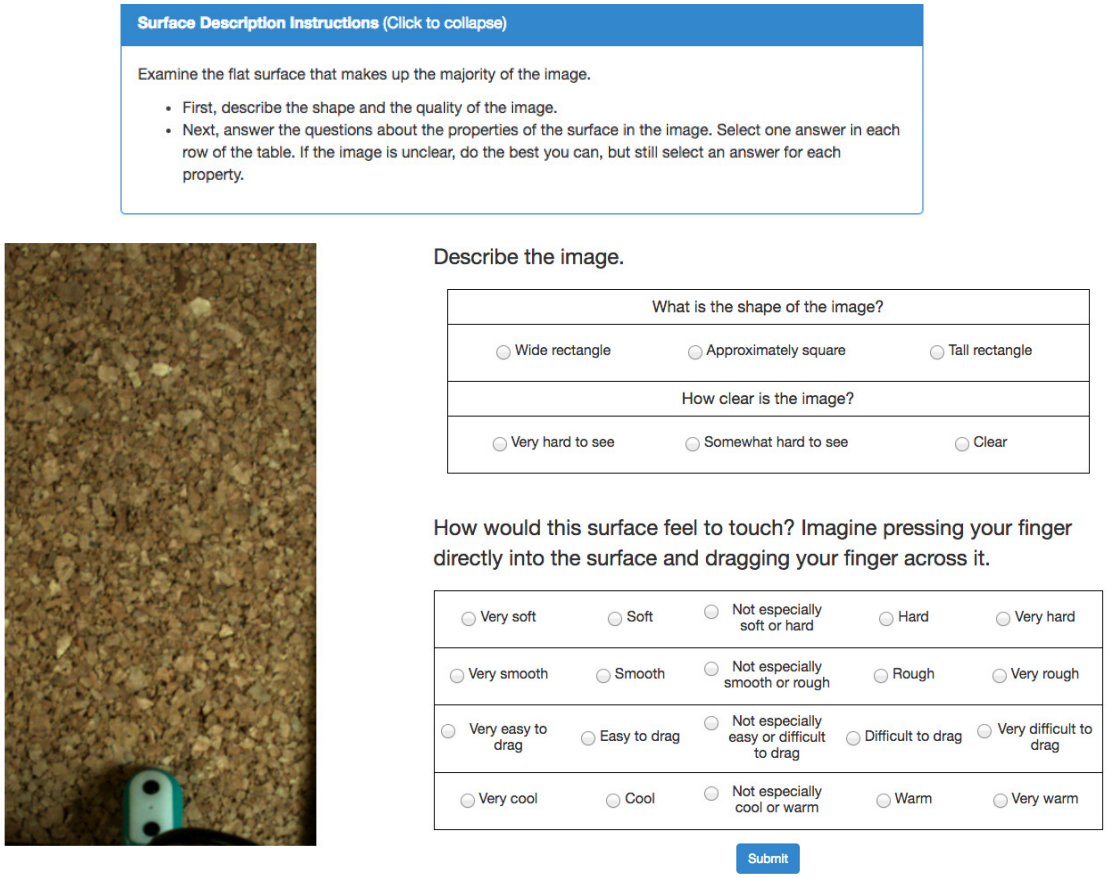


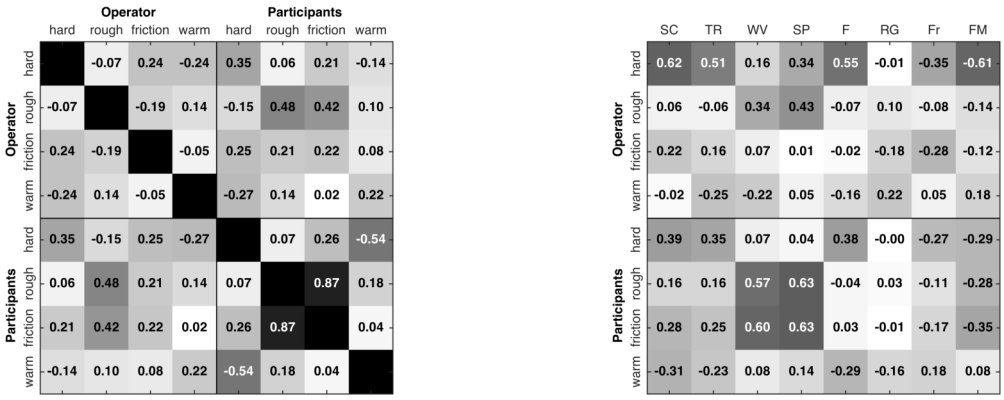
Figure 6.4: MTurk participant interface, showing a cork surface.

discarded. The second question, about image quality, helped us prune our dataset; responses specifying a poor-quality image were discarded as well (later, this assessment was done by the author for surfaces not included in the human-subject study). Finally, participants rated the depicted material on the four axes of hardness, roughness, friction, and warmness, using 5-point scales (see the figure for precise phrasing). Participants could rate as many images as they chose and were compensated $0.04 for each response; assuming each response takes approximately 18 seconds, this pay rate is equivalent to $8.00/hr, slightly more than local minimum wage. From the full set of 1127 images for the first 100 surfaces in our dataset, images were shown in a random order such that five distinct participants rated each image.

6.3.2. Data Cleaning

To regulate the response quality without being overly restrictive, in some cases multiple answers were accepted for the attention check question: "wide rectangle" was accepted when the image was wider than it was tall, "tall rectangle" otherwise, and "approximately square" if the ratio of width to height was between $3:4$ and $4:3$. After excluding 721 responses containing incorrect answers to this question and 138 responses indicating low quality images, we excluded images rated as low quality by a majority of the participants who saw them. If this criteria caused the removal of a majority of images of a surface, we excluded the surface entirely. In sum, this cleanup excluded 978 responses, 40 images, and 2 surfaces, leaving 4657 responses to 1087 images of 98 surfaces.

Finally, the human ratings and quantitative properties were combined across different viewings of the same surface. The operator rated the friction for each end-effector (since they have different frictional characteristics), so we used the median of all three ratings. Similarly, for each dimension we used the median of ratings given by all study participants to all views of a particular surface. Lastly, we used the scan-free features generated by the data from the metal tooling ball end-effector, because it generates the clearest vibration signals due to its rigidity.



(a) Operator vs. participants

(b) All ratings vs. properties

Figure 6.5: Pearson correlation coefficients among operator ratings, participant ratings, and scan-free features. Shading reflects the absolute value from 0 (white) to 1 (black).

6.3.3. Results

These methods yielded quantitative physical data about the surface from our haptic sensors (mainly motion, force, acceleration, and temperature over time), as well as human ratings from the operator and the study participants. Incorporating the cleanup and processing from Sec. 6.3.2, Fig. 6.5 shows the study responses compared with both the operator's ratings (left) and the quantitative properties (right).

Operator vs. Participants

Fig. 6.5a shows a Pearson correlation matrix of the experimenter and participant ratings on each dimension. The cross-correlations between the operator's ratings and the participants' ratings on the same dimensions were 0.36 for hardness, 0.48 for roughness, and 0.22 for stickiness and warmness. These values are not especially high, and in general the participant ratings are more correlated across dimensions than the operator's ratings. It is difficult to disentangle two effects that might cause this result. First, the judgments are simply more difficult for the participants, who never touched the surfaces nor saw them under various lighting conditions, as the operator did. Second, we did not include specific instructions that

the rating dimensions should be orthogonal or the precise differences between, for example, roughness and friction, whereas the operator was familiar with haptic sensing literature and strove to give consistent, orthogonal ratings.

Despite those issues, the correlations are encouraging and invite further exploration. The relationship between the different dimensions is similar between the operator and participants in some cases. For example, both exhibit the pattern that there is little correlation between hardness and roughness or between friction and warmness, yet there is a substantial negative correlation between hardness and warmness. This pattern may arise from the fact that softer surfaces tend to be fabrics, which naturally are low in heat conductivity and often visually associated with warmth (e.g., blankets). Previous studies such as Baumgartner et al. [88] have found similarly high correlations between ratings of images, including roughness-friction and hardness-warmness.

Ratings vs. Properties

We attempted to use quantitative physical properties of the surfaces, calculated from our motion, force, and vibration data, to explain the operator and participant ratings. Fig. 6.5b on page 117 shows the correlations between the ratings and properties.

We observe that more work needs to be done to craft quantitative properties explaining human ratings; however, some high correlations are to be noted. In particular, we find in agreement with Strese et al. [5] that high values of spectral centroid are associated with hardness, which makes sense because this property measures ringing after the end-effector taps on the surface. Meanwhile, waviness and spikiness, two of Strese et al.'s "macroscopic roughness features," are indeed associated with roughness (according to our participant data, they are associated with friction as well, but as discussed above, roughness and friction are highly correlated in the participant ratings).

6.3.4. Impact

Our results here have shown that humans can estimate certain haptic properties from images, though there is more of a tendency for property values to be correlated across dimensions when haptic information is not available. The human-subject study could be extended with more surfaces and perhaps with different wording to try to further distinguish surface properties (assuming that study participant confusion contributed to some part of the observed inter-rating correlation). The next section will detail our efforts to bring machine learning to bear on a similar task.

6.4. Machine Learning

Having established a baseline level of human performance on the task, we set out to train a machine learning algorithm to predict haptic surface properties from visual input.

After obtaining image crops using the procedure described in Sec. 5.1.2 on page 90, we specifically prepare them as a CNN training set by transforming them to square 227x227-pixel images. Instead of directly resizing the rectangular images, which might result in distorting the very textural features that we are attempting to discover, we crop to a square region and resize, thus preserving the aspect ratio. Most of the images are taller than they are wide, so to avoid throwing away valuable pixels we extract two square crops from such images: one from the top and one from the bottom of the original image.

For some exploration of the haptic properties used as learning targets, see Figs. 6.6 and 6.7 on page 121 and on page 122. Fig. 6.6 shows the calculated properties for several materials, selected to have a range of surface characteristics. Visually, it is clear that properties of stone tile and concrete floor (nearly the same material) are quite similar, while still bearing a passing resemblance to aluminum (another hard surface), but quite different from styrofoam and the couches. We can zoom out from this view of individual surfaces by looking at the MINC categories. In our dataset there are eight of those categories represented by at least thirty surfaces. The distributions of scan-free features in each category are plotted

in Fig. 6.7. These plots are noisier, but it is clear that different categories have different patterns of properties. For example, the spectral centroid is lower for fabrics than other types of materials, while waviness tends to be highest for carpets, metals, and tile.

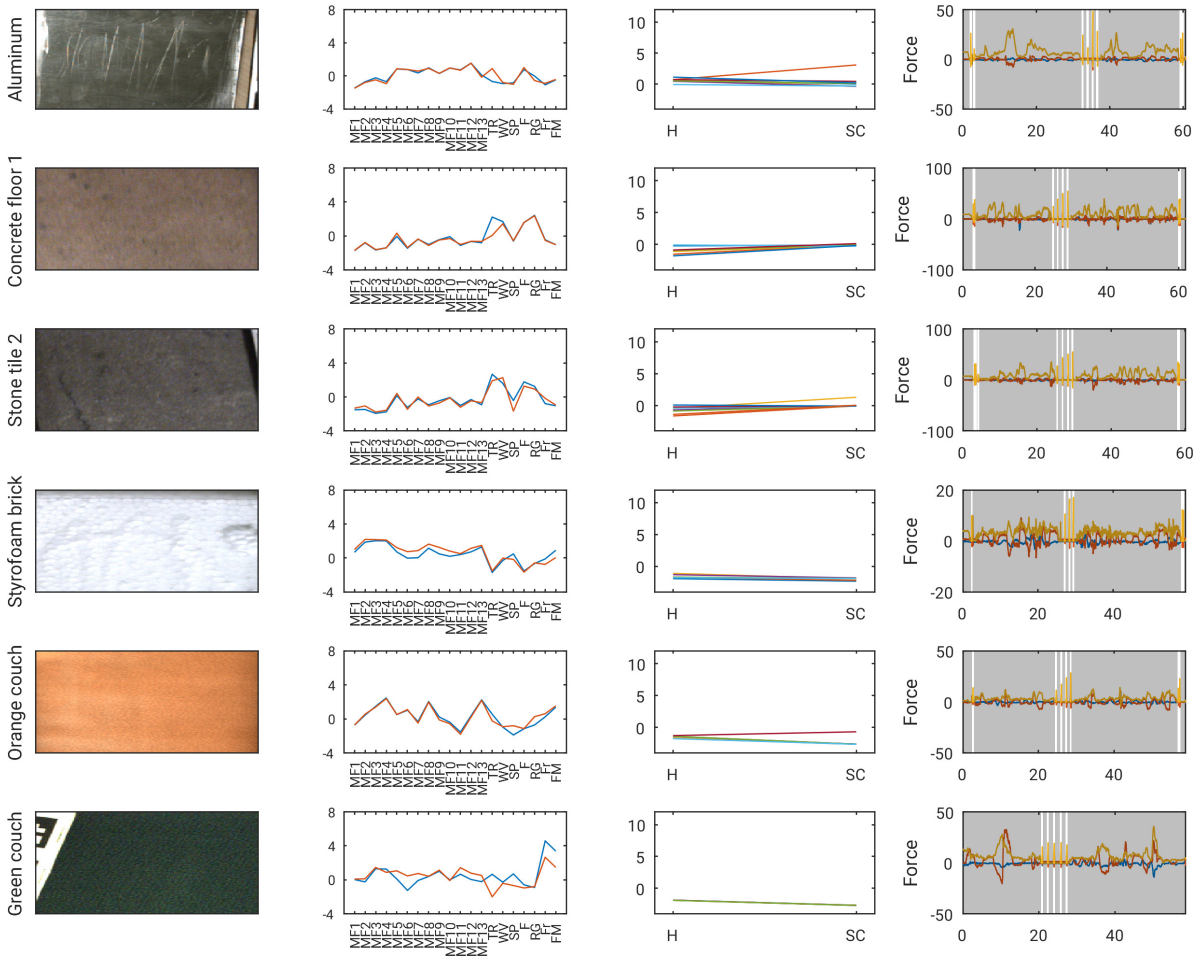


Figure 6.6: A selection of surfaces with their scan-free features shown graphically. The left column contains representative cropped images of the surfaces, while the rightmost column shows the contact force recorded by the ATI Mini40 six-axis force/torque sensor, transformed into the world frame (F_x, F_y, F_z color-coded red, blue, and yellow respectively), with the automatically segmented taps shaded white and the dragging periods shaded gray. In the middle are plots of the dragging-based scan-free features for each distinct dragging period (middle left), and the tapping-based scan-free features for each tap (middle right).

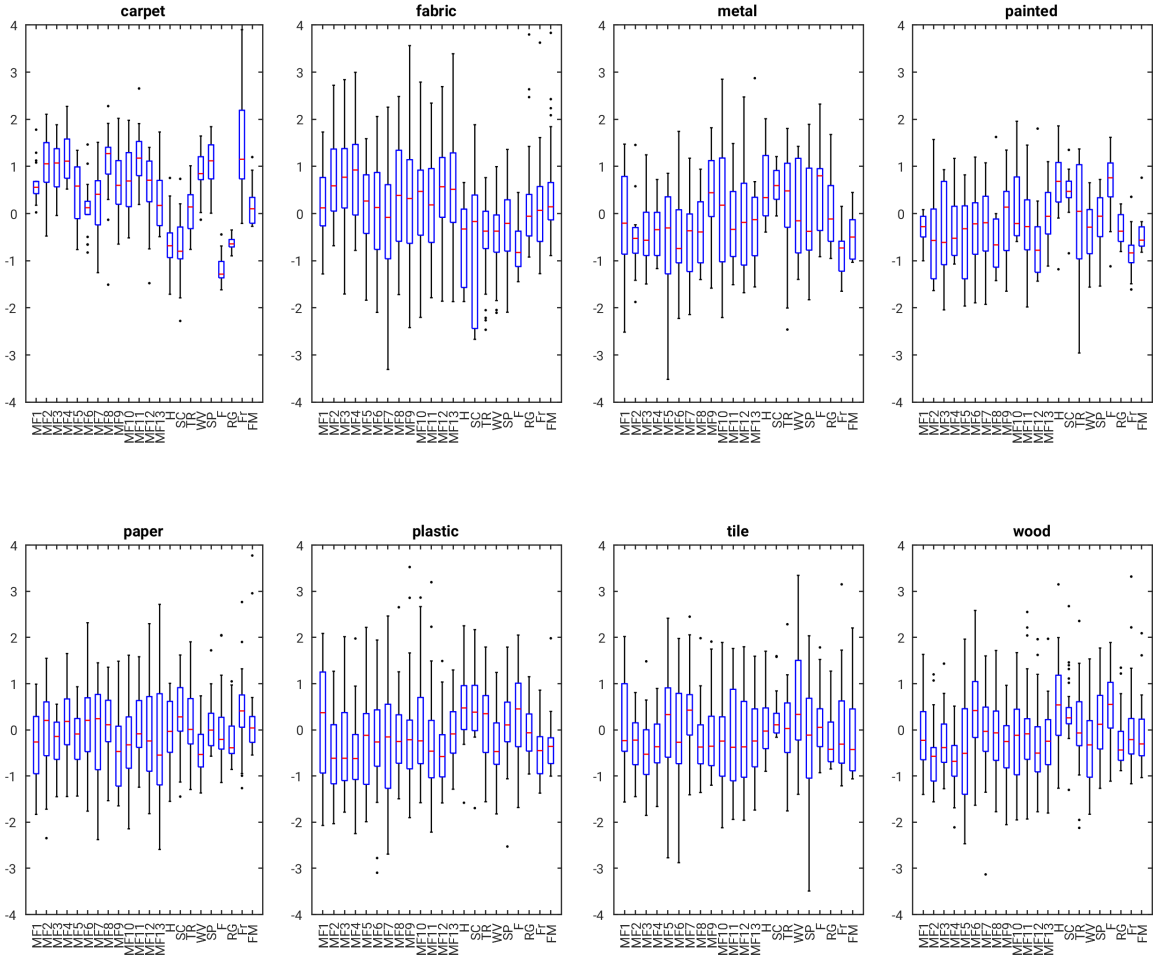


Figure 6.7: These eight plots show the distribution of scan-free features for each of the MINC categories represented in our dataset. For each category, box plots are shown for the all the features covered in Sec. 4.6.1 on page 73. In each subplot, there is one box plot for each of the haptic properties. In each, the horizontal red bar marks the median value of that property, the blue box shows the extent of the 25th and 75th percentiles, whiskers reach to the highest and lowest non-outlier values, and black dots mark automatically detected outliers.

To measure the haptic properties, we use the vibration data recorded by the two high-bandwidth accelerometers during end-effector/surface interaction. The recordings consist of alternating periods of tapping and dragging. These are automatically segmented by a peak-finding algorithm: specifically, MATLAB's `findpeaks` function with the criteria given in Table 6.1 on the next page. Any non-tapping periods of less than 5 seconds are discarded (these are usually the time in between taps when the end-effector was not actually contacting the surface). We then calculate Strese et al.'s hardness and spectral centroid on the tapping periods, while the dragging data is used to calculate the spikiness, waviness, temporal roughness, regularity, fineness, friction, and normalized friction properties defined in that work [4, 5]. In addition, there are several properties based on bands of vibration frequency: Strese et al.'s MFCC (melfrequency cepstral coefficients) and simple binned frequencies as in Romano et al.'s work. Lastly, Romano et al.'s classifiers use the tangential and normal forces as separate features for each window of a signal; instead, we take the ratio and average over an entire dragging period to get the coefficient of friction.

Due to space and time constraints, we did not use all of the frequency band properties: from melfrequency cepstral coefficient (MFCC) #1-13, we used 1, 4, 7, 10, and 13, while from the ten frequency bins we used 1 and 10. Taken together with the other features previously mentioned, we are fitting 17 different models to the same dataset and testing for significant correlation. This number would imply that instead of the traditional standard of $p < 0.05$ for significance, we should demand $p < 0.003$, applying the Bonferroni or Šidák correction.)

Since there is much more work on training computer vision classifiers, as opposed to regressors, we adopt a cross-training strategy. We use the three-level Texture CNN (T-CNN-3) developed by Andrearczyk and Whelan [89]. Its layer structure is shown graphically in Fig. 6.8 on page 125. Using some layers pretrained for object classification on ImageNET, this network is fine-tuned to classify materials in the Textures under varying Illumination, Pose, and Scale (KTH-TIPS2) dataset, which contains closeup images of materials under varying pose and illumination. We assume that by learning to classify, the network's internal

Parameter	Definition	Value
`MinPeakProminence`	Minimum height of the peak above the lowest valley on the way to the next peak	10
`MaxPeakWidth`	Maximum distance between the peak and the places where the signal dips below half the peak's prominence	1000
`MinPeakDistance`	Minimum separation between two peaks	500
`NPeaks`	Maximum number of peaks to find	500

Table 6.1: Configuration parameters used with MATLAB's `findpeaks` function to automatically locate taps using the ATI Mini40 six-axis force/torque sensor output.

representation will include surface characteristics that are relevant to haptic property regression as well. Accordingly, we use the fine-tuned T-CNN as a feature extractor. The outputs of the last layer before the classifier are fed into support-vector-based learning algorithms.

All neural network manipulation was done using the Caffe framework [90] (version 1.0.0) on a computer with 8 Intel Core i7 processors, 32 GB RAM and dual Nvidia Titan X GPUs (one GPU was used at a time). In the first step, the T-CNN network was fine-tuned from its original ImageNET training to the KTH-TIPS2 texture dataset for 5000 iterations, achieving 73% training accuracy. (In contrast to the original methodology, which performs cross-validation by training on only 25% of the dataset at a time, we no longer need to evaluate the classification, so we train it on all of the KTH-TIPS2 images. Surprisingly, this does not significantly affect the accuracy.) Referring again to the network architecture shown in Fig. 6.8 on the next page, we evaluate the network on each image in our dataset and extract the activations just before the "fc8bis" layer (note that the two dropout layers are removed after training the T-CNN). This process results in 4096 features for each image.

To condition the inputs and outputs for SVM training, we use the *z-score*: given a set of observations $x_1 \ldots x_N$, the z-score is calculated by centering at the mean and dividing by

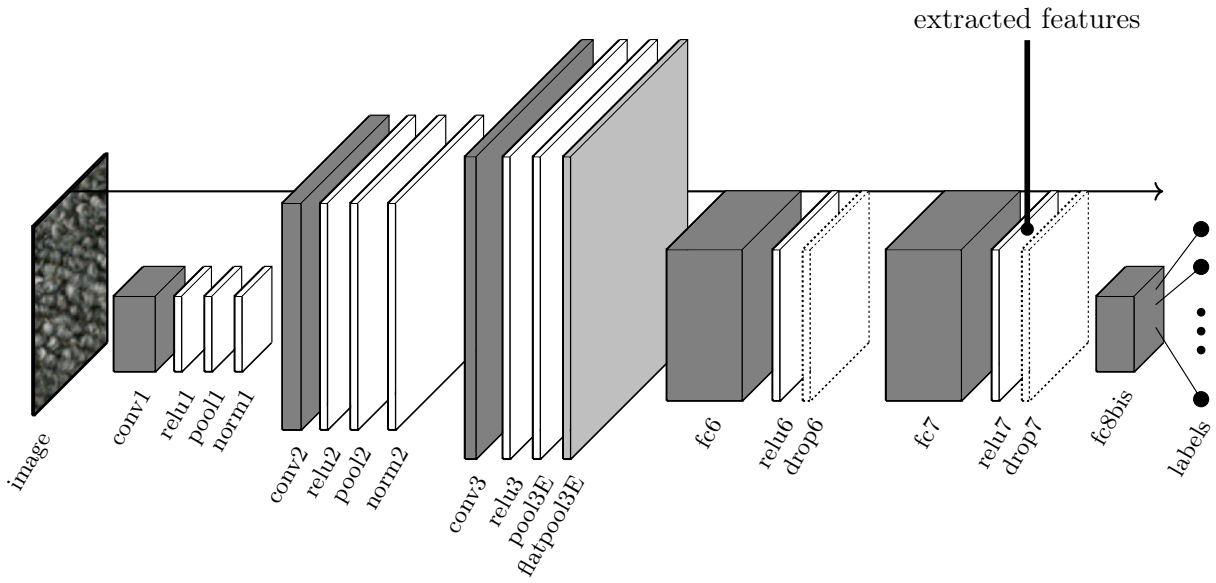


Figure 6.8: T-CNN architecture. Layer names are shown using the same nomenclature as in the T-CNN paper [89]. The input cropped surface image, shown at left, is processed by three convolution units in sequence, then three fully connected layers (dotted "dropout" layers are only present in the training phase). For our regression task, we extract features after the second-to-last fully connected layer.

the standard deviation:

$$z_i = \frac{x_i - \mu}{\sigma} = \frac{x_i - \frac{1}{N}\sum_{j=1}^{N} x_j}{\sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(x_j - \frac{1}{N}\sum_{k=1}^{N} x_k\right)^2}} \qquad \forall i = 1 \ldots N \qquad (6.1)$$

In other words, the z-score transforms a value from an absolute scale to a number of standard deviations away from the mean, removing the influence of units or scale. Whenever we create a model, the inputs are conditioned using the mean and standard deviation of the training set. In regression tasks, the outputs are also standardized the same way. This process prevents values with differing scales from dominating the learning algorithm, and it makes the error measures more comparable on the output side.

Category	Split 1		Split 2		Split 3		Split 4		Test	
	I	M	I	M	I	M	I	M	I	M
Brick	0	0	0	0	14	2	10	1	10	1
Carpet	60	8	33	5	85	13	49	8	57	6
Ceramic	0	0	0	0	0	0	5	1	0	0
Fabric	169	26	118	20	171	27	153	24	137	22
Leather	30	6	30	6	35	5	5	1	15	3
Metal	35	5	47	6	89	10	35	5	15	2
Other	188	29	140	24	100	17	116	20	120	19
Painted Surface	21	4	24	3	81	12	36	5	30	3
Paper	108	15	92	15	115	18	59	10	52	8
Plastic	87	12	89	14	115	16	70	12	30	5
Polished Stone	20	2	20	2	92	10	30	4	25	4
Stone	20	4	39	6	35	14	35	5	30	7
Tile	85	13	62	8	106	15	48	8	75	13
Wood	166	18	129	17	117	15	131	20	152	19

Table 6.2: Number of images (I) and materials (M) in each of the cross-validation splits and the held-out test set, separated by MINC [83] category.

6.4.1. Results

Having extracted features for all images in our dataset, we split the dataset into five folds: four used for training with cross validation, and one held out for testing. The splits are randomly determined, using the MINC categories to balance material types. The distribution of materials in each split is shown in Table 6.2. Holding out a test set, which is never processed until the model parameters have been fine-tuned using the training folds, simulates the real-world application of a trained machine learning model encountering new data which was not available during training. However, we still need to fine-tune the model parameters by training it on some data and then testing it on some other data. Further partitioning the training set risks making the available training data too small, so to mitigate this we use the technique of cross-validation: split the training set into some number of folds (here, four), and train the model that number of times, each time holding out one of the folds as a temporary test set. A particular model's performance can be estimated by averaging metrics over all four folds. When this process is finished, all folds of the training set are used to train the final model that is evaluated on the held-out test set. Held-out test sets and cross-validation are common strategies in the machine learning field for training robust, unbiased models.

For the scan-dependent and scan-free features, we use support vector regression (ν-SVR with a Gaussian kernel). To predict the experimenter ratings, we use support vector classification (C-SVC, also with a Gaussian kernel). Grid search[1] over the cross-validation splits helped to determine optimal values for ν, C, and γ for each model. We investigated the utility of feature selection to narrow down the 4096 inputs that we get from the T-CNN activations. This process involves iteratively including increasing numbers of features in the training process, chosen based on their utility in fitting a simple model to the data. Specifically, we used MATLAB's `sequentialfs` feature selector, optimizing the deviance of a generalized linear model (`glmfit`). Fig. 6.9 on the next page shows the performance of one regressor (waviness) in cross-validation. Clearly, feature selection is advantageous as a whole, at least on the training set, and the performance stabilizes at about 60 features. Based on this encouraging initial result, we decided to go forward using 60 of the 4096 features selected in this manner, holding out the first cross-validation fold to evaluate the performance. Unfortunately, the observed performance increase did not translate to the held-out test set. Both in cross-validation and using the held-out test set, feature selection reduced overfitting by generally reducing performance on the training set. But the corresponding improved performance on the test set was seen only in cross-validation. Possibly, 60 features was not an optimal number. It would be improper to backtrack and report test set results without feature selection, since any decisions (such as turning off feature selection) taken after observing the test set performance are, by definition, tainted. Therefore we present results both with and without feature selection.

Results are shown in Tables 6.5 through 6.4 on pages 132–131 with a variety of statistical tests. Bolded rows show the best-performing classifiers and regressors.

To evaluate regression, we can look at the correlation between the actual and expected output [91]. Measuring the correlation also produces a significance value p, which guards against relying on spurious correlation by indicating the likelihood that a particular corre-

[1] Grid search means trying all possible combinations of parameters and keeping the one that garners the best performance.

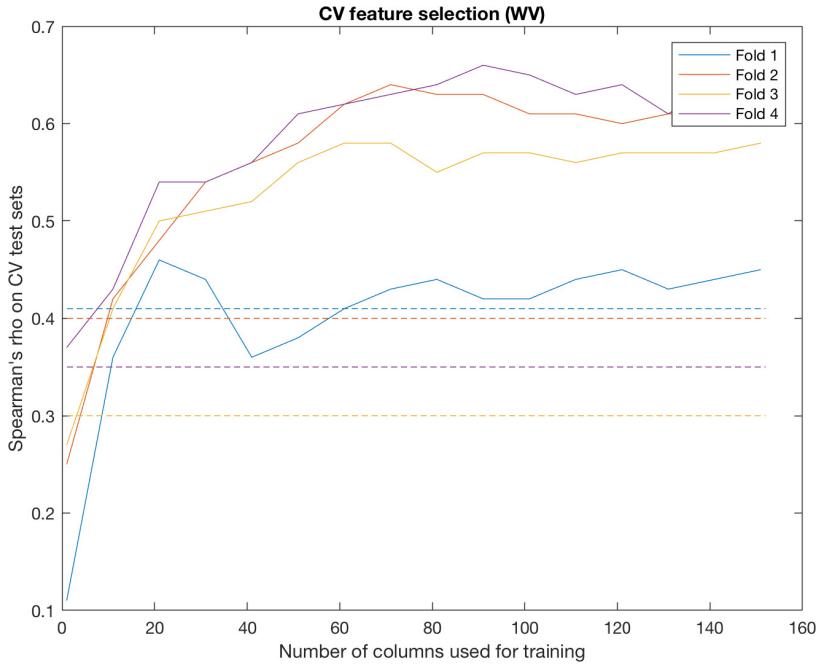


Figure 6.9: Increase in performance of regression under sequential feature selection. We analyzed the performance within the training set while holding out the first cross-validation fold as a test set. Regression correlations (solid lines) are plotted against the number of features selected. The dotted lines represent the regression correlation with all features included.

lation could arise from measuring variables that were actually independent. Besides this, we can fit a simple linear trend line to the regression output and observe the slope. Since a perfect regressor's output would have a slope of 1, while a random number generator would have a slope of 0, this is a reasonable measurement. We further calculate the R^2, a measure of how much variation in the output is explained by the input (higher is better). Finally, a direct assessment of the regression performance can be had by computing the root-mean-square error (RMSE) of the actual output with respect to the expected output.

For the classifiers, we also want to calculate a significance level, but it is not clear what to compare against. Since a very simple model might simply take the most common value in the training set and blindly output that while disregarding the input, we take this as the baseline and use McNemar's test [92] to compare it against the actual trained model. Also, we compute the accuracy, precision P, and recall R (higher is better). For a particular

output rating i, precision (or positive predictive value) is the ratio of the number of times i was *correctly* output to the total number of times it was output, while recall (or true positive rate) is the ratio of the number of times i was correctly output to the number of times some other value was produced when i was expected. These two metrics are more informative than simply calculating the accuracy because we have an unbalanced dataset: for example, the dummy model discussed above would have high recall but low precision. Finally, since this is being treated as a classification task, while the classes are actually numerically related, we can calculate the mean absolute error (MAE, or simply the average difference between the output rating and the Proton operator's rating).

Figs. 6.14 through 6.13 on pages 138–137 show the regression outputs respectively averaged over cross-validation training sets, averaged over cross-validation test sets, on the whole training set, and on the held-out test set, while Figs. 6.22 through 6.21 on pages 144–143 graphically show the classifier outputs.

For the regressor figures, each plot shows the expected output vs. actual support vector regression (SVR) output as blue dots, a trend line in red for that scatterplot, and the theoretically ideal trend line (where the actual output and expected output are identical) in yellow. Better models have the two trend lines closer together.

In the classifier figures, each plot shows a confusion matrix comparing the expected and actual outputs of the SVM. Better models have higher values (darker squares) on the diagonal of the matrix.

NOFS-CV

Property	CV Training set performance					CV Test set performance				
	ρ	m	p	R^2	RMSE	ρ	m	p	R^2	RMSE
Frequency bin 1	0.97	0.87	$< 10^{-100}$	0.94	0.24	0.27	0.091	$1.1 * 10^{-6}$	0.049	0.95
Frequency bin 10	0.91	0.59	$< 10^{-100}$	0.75	0.5	0.16	0.035	0.091	-0.036	0.97
Friction coefficient	**0.99**	**0.95**	$\mathbf{< 10^{-100}}$	**0.98**	**0.14**	**0.31**	**0.13**	$\mathbf{9.1 * 10^{-12}}$	**0.058**	**0.96**
Hardness	0.99	0.93	$< 10^{-100}$	0.97	0.17	0.28	0.11	0.0002	0.068	0.96
Spectral centroid	0.99	0.92	$< 10^{-100}$	0.97	0.17	0.18	0.051	0.011	-0.024	1
MF cepstral coeff. 1	**0.98**	**0.77**	$\mathbf{< 10^{-100}}$	**0.92**	**0.28**	**0.32**	**0.12**	$\mathbf{4.9 * 10^{-19}}$	**0.026**	**0.97**
MF cepstral coeff. 4	0.99	0.93	$< 10^{-100}$	0.97	0.17	0.29	0.12	$2.5 * 10^{-10}$	0.067	0.97
MF cepstral coeff. 7	0.97	0.75	$< 10^{-100}$	0.9	0.31	0.24	0.084	$1.7 * 10^{-8}$	0.019	0.99
MF cepstral coeff. 10	0.97	0.69	$< 10^{-100}$	0.87	0.35	0.23	0.047	0.00013	0.048	0.97
MF cepstral coeff. 13	0.99	0.93	$< 10^{-100}$	0.97	0.17	0.25	0.1	$1.1 * 10^{-11}$	-0.0045	0.99
Spikiness	0.99	0.92	$< 10^{-100}$	0.97	0.18	0.22	0.073	$3 * 10^{-6}$	0.025	0.98
Temporal roughness	0.99	0.94	$< 10^{-100}$	0.98	0.14	0.18	0.062	0.011	-0.037	0.99
Waviness	**0.99**	**0.94**	$\mathbf{< 10^{-100}}$	**0.97**	**0.16**	**0.36**	**0.14**	$\mathbf{3.8 * 10^{-13}}$	**0.12**	**0.93**
Regularity	0.96	0.72	$< 10^{-100}$	0.88	0.35	0.12	0.026	0.18	-0.043	1
Fineness	0.99	0.94	$< 10^{-100}$	0.98	0.14	0.28	0.1	$4.1 * 10^{-5}$	0.055	0.97
Friction	0.99	0.92	$< 10^{-100}$	0.97	0.18	0.2	0.066	0.003	-0.05	1
Friction (normalized)	0.98	0.87	$< 10^{-100}$	0.95	0.23	0.1	0.034	0.11	-0.06	1

Rating	CV Training set performance					CV Test set performance				
	%	p	P	R	MAE	%	p	P	R	MAE
Hardness	100	$< 10^{-100}$	1	1	0	67	1	0.2	0.58	0.71
Roughness	**100**	$\mathbf{< 10^{-100}}$	**0.99**	**1**	**0.0025**	**49**	**0.07**	**0.33**	**0.43**	**0.61**
Warmness	100	$< 10^{-100}$	1	1	0	77	1	0.25	0.77	0.26
Stickiness (tooling ball)	100	$< 10^{-100}$	1	1	0	63	1	0.25	0.55	0.48
Stickiness (OptoForce)	100	$< 10^{-100}$	1	1	0	37	0.18	0.25	0.34	0.81
Stickiness (BioTac)	100	$< 10^{-100}$	1	1	0.0031	37	0.31	0.32	0.31	0.86

Table 6.3: Results of regression and classification without feature selection, in cross validation, with summary statistics. All metrics shown are averaged over cross-validation splits. For the regression tasks, we calculate the correlation ρ between desired and actual output, slope m of the trend line, p-value, R^2, and RMSE, while classification accuracy, p-value with respect to a dummy classifier, precision, recall, and mean absolute error (MAE) are shown for the the classification tasks. Bolded rows show the best-performing models. This table corresponds to Figs. 6.10, 6.11, 6.18 and 6.19.

NOFS-FINAL

Property	Training set performance					Test set performance				
	ρ	m	p	R^2	RMSE	ρ	m	p	R^2	RMSE
Frequency bin 1	0.996	0.864	$< 10^{-100}$	0.94	0.246	0.408	0.144	$1.99 * 10^{-31}$	0.159	0.896
Frequency bin 10	0.995	0.571	$< 10^{-100}$	0.737	0.513	0.292	0.138	$4.02 * 10^{-16}$	0.121	0.739
Friction coefficient	0.995	0.943	$< 10^{-100}$	0.978	0.148	0.279	0.139	$7.63 * 10^{-15}$	0.0417	1.01
Hardness	0.992	0.926	$< 10^{-100}$	0.968	0.178	0.283	0.118	$3.25 * 10^{-15}$	-0.0882	0.866
Spectral centroid	0.997	0.916	$< 10^{-100}$	0.969	0.177	0.303	0.106	$2.26 * 10^{-17}$	0.0215	0.861
MF cepstral coeff. 1	**0.969**	**0.756**	$\mathbf{< 10^{-100}}$	**0.911**	**0.299**	**0.464**	**0.155**	$\mathbf{3.46 * 10^{-41}}$	**0.112**	**0.978**
MF cepstral coeff. 4	0.99	0.926	$< 10^{-100}$	0.968	0.18	0.272	0.0838	$3.37 * 10^{-14}$	0.0147	1.13
MF cepstral coeff. 7	0.958	0.733	$< 10^{-100}$	0.894	0.326	0.172	0.0473	$2.19 * 10^{-6}$	-0.0347	0.878
MF cepstral coeff. 10	0.97	0.695	$< 10^{-100}$	0.876	0.352	0.0665	0.0346	0.0692	-0.012	0.952
MF cepstral coeff. 13	0.991	0.923	$< 10^{-100}$	0.964	0.189	0.265	0.11	$1.91 * 10^{-13}$	0.0433	0.983
Spikiness	**0.994**	**0.917**	$\mathbf{< 10^{-100}}$	**0.965**	**0.188**	**0.457**	**0.22**	$\mathbf{6.34 * 10^{-40}}$	**0.241**	**0.817**
Temporal roughness	0.993	0.934	$< 10^{-100}$	0.975	0.157	0.253	0.0627	$2.19 * 10^{-12}$	0.0135	1.04
Waviness	**0.995**	**0.937**	$\mathbf{< 10^{-100}}$	**0.974**	**0.162**	**0.433**	**0.176**	$\mathbf{1.38 * 10^{-35}}$	**0.166**	**0.847**
Regularity	0.975	0.711	$< 10^{-100}$	0.872	0.357	0.148	0.0641	$4.85 * 10^{-5}$	-0.0168	0.756
Fineness	0.991	0.929	$< 10^{-100}$	0.976	0.156	0.26	0.0932	$5.41 * 10^{-13}$	0.0291	1.11
Friction	0.994	0.907	$< 10^{-100}$	0.963	0.193	0.255	0.192	$1.32 * 10^{-12}$	0.221	0.997
Friction (normalized)	0.997	0.862	$< 10^{-100}$	0.944	0.237	0.288	0.0732	$9.22 * 10^{-16}$	0.0311	0.776

Rating	Training set performance					Test set performance				
	%	p	P	R	MAE	%	p	P	R	MAE
Hardness	100	$< 10^{-100}$	1	1	0	56	1	0.2	0.56	0.71
Roughness	100	$< 10^{-100}$	0.98	1	0.0043	38	0.31	0.24	0.3	0.74
Warmness	100	$< 10^{-100}$	1	1	0	81	1	0.25	0.81	0.24
Stickiness (tooling ball)	100	$< 10^{-100}$	1	1	0	66	1	0.33	0.66	0.47
Stickiness (OptoForce)	100	$< 10^{-100}$	1	1	0	36	0.043	0.27	0.29	0.7
Stickiness (BioTac)	**100**	$\mathbf{< 10^{-100}}$	**0.99**	**1**	**0.0043**	**28**	$\mathbf{3.7 * 10^{-7}}$	**0.22**	**0.37**	**0.93**

Table 6.4: Results of regression and classification without feature selection, on the entire training set and held-out test set, with summary statistics. For the regression tasks, we calculate the correlation ρ between desired and actual output, slope m of the trend line, p-value, R^2, and RMSE, while classification accuracy, p-value with respect to a dummy classifier, precision, recall, and MAE are shown for the the classification tasks. Bolded rows show the best-performing models. This table corresponds to Figs. 6.12, 6.13, 6.20 and 6.21.

FS-CV

Property	CV Training set performance					CV Test set performance				
	ρ	m	p	R^2	RMSE	ρ	m	p	R^2	RMSE
Frequency bin 1	0.648	0.328	$< 10^{-100}$	0.393	0.778	0.448	0.212	$4.47 * 10^{-14}$	0.188	0.895
Frequency bin 10	0.567	0.323	$< 10^{-100}$	0.465	0.73	0.298	0.12	$9.6 * 10^{-12}$	0.0438	0.939
Friction coefficient	**0.714**	**0.454**	$\mathbf{< 10^{-100}}$	**0.508**	**0.7**	**0.551**	**0.316**	$\mathbf{4.75 * 10^{-14}}$	**0.266**	**0.826**
Hardness	0.684	0.408	$< 10^{-100}$	0.485	0.716	0.53	0.258	$6.75 * 10^{-16}$	0.243	0.85
Spectral centroid	0.626	0.371	$< 10^{-100}$	0.452	0.736	0.472	0.185	0.00316	0.123	0.898
MF cepstral coeff. 1	0.723	0.447	$< 10^{-100}$	0.509	0.699	0.55	0.316	$2.31 * 10^{-34}$	0.267	0.84
MF cepstral coeff. 4	**0.708**	**0.404**	$\mathbf{< 10^{-100}}$	**0.467**	**0.728**	**0.574**	**0.29**	$\mathbf{3.96 * 10^{-31}}$	**0.282**	**0.835**
MF cepstral coeff. 7	0.684	0.386	$< 10^{-100}$	0.467	0.728	0.544	0.269	$2.42 * 10^{-17}$	0.225	0.861
MF cepstral coeff. 10	0.685	0.404	$< 10^{-100}$	0.481	0.717	0.492	0.243	$6.76 * 10^{-10}$	0.219	0.869
MF cepstral coeff. 13	0.684	0.401	$< 10^{-100}$	0.481	0.718	0.508	0.26	$3.1 * 10^{-15}$	0.193	0.865
Spikiness	0.695	0.375	$< 10^{-100}$	0.454	0.737	0.488	0.236	$5.87 * 10^{-9}$	0.241	0.861
Temporal roughness	0.64	0.387	$< 10^{-100}$	0.473	0.723	0.474	0.252	$2.19 * 10^{-16}$	0.232	0.851
Waviness	**0.732**	**0.512**	$\mathbf{< 10^{-100}}$	**0.574**	**0.652**	**0.562**	**0.367**	$\mathbf{3.18 * 10^{-39}}$	**0.347**	**0.796**
Regularity	0.585	0.391	$< 10^{-100}$	0.511	0.696	0.305	0.127	0.218	0.0461	0.943
Fineness	0.696	0.411	$< 10^{-100}$	0.462	0.732	0.528	0.28	$5.63 * 10^{-25}$	0.252	0.857
Friction	0.644	0.39	$< 10^{-100}$	0.46	0.732	0.434	0.218	$3.71 * 10^{-5}$	0.146	0.899
Friction (normalized)	0.619	0.392	$< 10^{-100}$	0.49	0.709	0.353	0.148	$5.37 * 10^{-9}$	0.0177	0.942

Rating	CV Training set performance					CV Test set performance				
	%	p	P	R	MAE	%	p	P	R	MAE
Hardness	71.3	$9.09 * 10^{-13}$	0.336	0.869	0.568	67	0.224	0.231	0.534	0.687
Roughness	**81**	$\mathbf{< 10^{-100}}$	**0.8**	**0.875**	**0.21**	**55.3**	**0.00556**	**0.411**	**0.397**	**0.519**
Warmness	77.9	0.25	0.305	0.873	0.249	77	0.75	0.249	0.668	0.266
Stickiness (tooling ball)	72.2	$1.59 * 10^{-34}$	0.478	0.863	0.342	63	0.04	0.281	0.553	0.458
Stickiness (OptoForce)	90.6	$< 10^{-100}$	0.919	0.936	0.115	39.3	0.388	0.325	0.327	0.784
Stickiness (BioTac)	60.8	$9.96 * 10^{-86}$	0.491	0.705	0.52	39.2	0.0581	0.349	0.385	0.769

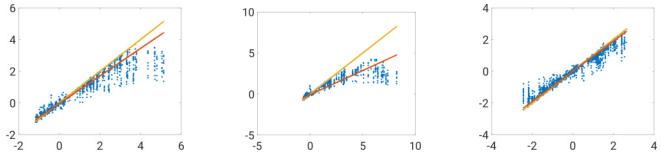
Table 6.5: Results of regression and classification after feature selection, in cross validation, with summary statistics. All metrics shown are averaged over cross-validation splits. For the regression tasks, we calculate the correlation ρ between desired and actual output, slope m of the trend line, p-value, R^2, and RMSE, while classification accuracy, p-value with respect to a dummy classifier, precision, recall, and MAE are shown for the the classification tasks. Bolded rows show the best-performing models. This table corresponds to Figs. 6.14, 6.15, 6.22 and 6.23.

FS-FINAL

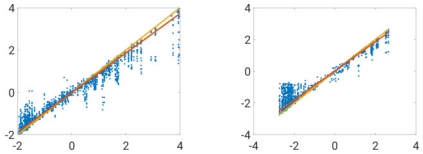
Property	Training set performance					Test set performance				
	ρ	m	p	R^2	RMSE	ρ	m	p	R^2	RMSE
Frequency bin 1	0.999	0.853	$< 10^{-100}$	0.931	0.262	0.378	0.0755	$6.96 * 10^{-27}$	0.0995	0.927
Frequency bin 10	0.975	0.333	$< 10^{-100}$	0.508	0.702	0.0999	0.000494	0.00624	-0.00234	0.789
Friction coefficient	0.998	0.955	$< 10^{-100}$	0.986	0.119	0.248	0.094	$6.22 * 10^{-12}$	0.0568	0.998
Hardness	0.997	0.931	$< 10^{-100}$	0.973	0.165	0.26	0.0761	$4.69 * 10^{-13}$	-0.067	0.858
Spectral centroid	0.997	0.916	$< 10^{-100}$	0.969	0.177	0.303	0.106	$2.24 * 10^{-17}$	0.0216	0.861
MF cepstral coeff. 1	**0.998**	**0.955**	$\mathbf{< 10^{-100}}$	**0.986**	**0.117**	**0.446**	**0.102**	$\mathbf{6.93 * 10^{-38}}$	**0.0875**	**0.991**
MF cepstral coeff. 4	0.997	0.936	$< 10^{-100}$	0.977	0.151	0.237	0.054	$5.73 * 10^{-11}$	0.0299	1.12
MF cepstral coeff. 7	0.997	0.925	$< 10^{-100}$	0.972	0.167	0.201	0.0284	$3.04 * 10^{-8}$	-0.0016	0.864
MF cepstral coeff. 10	0.998	0.944	$< 10^{-100}$	0.98	0.142	0.102	0.0529	0.00504	0.00755	0.942
MF cepstral coeff. 13	0.997	0.931	$< 10^{-100}$	0.973	0.163	0.233	0.066	$1.17 * 10^{-10}$	0.032	0.989
Spikiness	**0.994**	**0.917**	$\mathbf{< 10^{-100}}$	**0.965**	**0.188**	**0.457**	**0.22**	$\mathbf{6.34 * 10^{-40}}$	**0.241**	**0.817**
Temporal roughness	0.998	0.931	$< 10^{-100}$	0.977	0.152	0.17	0.0247	$2.85 * 10^{-6}$	-0.00663	1.05
Waviness	0.998	0.924	$< 10^{-100}$	0.973	0.164	0.405	0.0953	$5.97 * 10^{-31}$	0.0958	0.882
Regularity	0.958	0.396	$< 10^{-100}$	0.598	0.634	0.105	0.00134	0.0042	0.000717	0.75
Fineness	0.996	0.954	$< 10^{-100}$	0.986	0.119	0.216	0.056	$2.23 * 10^{-9}$	0.0374	1.1
Friction	0.999	0.911	$< 10^{-100}$	0.963	0.191	0.254	0.153	$1.63 * 10^{-12}$	0.202	1.01
Friction (normalized)	0.999	0.818	$< 10^{-100}$	0.909	0.301	0.267	0.0428	$1.17 * 10^{-13}$	0.037	0.774

Rating	Training set performance					Test set performance				
	%	p	P	R	MAE	%	p	P	R	MAE
Hardness	100	$< 10^{-100}$	1	1	0	55.6	0.659	0.219	0.34	0.699
Roughness	100	$< 10^{-100}$	1	1	0	36.1	0.0664	0.222	0.465	0.79
Warmness	99.8	$< 10^{-100}$	0.992	0.999	0.00242	78.5	0.000508	0.264	0.505	0.265
Stickiness (tooling ball)	99.9	$< 10^{-100}$	1	1	0.000538	66.7	0.728	0.401	0.623	0.456
Stickiness (OptoForce)	100	$< 10^{-100}$	1	1	0	36.4	0.0421	0.267	0.274	0.707
Stickiness (BioTac)	**99.9**	$\mathbf{< 10^{-100}}$	**0.999**	**0.999**	**0.001**	**32.2**	**0.000253**	**0.203**	**0.256**	**0.864**

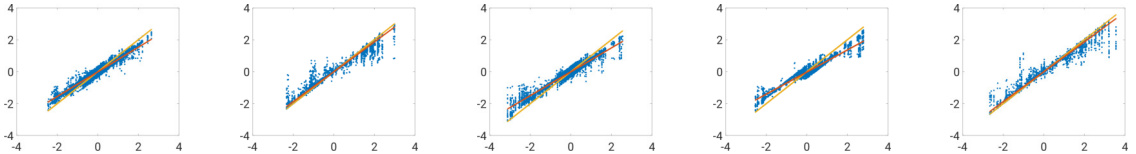
Table 6.6: Results of regression and classification after feature selection, on the entire training set and held-out test set, with summary statistics. For the regression tasks, we calculate the correlation ρ between desired and actual output, slope m of the trend line, p-value, R^2, and RMSE, while classification accuracy, p-value with respect to a dummy classifier, precision, recall, and MAE are shown for the the classification tasks. Bolded rows show the best-performing models. This table corresponds to Figs. 6.16, 6.17, 6.24 and 6.25.



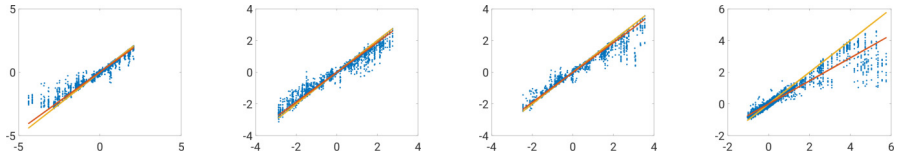
(a) Frequency bin 1 (b) Frequency bin 10 (c) Friction coefficient



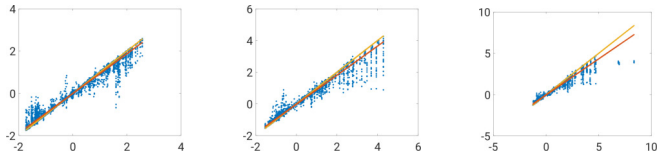
(d) Hardness (e) Spectral centroid



(f) MF cepstral coeff. 1 (g) MF cepstral coeff. 4 (h) MF cepstral coeff. 7 (i) MF cepstral coeff. 10 (j) MF cepstral coeff. 13

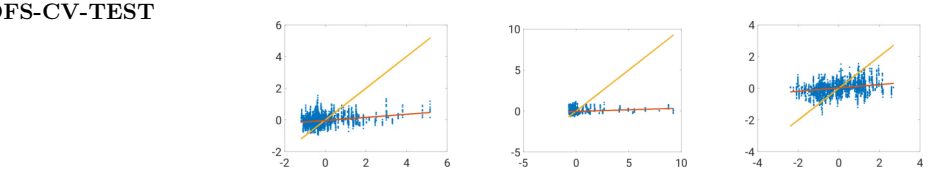


(k) Spikiness (l) Temporal roughness (m) Waviness (n) Regularity

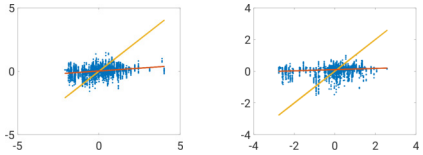


(o) Fineness (p) Friction (q) Friction (norm.)

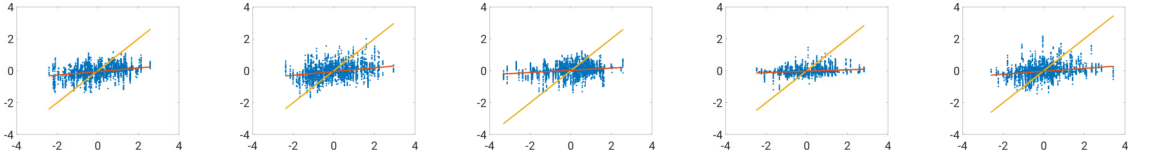
Figure 6.10: Outputs of the model trained on haptic properties, aggregated over cross-validation training sets without feature selection. In each, the red line shows the trend of line of the output, while the yellow line is an ideal regressor ($y = x$). Top row: scan-dependent features. Second row: tapping-based scan-free features. Lower rows: dragging-based scan-free features. See Table 6.3.



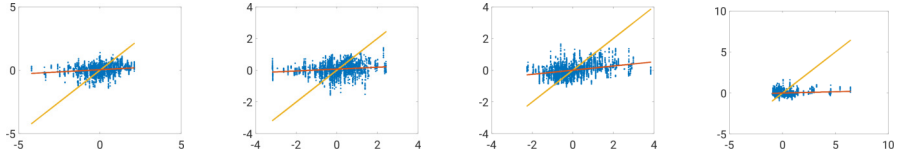
(a) Frequency bin 1 (b) Frequency bin 10 (c) Friction coefficient



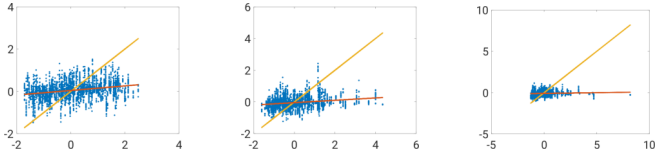
(d) Hardness (e) Spectral centroid



(f) MF cepstral coeff. 1 (g) MF cepstral coeff. 4 (h) MF cepstral coeff. 7 (i) MF cepstral coeff. 10 (j) MF cepstral coeff. 13

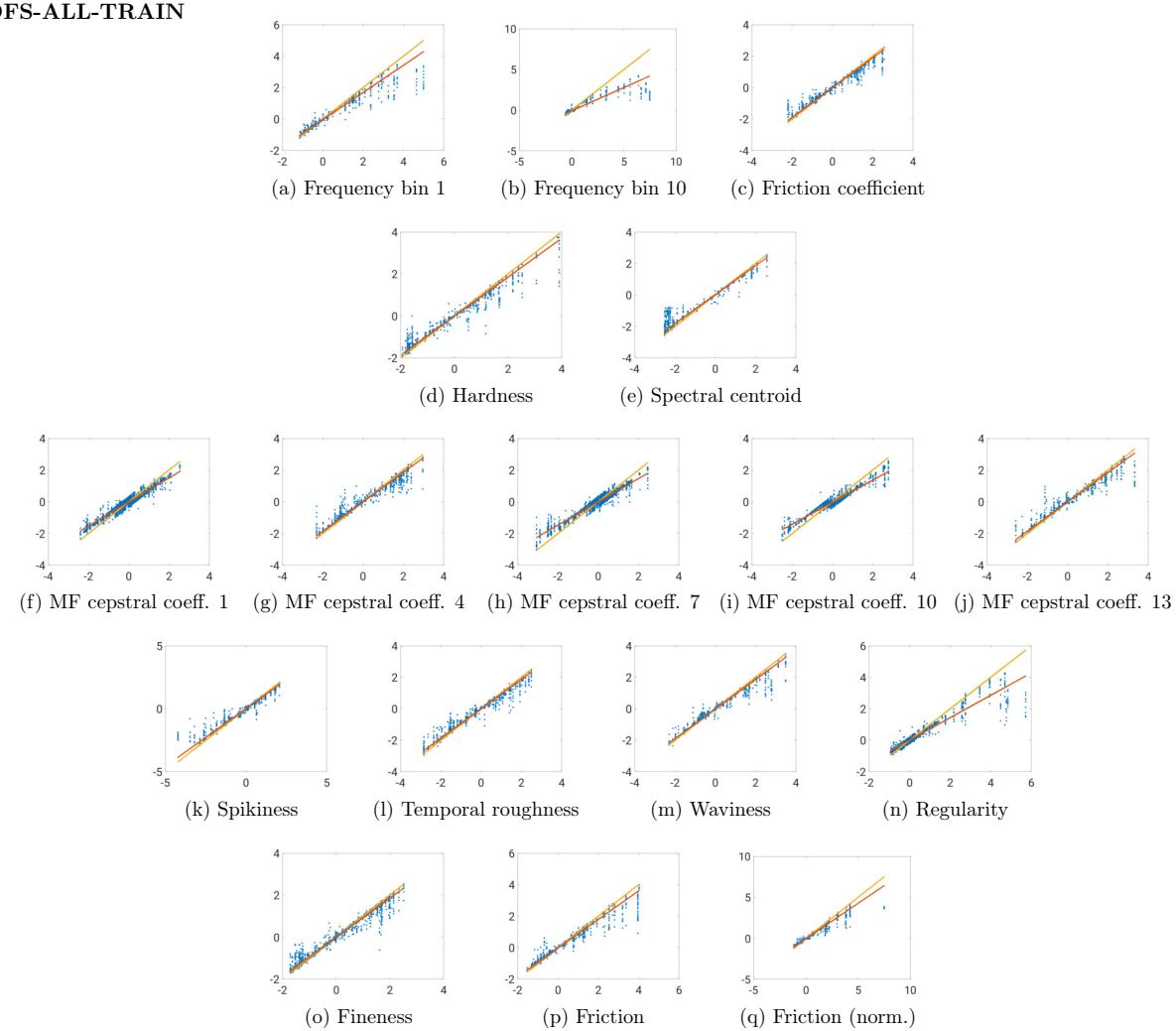


(k) Spikiness (l) Temporal roughness (m) Waviness (n) Regularity



(o) Fineness (p) Friction (q) Friction (norm.)

Figure 6.11: Outputs of the model trained on haptic properties, aggregated over cross-validation test sets without feature selection. In each, the red line shows the trend of line of the output, while the yellow line is an ideal regressor ($y = x$). Top row: scan-dependent features. Second row: tapping-based scan-free features. Lower rows: dragging-based scan-free features. See Table 6.3.



(a) Frequency bin 1 (b) Frequency bin 10 (c) Friction coefficient

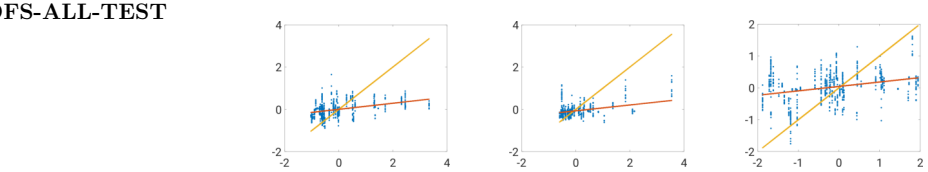
(d) Hardness (e) Spectral centroid

(f) MF cepstral coeff. 1 (g) MF cepstral coeff. 4 (h) MF cepstral coeff. 7 (i) MF cepstral coeff. 10 (j) MF cepstral coeff. 13

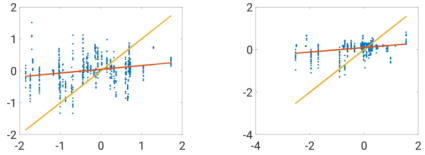
(k) Spikiness (l) Temporal roughness (m) Waviness (n) Regularity

(o) Fineness (p) Friction (q) Friction (norm.)

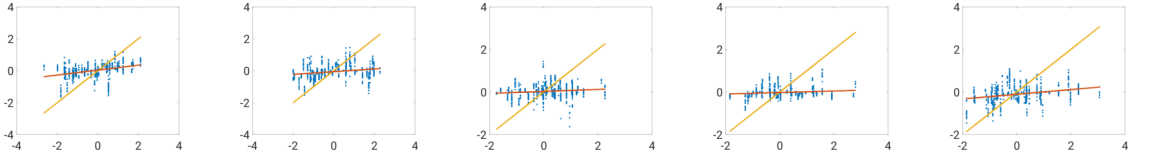
Figure 6.12: Outputs of the model trained on haptic properties, on the whole training set without feature selection. In each, the red line shows the trend of line of the output, while the yellow line is an ideal regressor ($y = x$). Top row: scan-dependent features. Second row: tapping-based scan-free features. Lower rows: dragging-based scan-free features. See Table 6.4.



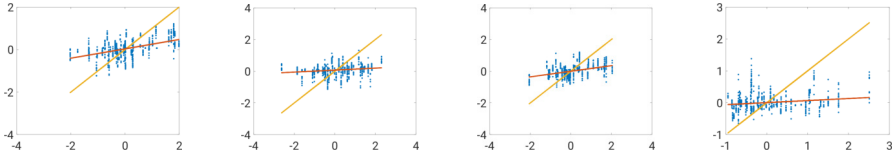
(a) Frequency bin 1 (b) Frequency bin 10 (c) Friction coefficient



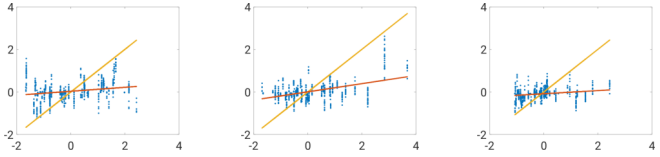
(d) Hardness (e) Spectral centroid



(f) MF cepstral coeff. 1 (g) MF cepstral coeff. 4 (h) MF cepstral coeff. 7 (i) MF cepstral coeff. 10 (j) MF cepstral coeff. 13



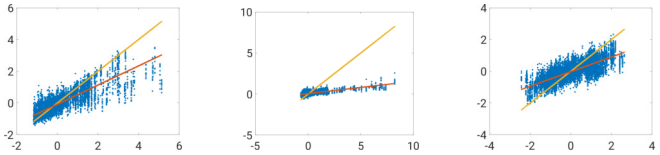
(k) Spikiness (l) Temporal roughness (m) Waviness (n) Regularity



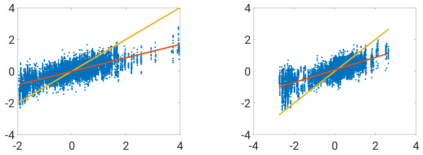
(o) Fineness (p) Friction (q) Friction (norm.)

Figure 6.13: Outputs of the model trained on haptic properties, on the held-out test set without feature selection. In each, the red line shows the trend of line of the output, while the yellow line is an ideal regressor ($y = x$). Top row: scan-dependent features. Second row: tapping-based scan-free features. Lower rows: dragging-based scan-free features. See Table 6.4.

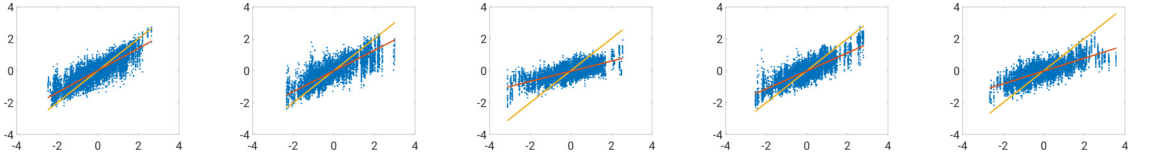
FS-CV-TRAIN



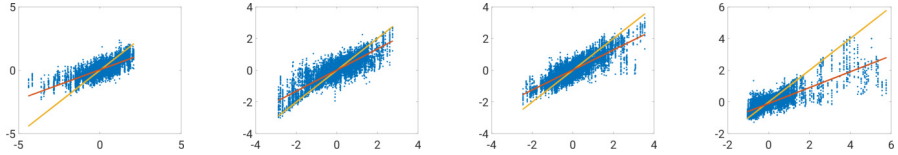
(a) Frequency bin 1 (b) Frequency bin 10 (c) Friction coefficient



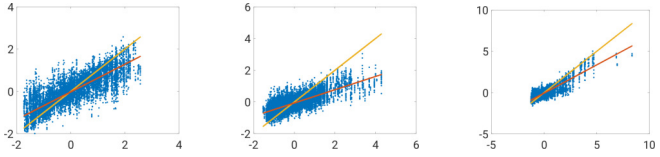
(d) Hardness (e) Spectral centroid



(f) MF cepstral coeff. 1 (g) MF cepstral coeff. 4 (h) MF cepstral coeff. 7 (i) MF cepstral coeff. 10 (j) MF cepstral coeff. 13

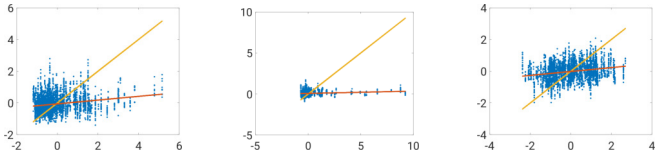


(k) Spikiness (l) Temporal roughness (m) Waviness (n) Regularity

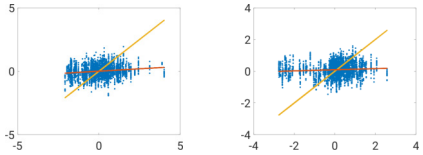


(o) Fineness (p) Friction (q) Friction (norm.)

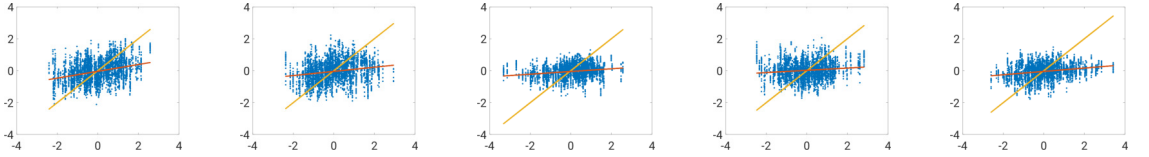
Figure 6.14: Outputs of the model trained on haptic properties, aggregated over cross-validation training sets with feature selection. In each, the red line shows the trend of line of the output, while the yellow line is an ideal regressor ($y = x$). Top row: scan-dependent features. Second row: tapping-based scan-free features. Lower rows: dragging-based scan-free features. See Table 6.5.



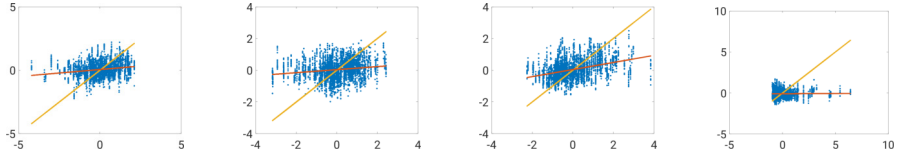
(a) Frequency bin 1 (b) Frequency bin 10 (c) Friction coefficient



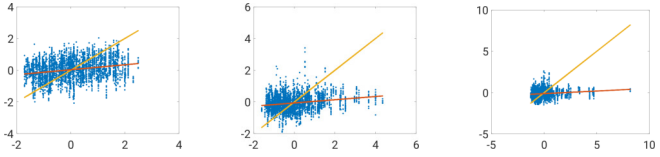
(d) Hardness (e) Spectral centroid



(f) MF cepstral coeff. 1 (g) MF cepstral coeff. 4 (h) MF cepstral coeff. 7 (i) MF cepstral coeff. 10 (j) MF cepstral coeff. 13

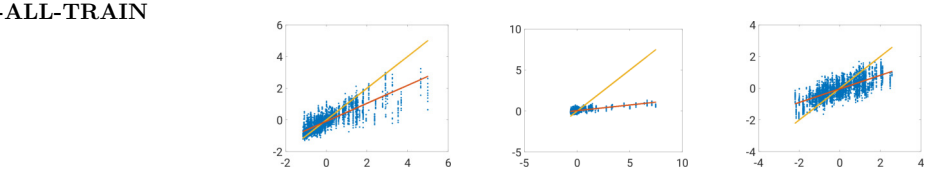


(k) Spikiness (l) Temporal roughness (m) Waviness (n) Regularity

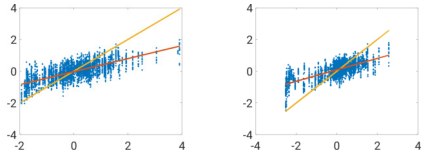


(o) Fineness (p) Friction (q) Friction (norm.)

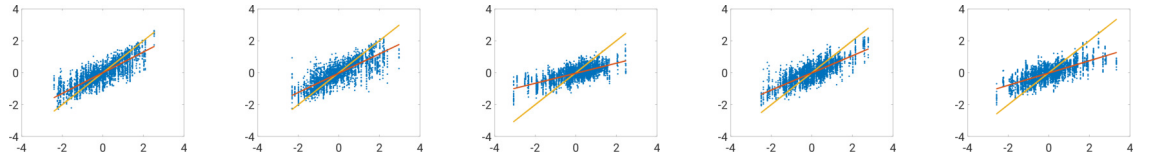
Figure 6.15: Outputs of the model trained on haptic properties, aggregated over cross-validation test sets with feature selection. In each, the red line shows the trend of line of the output, while the yellow line is an ideal regressor ($y = x$). Top row: scan-dependent features. Second row: tapping-based scan-free features. Lower rows: dragging-based scan-free features. See Table 6.5.



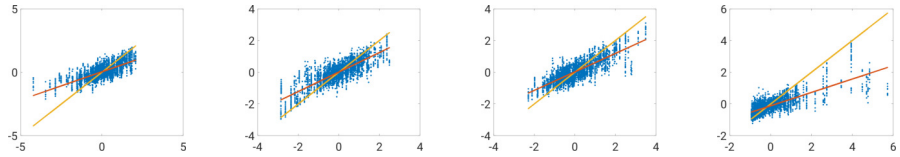
(a) Frequency bin 1 (b) Frequency bin 10 (c) Friction coefficient



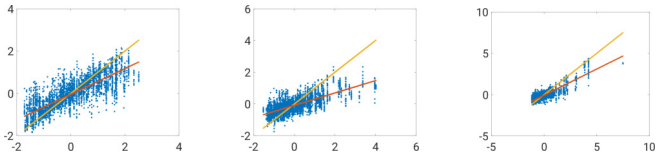
(d) Hardness (e) Spectral centroid



(f) MF cepstral coeff. 1 (g) MF cepstral coeff. 4 (h) MF cepstral coeff. 7 (i) MF cepstral coeff. 10 (j) MF cepstral coeff. 13



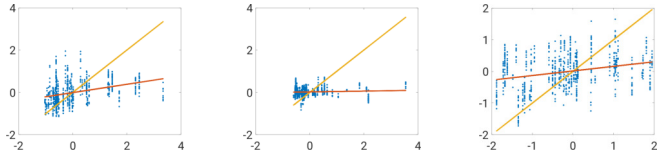
(k) Spikiness (l) Temporal roughness (m) Waviness (n) Regularity



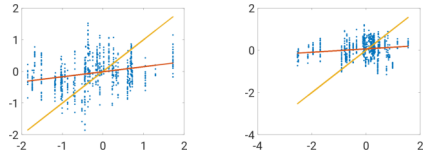
(o) Fineness (p) Friction (q) Friction (norm.)

Figure 6.16: Outputs of the model trained on haptic properties, on the whole training set with feature selection. In each, the red line shows the trend of line of the output, while the yellow line is an ideal regressor ($y = x$). Top row: scan-dependent features. Second row: tapping-based scan-free features. Lower rows: dragging-based scan-free features. See Table 6.6.

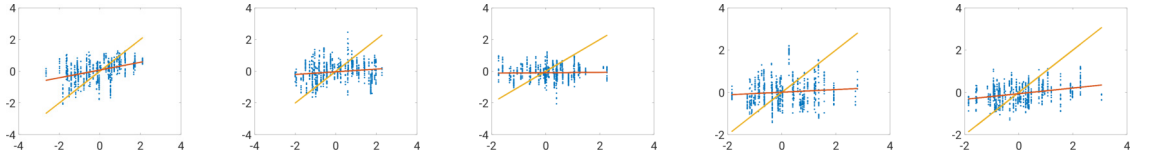
FS-ALL-TEST



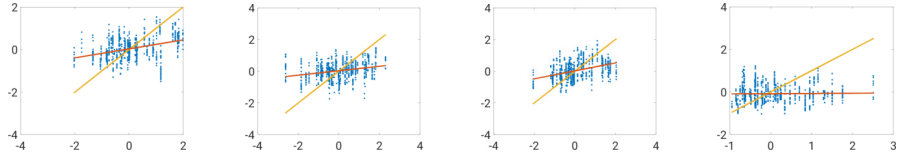
(a) Frequency bin 1 (b) Frequency bin 10 (c) Friction coefficient



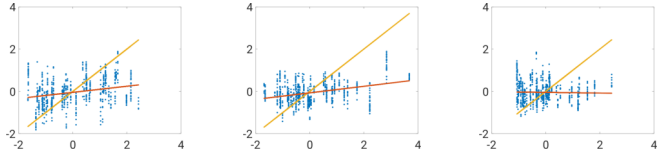
(d) Hardness (e) Spectral centroid



(f) MF cepstral coeff. 1 (g) MF cepstral coeff. 4 (h) MF cepstral coeff. 7 (i) MF cepstral coeff. 10 (j) MF cepstral coeff. 13



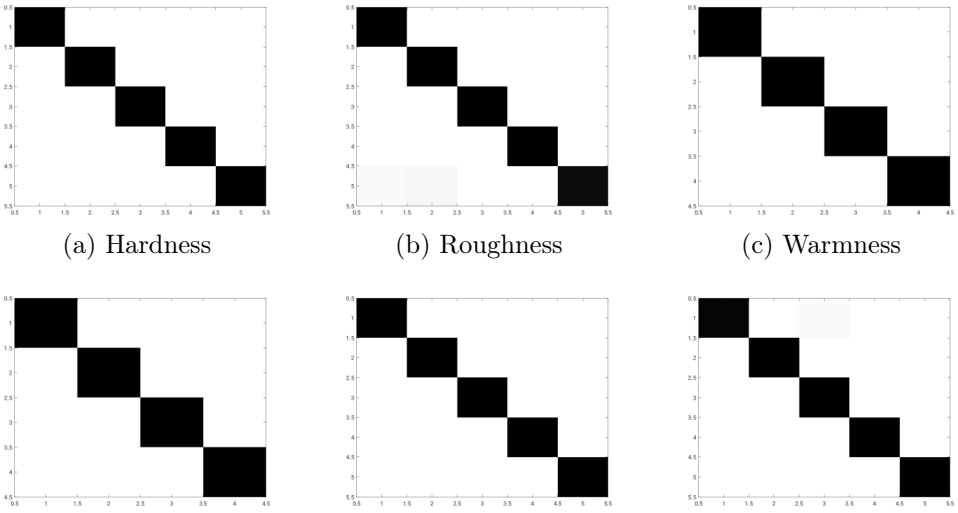
(k) Spikiness (l) Temporal roughness (m) Waviness (n) Regularity



(o) Fineness (p) Friction (q) Friction (norm.)

Figure 6.17: Outputs of the model trained on haptic properties, on the held-out test set with feature selection. In each, the red line shows the trend of line of the output, while the yellow line is an ideal regressor ($y = x$). Top row: scan-dependent features. Second row: tapping-based scan-free features. Lower rows: dragging-based scan-free features. See Table 6.6.

NOFS-CV-TRAIN

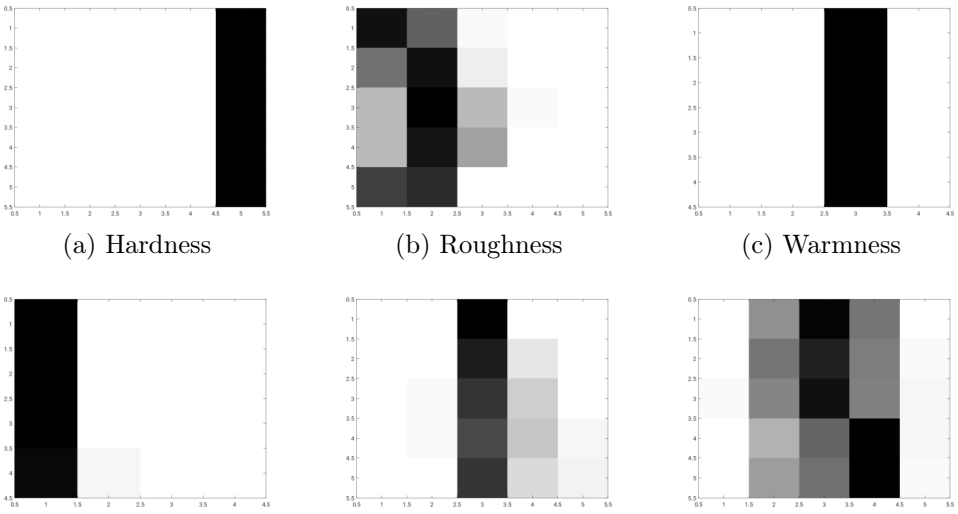


(a) Hardness (b) Roughness (c) Warmness

(d) Stickiness (tooling ball) (e) Stickiness (OptoForce) (f) Stickiness (BioTac)

Figure 6.18: Outputs of the model trained on human ratings, aggregated over cross-validation training sets without feature selection. Rows represent the expected output, while columns identify the actual output. Darker is better. See Table 6.3.

NOFS-CV-TEST

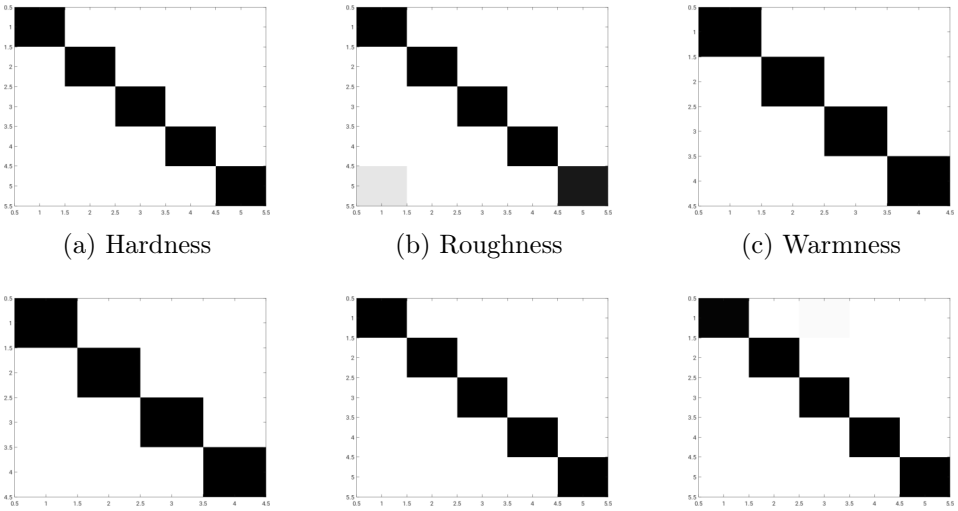


(a) Hardness (b) Roughness (c) Warmness

(d) Stickiness (tooling ball) (e) Stickiness (OptoForce) (f) Stickiness (BioTac)

Figure 6.19: Outputs of the model trained on human ratings, aggregated over cross-validation test sets without feature selection. Rows represent the expected output, while columns identify the actual output. Darker is better. See Table 6.3.

NOFS-ALL-TRAIN

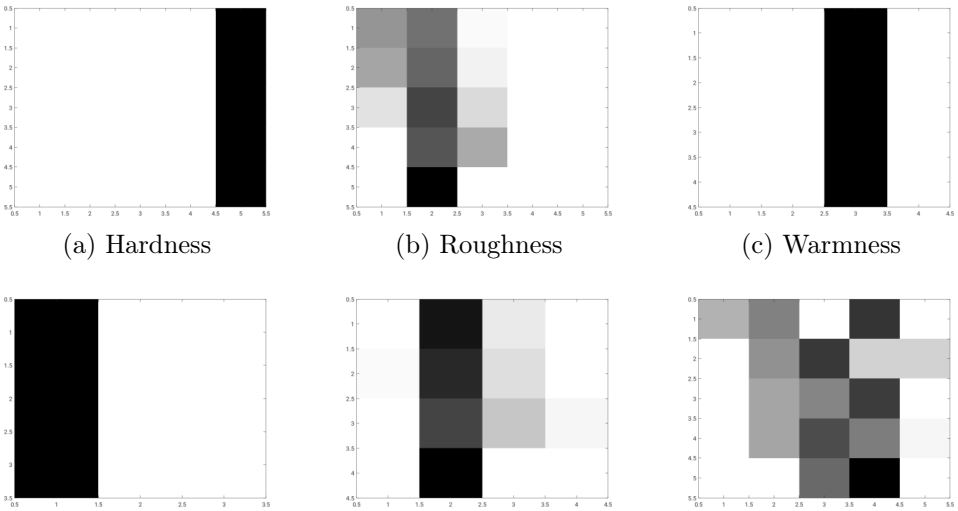


(a) Hardness (b) Roughness (c) Warmness

(d) Stickiness (tooling ball) (e) Stickiness (OptoForce) (f) Stickiness (BioTac)

Figure 6.20: Outputs of the model trained on human ratings, on the whole training set without feature selection. Rows represent the expected output, while columns identify the actual output. Darker is better. See Table 6.4.

NOFS-ALL-TEST

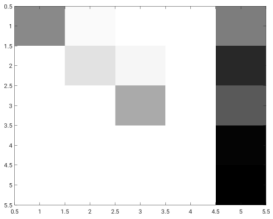


(a) Hardness (b) Roughness (c) Warmness

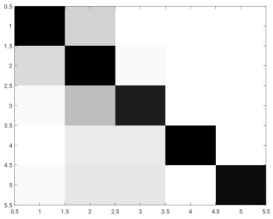
(d) Stickiness (tooling ball) (e) Stickiness (OptoForce) (f) Stickiness (BioTac)

Figure 6.21: Outputs of the model trained on human ratings, on the held-out test set without feature selection. Rows represent the expected output, while columns identify the actual output. Darker is better. See Table 6.4.

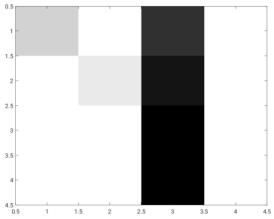
FS-CV-TRAIN



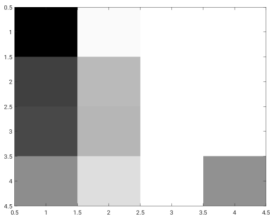
(a) Hardness



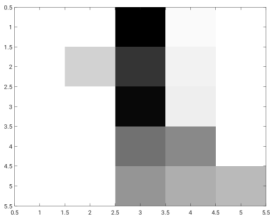
(b) Roughness



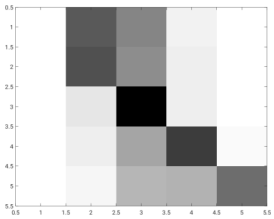
(c) Warmness



(d) Stickiness (tooling ball)



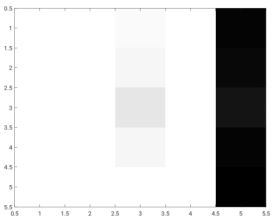
(e) Stickiness (OptoForce)



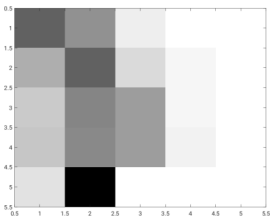
(f) Stickiness (BioTac)

Figure 6.22: Outputs of the model trained on human ratings, aggregated over cross-validation training sets with feature selection. Rows represent the expected output, while columns identify the actual output. Darker is better. See Table 6.5.

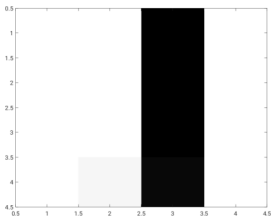
FS-CV-TEST



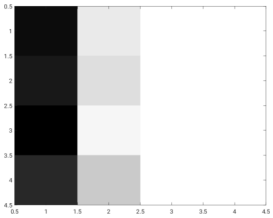
(a) Hardness



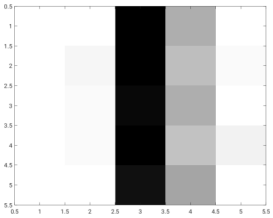
(b) Roughness



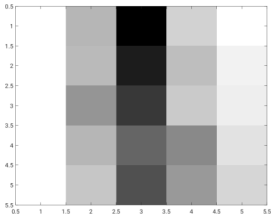
(c) Warmness



(d) Stickiness (tooling ball)



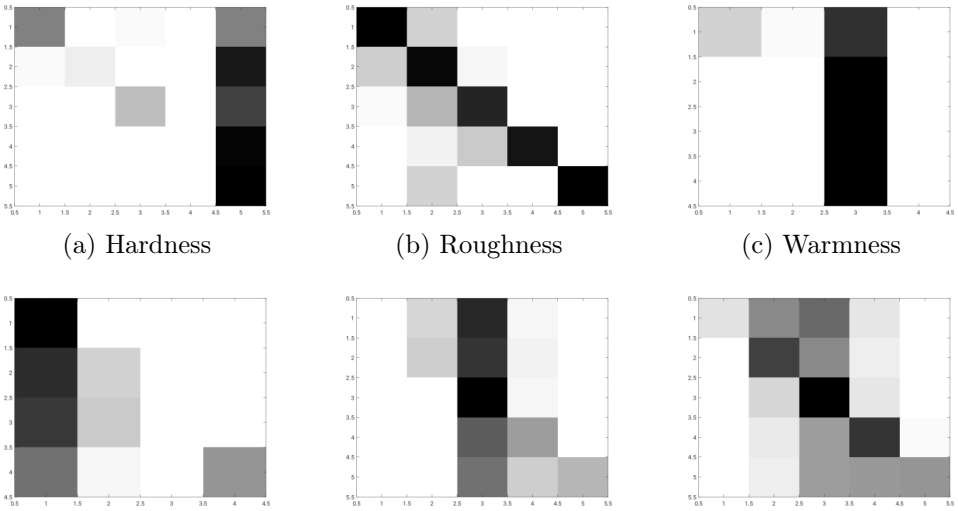
(e) Stickiness (OptoForce)



(f) Stickiness (BioTac)

Figure 6.23: Outputs of the model trained on human ratings, aggregated over cross-validation test sets with feature selection. Rows represent the expected output, while columns identify the actual output. Darker is better. See Table 6.5.

FS-ALL-TRAIN

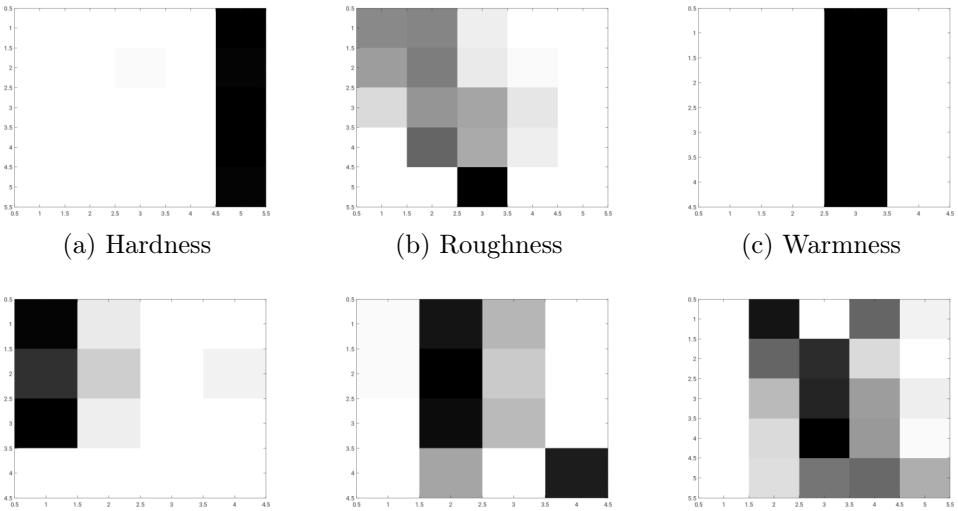


(a) Hardness (b) Roughness (c) Warmness

(d) Stickiness (tooling ball) (e) Stickiness (OptoForce) (f) Stickiness (BioTac)

Figure 6.24: Outputs of the model trained on human ratings, on the whole training set with feature selection. Rows represent the expected output, while columns identify the actual output. Darker is better. See Table 6.6.

FS-ALL-TEST



(a) Hardness (b) Roughness (c) Warmness

(d) Stickiness (tooling ball) (e) Stickiness (OptoForce) (f) Stickiness (BioTac)

Figure 6.25: Outputs of the model trained on human ratings, on the held-out test set with feature selection. Rows represent the expected output, while columns identify the actual output. Darker is better. See Table 6.6.

6.4.2. Analysis

Taking stock of these tables and figures, the machine learning performance is not as might be desired for a practical system to be deployed on a robot, and overfitting has clearly occurred in all cases. This problem occurred despite taking countermeasures against overfitting, including feature selection and adjusting hyperparameters in cross-validation. Since there are multiple images of each surface, we suspect that the high training set accuracy reflects the model memorizing the identity of each surface in the training set, but failing to fully generalize over the association between visual appearance and haptic properties. Strategies for combating this effect may involve more training data, as well as different network structures that emphasize patterns in the images that correspond to material properties rather than identity.

Comparing the results with and without feature selection, it's clear that feature selection does reduce overfitting on the training set and improve test set performance during cross-validation, but the performance on the held-out test set is not improved or even degraded.

Despite the regression accuracies being low, many of the significance measures are well below the $\alpha = 0.003$ level that we set. Note that in general our results show co-occurrence of smaller significance values and higher trend line slopes (closer to an ideal regressor). In the results including feature selection, the best properties to predict during cross-validation (Table 6.5) were the friction coefficient (Figs. 6.14c and 6.15c), MFCC #4 (Figs. 6.14g and 6.15g), and waviness (Figs. 6.14m and 6.15m). But on the held-out test set (Table 6.6), a different MFCC, #1, stands out (Figs. 6.16f and 6.17f), as well as spikiness (Figs. 6.16k and 6.17k). With feature selection, we see much the same pattern, except that the held-out test set correlations are lower.

The classification results seem to be overfit as well, though feature selection helps to reduce this. One problem with the classification task is that our data is highly unbalanced: for example, many surfaces have a hardness value of 5 (very hard), while even more have a

warmness value of 3 (neutral), so it is easy for a classifier to simply learn the mode of the training set and ignore the patterns in the data. This unbalancing includes surfaces such as fabrics, which may have the appearance of being soft, but in our data collection were often secured on top of a hard table, affecting the hardness rating. We do have soft-rated fabrics in the database, such as couches, but the large number of hard-backed fabric samples (especially from the Penn Haptic Texture Toolkit) may confuse the classifier.

Comparison with Human Subjects as Baseline

It is instructive to compare the results of the human subjects and machine learning studies. After all, we humans generally consider ourselves to be good at anticipating the properties of the haptic surfaces around us, before actually touching them. However, the human-subject study clearly demonstrated that the task is actually quite difficult. Subjects given access only to images of the surfaces agreed with the Proton operator relatively rarely. Table 6.7 on the following page shows the participant ratings evaluated against the Proton operator's ratings on the same metrics as the SVM classifiers. Against these numbers, the machine learning methods match human performance.

Rating	%	p	P	R	MAE
Hardness	9.00	$1.05 * 10^{-22}$	0.210	0.190	1.40
Roughness	29.0	$6.94 * 10^{-18}$	0.340	0.280	0.95
Warmness	71.0	$1.19 * 10^{-7}$	0.290	0.250	0.35
Stickiness (median)	31.0	$2.78 * 10^{-17}$	0.310	0.250	1.00
Human mean	35.0	$< 1 * 10^{-100}$	0.280	0.240	0.93

Rating	%	p	P	R	MAE
Hardness	56.0	1.00	0.200	0.189	0.694
Roughness	36.5	0.178	0.266	0.334	0.751
Warmness	80.7	0.125	0.249	0.405	0.239
Stickiness (tooling ball)	63.6	0.0549	0.385	0.356	0.495
Stickiness (OptoForce)	33.9	0.00193	0.354	0.308	0.870
Stickiness (BioTac)	39.0	0.388	0.232	0.265	0.779
Machine mean	51.6	0.00590	0.281	0.310	0.638

Table 6.7: Machine learning algorithm performance metrics applied to the human ratings from the Amazon Mechanical Turk study (top), with the machine learning results from Table 6.6 repeated for comparison (bottom). The last row of each section shows average performance (all values are the arithmetic mean, except for p which is combined using Fisher's method.)

6.5. Summary

In this chapter, we have presented our work toward designing machine learning pipelines to infer haptic properties and adjectival ratings from visual input, against a baseline of humans doing the same task. Though the accuracies we achieved here are relatively low, this is one of the first attempts to use visual images for regression of haptic properties, so better results are to be expected in the future.

6.5.1. Future Work

Plenty of work remains to improve upon the results presented here and make more progress on inferring the relationship between vision and haptics from data. The first limiting factor in machine learning experiments is always lack of data, so it is important to continue collecting data with the Proton or other devices, and publish datasets so that haptics researchers can collaborate on analysis. Datasets should be deliberately balanced in terms of types of materials and their haptic properties, and avoid deceptive materials (such as the hard-backed fabrics in our database). Besides balancing, the metric for evaluating performance is a bit

up in the air. In this chapter, we used several different metrics, but they sometimes conflict: for example, a classifier can have a high percentage of correct answers but low precision and a poor significance level. We may need a more sophisticated performance metric, such as a fuzzy inference model [93].

Also, the machine learning strategy applied here could be adapted to use a different network architecture or fine-tune using a different image dataset. Zhang et al. [94] recently surveyed several likely methods, including their novel Deep-TEN network, and a few currently available material image datasets (including MINC and KTH-TIPS2 which we have covered in this chapter). One limitation is that the network we use has not been trained for the regression task or on our data: we use a network trained on similar data that is designed for material type classification, and we assume that the features it learns are useful for material property understanding in general. We did attempt to train deep networks on our data and directly to the regression objective, but without success. This problem is likely due to the relatively small size of our dataset when compared to other deep learning projects.

CHAPTER 7 : CONCLUSION

Robotic haptic sensing is a rapidly evolving field with important applications for mobile robots (including self-driving cars, household assistants, and search-and-rescue/drone operations), robotic surgery, virtual reality, and others. Many applications involve either deducing the properties of surfaces in the environment or reproducing these properties for human users (or both). Both tasks require understanding the variations in different properties of surfaces, as well as the ways that those properties manifest in the haptic feeling and visual appearance of surfaces. Humans and other animals possess a wealth of high-fidelity visual and haptic sensors, while robots typically have one or two cameras and unimpressive haptic sensors – usually discrete rather than distributed like organic skin, and focused on only one modality of touch (e.g., force or vibration).

The research problem is made more difficult by a number of conditions unique to the field of haptics that make sharing data more difficult. First, the haptics community has not yet coalesced around a common set of sensors, so any attempt to combine databases runs into the issue of comparing data from different end-effectors and sensing modalities (e.g., one end-effector with a pressure sensor and one with accelerometers). Furthermore, the data gathered in any modality depends strongly on the conditions of contact (what we have called "scan-time parameters" in this thesis) – similar to the way images and video change appearance depending on ambient lighting, but we don't yet know how to adjust haptic data for scan-time parameters as easily as brightening or white-balancing an image. Finally, it is labor-intensive to generate haptic data; there is no ready source that occurs as a byproduct of our society, as there is with images and video uploaded to social media (perhaps this will change with the rise of haptic gaming accessories). Nonetheless, collaboration of open datasets in related fields, such as computer vision, has led to such success that we believe

it is time to work towards overcoming these obstacles, so that we can try it out for haptics research as well.

7.1. Contributions

In this thesis we developed a multimodal sensing device, tying together vision and several haptic percepts, in order to provide a model for multimodal data collection and create a useful large-scale surface interaction database. Having collected a database we presented efforts toward applying machine learning to anticipate haptic properties from visual input.

7.1.1. The Proton Pack

The Proton Pack was designed to be portable, self-contained and usable by one human operator (though it is quite useful to have an assistant). This goal dictated many elements of the hardware and software designs. Unlike earlier instruments which may use a bench power supply or a ceiling-mounted localization system, we made it possible to carry all necessary support components (battery, computation, data storage, and sensor support circuitry) in a backpack and used the onboard camera for self-localization.

Three interchangeable end-effectors are provided, which range in sensing modality and expense. They are an unadorned steel tooling ball, an OptoForce (a mid-range stiff rubber three-axis force sensor), and a BioTac (a high-end biomimetic fingertip sensor). These end-effectors are "parked" on the handheld apparatus as well and may be swapped using a quick-release clamp and slip-fit system which ensures repeatable positioning. Each end-effector features accelerometers close to the point of surface contact, and regardless of end-effector three additional sensing modalities are provided by a ATI Mini40 six-axis force/torque sensor, IMU and microphone.

As for visual sensing, an overhead-mounted RGB camera provides continuous video of the surface, which is also used to localize the Proton using manually placed fiducial markers. Additionally, a Structure Sensor measures depth using structured light.

Custom software running on the computer in the backpack interfaces with all of the sensors in real time (assisted by a daughterboard with a Teensy 3.1 microprocessor) and presents a user interface to the human operator, in the form of a web page that can be loaded on a smartphone for portable operation.

7.1.2. Surface Discrimination

We conducted several surface discrimination experiments before embarking on large-scale data collection, in order to prove the efficacy of the Proton and help with calibration. All of these experiments used surfaces from the Penn Haptics Texture Toolkit [48].

An early experiment [2] tested classification of five different surfaces. Here the design of the end-effectors was not yet finalized, and we were still using a fixed Vicon localization system. The classification was done by calculating features of the vibration data, using methods pioneered by Romano and Kuchenbecker [19], and it showed that the Proton data was useful but needed improvement. We used the lessons from this experiment to improve the end-effector design, including replacing the accelerometers and moving them closer to the contact point. In the next experiment [3] we also settled on an algorithm for localizing the Proton position from the fiducial markers using the RGB camera frames.

Further experimentation explored characteristics of haptic surface interaction data. We related our work to similar research [4, 5, 11] at the Technical University of Munich in a study [6] that examined acceleration features designed to be independent of "scan-time parameters" (i.e., the speed of end-effector dragging and the magnitude of the normal force). We replicated that lab's results using published data in order to ensure an accurate implementation of the described features, then verified their independence from scan-time parameter using our own data (since the Proton can actually measure those parameters). We found that contrary to assumptions, these "scan-free features" worked better for surface classification, at least on a limited set of surfaces.

7.1.3. Dataset

As a prelude to large-scale data collection, a small experiment [7] suggested that approximately one minute of data should be enough to characterize surface features. This was the duration we used in our data collection, collecting about one minute of data per end-effector for each surface (though some surface/end-effector combinations were skipped due to concerns about damaging one or the other). We surveyed surfaces at the University of Pennsylvania, the Technical University of Munich, the Material ConneXion [72] exhibit at Princeton University, as well as objects collected at the Max Planck Institute for Intelligent Systems and in two previous databases, the Penn Haptic Texture Toolkit [48] and the YCB object set [68].

The dataset will be available online for other members of the haptics community to use in their research, federate with other data, and take as inspiration for further data collection projects.

7.1.4. Machine Learning

Our investigation of machine learning on the large-scale dataset has revealed mixed results thus far. We used a deep neural network that was pre-trained to categorize close-up images of materials, extracting its most abstract feature representations of the images in our dataset and using that to train support vector machines to output human ratings of surface properties as well as quantitative measures generated from the force and vibration data, but the achieved accuracies were low. This may reflect on the data quality, the feasibility of the problem, or the particular approach we chose. However, we also studied the performance of humans on estimating the property ratings, given the same images of a subset of the surfaces, and their accuracy was similarly low, indicating that the problem of visually estimating surface properties from close-ups without context is a difficult one.

7.2. Future Directions

As with many projects, this thesis is both a culmination and a starting point. There are many open directions for future research, in all the areas that we have touched on: multimodal sensing, large-scale data collection, and machine learning. We hope that research will continue on these questions, using the data we have presented or new data, whether collected with an upgraded Proton or an entirely different instrument.

7.2.1. Multimodal Sensing

The Proton sports a smattering of sensors which, together, approximate some of the sensing modalities that humans use to perceive surfaces and their texture in the world. We tried to select haptic sensors that span a broad range in terms of fidelity and haptic properties. However, the field is still young and many haptic sensors are available in the literature: for example, the TakkTile sensor [95] which measures contact pressure using barometers, the GelSight [15] which visually observes contact force via the deformation of a rubber sphere, successors to the BioTac sensor from SynTouch, artificial robotic skin [10] and others.

The structure of the Proton hardware and software allows expansion with different types of sensors. Thanks to the interchangeable end-effectors and modular software design, new sensors can be added with a small amount of engineering effort. This would allow the dataset to be extended with new materials recorded in additional parallel modalities. In fact, we did not use all of the data collected by the Proton. There are future research opportunities in analyzing the audio and depth data included in our released dataset, as well as the tactile data from the end-effectors that directly contacted the surfaces (OptoForce and BioTac).

Another issue in data collection is selection of surfaces. Given the relatively scarcity of haptic datasets, there is not yet a generally accepted formulation of the dimensions of haptic sensation (for example, we assume that hardness, roughness, slipperiness, stickiness, and temperature are among the salient dimensions, but there are likely others). This need was the subject of a keynote speech at Haptics Symposium 2018 [96]. Future data collectors should

identify salient haptic features and intentionally select surfaces with adequate representation in different regions of the feature space. To ensure that data is collected with sufficient coverage of the scan-time-parameter space (i.e., wide ranges of applied forces and motion speeds), the Proton could analyze a subset of data in real-time and apply an algorithm inspired by our investigation of appropriate episode length (Sec. 5.2), and provide feedback to the human operator.

7.2.2. Data Sharing

Researchers interested in current linguistic trends can mine Twitter, while there is a flood of image and video data available on public sharing platforms such as Flickr and YouTube. Unfortunately, there is no similar ready source of haptic data. Perhaps with the coming rise of consumer virtual reality devices, which will eventually have ubiquitous haptic attachments, passive data collection from the masses will become possible. Until then, the best way to ensure that large datasets are available for research is to collaborate and federate our data. It's important to settle on common data formats and processing strategies so that data collected by different researchers can be federated and avoid starting from scratch with each new project.

7.2.3. Machine Learning

Several machine learning results have been presented in this thesis. At first, we used traditional techniques, including manual feature engineering and SVMs, to classify small sets of surfaces. These experiments verified the utility of the Proton as a device for gathering haptic data. After collecting a large dataset, we presented work toward cross-modal machine learning to deduce haptic properties from visual input. There is much research available to extend this work and to apply machine learning to visuo-haptic data in different ways. For example, there are many ways that visual input can be confusing to a haptic observer: reflective surfaces, shadows, and paint can create visual texture where there is no textural discontinuity. Machine learning could be applied to the problem of removing shadows and

reflections, or determining when something is painted to look like a different material, and use context clues to infer the actual underlying material.

Another open question is cross-modal learning outside the vision/vibration binary. In the Proton dataset, we have many surfaces with vibration data collected using end-effectors with different frictional properties, as well as the two point-of-contact sensors. The OptoForce sensor is uncalibrated and the BioTac does not directly measure force and torque, but since there is synchronized data available from the Mini40, it would be possible to calibrate them. Additionally, translating perceptions between sensors could be interesting. Given a BioTac recording, how does the surface feel to an OptoForce end-effector, and vice versa? The answer could help robots with different end-effectors compare their experiences.

APPENDIX A : CALIBRATION

Before conducting any experiments, we designed and executed a calibration procedure to compensate for differences between the actual Proton Pack and its CAD model, and to zero the ATI Mini40 six-axis force/torque sensor.

Portions of this appendix are adapted with permission from a paper that was published in the proceedings of MFI [2] (© 2016 IEEE).

To allow researchers to interpret the data recorded from surface interaction, the Proton must be able to track its own motion. This requirement stems from a desire to know the speed at which the end-effector scans across the surface under study, as well as decomposing the force and torque measured by the Mini40 into tangential and normal components – which means interpreting those readings in the world frame. We included sensors in the Proton to allow this tracking to be performed onboard, similar to the way mobile robots track their position using dead reckoning and landmarks in their environment. However, in the initial calibration and validation experiments, an external Vicon motion tracking system was used in place of the onboard IMU and cameras to ensure accuracy and provide a baseline for subsequent implementation of standalone motion tracking.

The goal of the calibration procedure is to unify the various coordinate frames on the sensing rig: the IMU, accelerometers, Mini40, OptoForce, BioTac, and cameras all take measurements that are referenced to their individual coordinate frame's origin and orientation, but they all need reference a common frame for analysis. One way to do this alignment would be open-loop calibration, using our device's CAD model to estimate the translation and rotation offsets. These estimates might be reasonable, but they do not account for small variations in manufacturing and assembly. Furthermore the Vicon coordinate frame (not

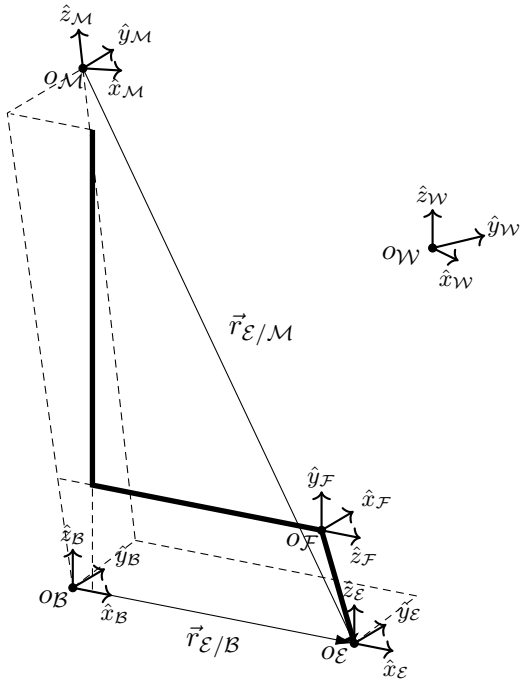


Figure A.1: Coordinate frames (repeated from Fig. 3.5 for reference)

shown in earlier photos such as Figs. 2.3 and 2.4) is attached to the sensing rig at an arbitrary location, and we must recover this transformation in order to validate the motion tracking.

Notation in this appendix uses a superscript on vectors and transformation matrices to denote the frame in which they are expressed; e.g., the origin of frame $\mathcal{A}$ expressed in frame $\mathcal{B}$ is $\vec{o}_{\mathcal{A}}^{\mathcal{B}}$. For a vector starting at the origin of frame $\mathcal{A}$ and ending at the origin of $\mathcal{B}$, expressed in a third frame $\mathcal{C}$, a second subscript is used: $\vec{v}_{\mathcal{B}/\mathcal{A}}^{\mathcal{C}}$. $\mathbf{R}$ refers to a 3D rotation matrix (3×3), while $\mathbf{H}$ represents a *homogeneous transformation*, i.e., a 4×4 representing both the rotation and translation from one frame to another: $\mathbf{H}_{\mathcal{A}}^{\mathcal{B}} = \begin{bmatrix} \mathbf{R}_{\mathcal{A}}^{\mathcal{B}} & \vec{o}_{\mathcal{A}}^{\mathcal{B}} \\ \vec{0} & 1 \end{bmatrix}$. Such a transformation is applied as follows:

$$\mathbf{H}_{\mathcal{A}}^{\mathcal{B}} \begin{bmatrix} \vec{v}^{\,\mathcal{A}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathcal{A}}^{\mathcal{B}} & \vec{o}_{\mathcal{A}}^{\mathcal{B}} \\ \vec{0} & 1 \end{bmatrix} \begin{bmatrix} \vec{v}^{\,\mathcal{A}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathcal{A}}^{\mathcal{B}} \vec{v}^{\,\mathcal{A}} + \vec{o}_{\mathcal{A}}^{\mathcal{B}} \\ 1 \end{bmatrix} = \begin{bmatrix} \vec{v}^{\,\mathcal{B}} \\ 1 \end{bmatrix} \tag{A.1}$$

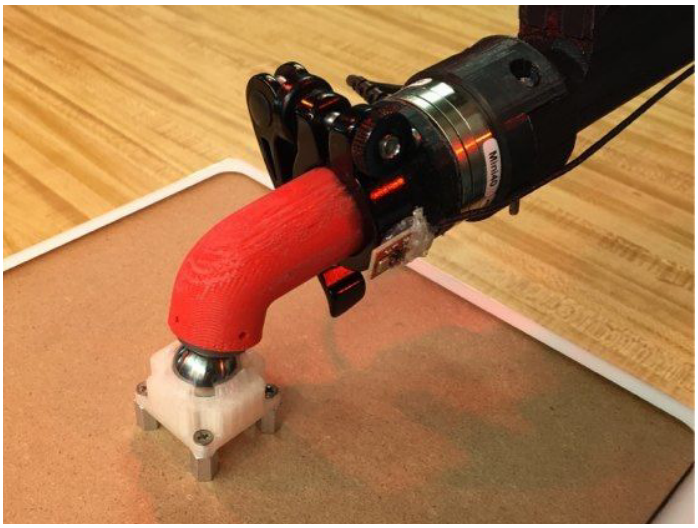


Figure A.2: End-effector fixed in place for Vicon motion tracking calibration. Though we later changed the size of the tooling ball, each end-effector was designed to put the origin of its coordinate frame in the same position.

A.1. Vicon Motion Tracking Calibration

The Vicon motion tracking system consists of an array of infrared projectors and cameras affixed to scaffolding above and around the workspace, plus supporting computer hardware. (A human-assisted calibration routine allows the cameras to learn their relative positions.) To track an object, operators attach infrared reflective markers, ideally at least four in an asymmetric, non-coplanar configuration, and model this configuration in the Vicon software through an interactive interface. Each camera projects a ray from itself through each marker in its field of view, and the markers are deduced to reside at the intersections of those rays. The Vicon software then automatically aligns constellations of markers with the pre-specified object models, discards anomalies and filters out noise. Position and orientation of each tracked model is reported at up to 100 Hz.

The first calibration procedure determines the vector from the marker frame to the end-effector frame, $\vec{r}^{\,\mathcal{M}}_{\mathcal{E}/\mathcal{M}}$. Because these markers are placed and modeled in software by hand, we cannot rely on CAD estimates.

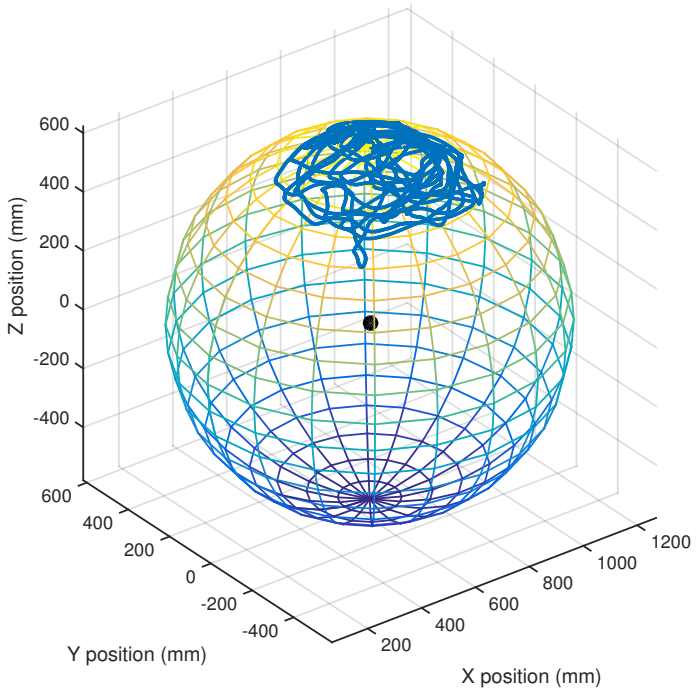


Figure A.3: Spherical fit to data from Vicon motion tracking calibration.

We fix the tooling ball end-effector in place using a socket (see Fig. A.2 on page 159) and rotate the Proton in all directions while recording $\mathbf{H}_{\mathcal{M}}^{\mathcal{W}}(t)$. Each world-frame position measurement will be at a constant distance from the end-effector, lying on a sphere whose radius is the distance from the origin of $\mathcal{M}$ to the origin of $\mathcal{E}$:

$$\vec{o}_{\mathcal{M}}^{\mathcal{W}}(t) = \vec{o}_{\mathcal{E}}^{\mathcal{W}} + \vec{r}_{\mathcal{M}/\mathcal{E}}^{\mathcal{W}}(t) \qquad \forall t \tag{A.2}$$

Furthermore, no matter the orientation of the Proton, the vector we seek remains constant with respect to the moving frames. We fit a sphere to the time-varying $\vec{o}_{\mathcal{M}}^{\mathcal{W}}$ measurements, using a closed-form solution for least-squares minimization [97] and RANdom SAmple Consensus (RANSAC) to exclude outliers. This fit, shown in Fig. A.3, recovered both the center of the sphere, $\vec{o}_{\mathcal{E}}^{\mathcal{W}}$, and its radius, $||\vec{r}_{\mathcal{E}/\mathcal{M}}||$, with 68% inliers and RMS error of 8.4 mm. From

there we can recover the vector:

$$\vec{r}^{\,\mathcal{M}}_{\mathcal{E}/\mathcal{M}} = \mathbf{H}^{\mathcal{M}}_{\mathcal{W}}(t)\ \vec{r}^{\,\mathcal{W}}_{\mathcal{E}/\mathcal{M}}(t) \qquad \forall t \tag{A.3}$$

We average the results of $\mathbf{H}^{\mathcal{M}}_{\mathcal{W}}(t)\ \vec{r}^{\,\mathcal{W}}_{\mathcal{E}/\mathcal{M}}(t)$ over the calibration period to estimate $\vec{r}^{\,\mathcal{M}}_{\mathcal{E}/\mathcal{M}}$, as given in Table A.1 on page 165.

A.2. Gravity Compensation

We want to measure the normal force and torque exerted by the surface on the end-effector. However, the end-effector itself has mass which exerts an additional force and torque, due to gravity. In a stationary system such as a desk scale, this requires only one calibration measurement (i.e., "tare") to zero out. However, our Mini40 moves with the handheld rig, so we must measure the mass and center-of-mass (CoM) of the end-effector, as well as know the world-frame orientation of the sensor, in order to calibrate out the influence of the end-effector's mass.

The second calibration recovers this bias, the orientation of the body frame with respect to the marker frame, and the mass and center of mass of each end-effector (which are used to perform gravity compensation). We record position, orientation, force, and torque while slowly rotating the Proton in all directions in free space – that is, without the end-effector contacting anything, as pictured in Fig. A.4 on the following page. The procedure was repeated twice for each end-effector, so fit values and errors in this section are aggregated over the six runs. Under these conditions, the expected force and torque measurements can be described by these simple equations (see the free-body diagram in Fig. A.5 on the next page):

$$\vec{F}^{\,\mathcal{B}}_{R}(t) = \vec{b}^{\,\mathcal{B}}_{F} + m\vec{g}^{\,\mathcal{B}}(t) \tag{A.4}$$

$$\vec{\tau}^{\,\mathcal{B}}_{R}(t) = \vec{b}^{\,\mathcal{B}}_{\tau} + \vec{r}^{\,\mathcal{B}} \times m\vec{g}^{\,\mathcal{B}}(t) \tag{A.5}$$

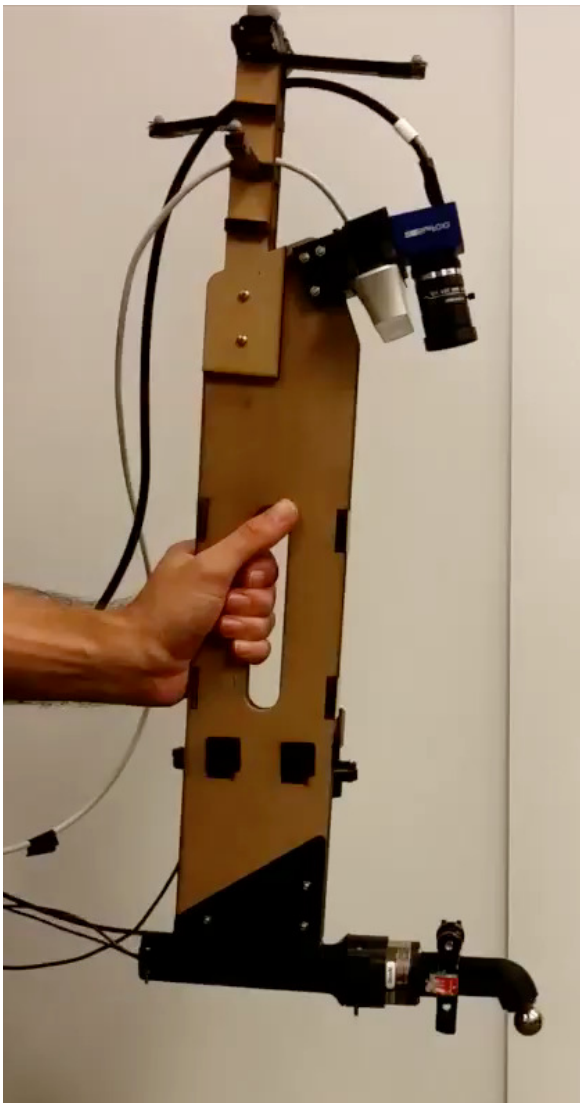


Figure A.4: An operator holding the Proton as part of the gravity compensation calibration procedure. The only force measured by the Mini40 is the force of gravity on the end-effector.

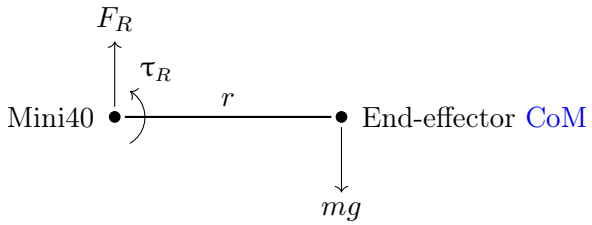


Figure A.5: Free-body diagram of the force/torque sensor and end-effector during gravity compensation calibration. The Mini40 measures the reactions $\vec{F}_R$ and $\vec{\tau}_R$. Sensor bias is not shown. Reprinted from [2] © 2016 IEEE.

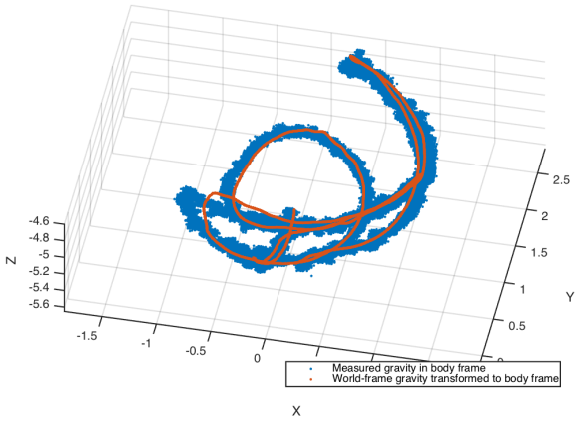


Figure A.6: Spherical fit to force data from gravity compensation calibration.

The total force on the sensor (A.4) comes from a constant offset $\vec{b}_F$, the sensor's internal force bias, plus the force of gravity, $m\vec{g}(t)$; here, m is the mass of the end-effector, and $\vec{g}(t)$ is the constant-magnitude gravitational acceleration vector, whose direction varies over time. Similarly, the total torque (A.5) comes from the internal torque bias $\vec{b}_\tau$ plus the torque contributed by gravity acting on the mass of the end-effector, $\vec{r} \times m\vec{g}(t)$, where $\vec{r}$ is the constant vector from the sensor to the end-effector's center of mass.

(A.4) has the form of another sphere equation, so fitting a sphere to the force data gives the solution for $\vec{b}_F$ as the center and m as the radius. The fit, shown in Fig. A.6, has an average of 93.6% inliers ($\pm$8.5 percentage points std) and RMS error of $-1.079\,\mathrm{N}$ ($\pm 0.013\,\mathrm{N}$ std); numerical values are given in Table A.1 on page 165. Then we compare gravity vectors to determine the relative orientation of the two frames. In the body frame, the direction of gravity is given by the force measurement minus the force bias. In the world frame, we know that gravity points along the negative $\hat{z}$-axis, so we transform this vector into the marker frame using the Vicon measurements. We solve for the rotation between those two sets of gravity vectors using the singular value decomposition (SVD) method of [98]. This result gives the desired transformation between $\mathcal{M}$ and $\mathcal{B}$.

To extract the torque bias, we substitute (A.4) into (A.5):

$$\vec{\tau}_R(t) = \vec{b}_\tau + \vec{r} \times (\vec{F}_R(t) - \vec{b}_F) \tag{A.6}$$

Then we apply robust least squares estimation to solve (A.6) for $\vec{b}_\tau$ and $\vec{r}$. The fit values, which have an average σ of 0.00095 Nm ($\pm$0.00036 Nm std), are shown in Table A.1 on the following page.

After performing these two calibrations, we use the results to construct two homogeneous transformations: $\mathbf{H}_{\mathcal{B}}^{\mathcal{M}}$, which gives the pose of the body frame in the marker frame, and $\mathbf{H}_{\mathcal{E}}^{\mathcal{B}}$, which gives the pose of the end-effector frame in the body frame.

We have free choice of the origin of $\mathcal{B}$. To reduce the number of unknowns, we specify that the origin of $\mathcal{E}$ is on the x-axis of $\mathcal{B}$ with no rotation between those two frames, and that the origin of $\mathcal{B}$ is in the yz-plane of $\mathcal{M}$ (refer back to Fig. A.1 on page 158). Then the combined transformation $\mathbf{H}_{\mathcal{B}}^{\mathcal{M}}\mathbf{H}_{\mathcal{E}}^{\mathcal{B}}$ has $\vec{r}_{\mathcal{E}/\mathcal{M}}^{\mathcal{M}}$ as its translational component, leading to the constraint:

$$\begin{bmatrix} & \mathbf{R}_{\mathcal{B}}^{\mathcal{M}} & & 0 \\ & & & y \\ & & & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} & I_3 & & x \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \vec{r}_{\mathcal{E}/\mathcal{M}}^{\mathcal{M}} \\ 1 \end{bmatrix} \tag{A.7}$$

Solving this equation symbolically yields the unknown matrix elements x, y, and z.

Table A.1 shows a summary of the calibration results. The top section shows the results from the gravity compensation calibration, and the lower two sections show results from the Vicon calibration. As expected, all three end-effectors have similar masses, but the tooling ball is the heaviest and its CoM is farthest from the Mini40. These values are plausible though they do not match the CAD model exactly due to fabrication variations.

Parameter	Symbol	Value
Vector from $\mathcal{M}$ to $\mathcal{E}$	$\vec{r}^{\,\mathcal{M}}_{\mathcal{E}/\mathcal{M}}$	$[252.2, 30.34, -545.1]$ mm
Rotation from $\mathcal{M}$ to $\mathcal{B}$	$\mathbf{R}^{\mathcal{B}}_{\mathcal{M}}$	$\begin{bmatrix} 0.991 & -0.024 & -0.130 \\ 0.017 & 0.998 & -0.058 \\ 0.132 & 0.055 & 0.990 \end{bmatrix}$
Vector from camera frame to $\mathcal{E}$	$\vec{r}^{\,\mathcal{M}}_{\mathcal{E}/\mathcal{C}}$	$[0.000, 204.4, 353.7]$ mm
Rotation from camera frame to $\mathcal{B}$	$\mathbf{R}^{\mathcal{B}}_{\mathcal{C}}$	$\begin{bmatrix} 0.000 & -0.969 & -0.249 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 0.249 & 0.969 \end{bmatrix}$
Mini40 force bias	$\vec{b}^{\,\mathcal{F}}_{F}$	$[\ 1.552, -4.345, -0.182]$ N
Mini40 torque bias	$\vec{b}^{\,\mathcal{F}}_{\tau}$	$[-0.106,\ \ 0.037, -0.029]$ Nm
Tooling ball end-effector mass	m_t	150.3 g
Tooling ball end-effector CoM	$\bar{r}^{\,\mathcal{F}}_{t}$	$[-2.29, -3.01, 34.8]$ mm
OptoForce end-effector mass	m_o	110.2 g
OptoForce end-effector CoM	$\bar{r}^{\,\mathcal{F}}_{o}$	$[-5.56, 1.41, 21.8]$ mm
BioTac end-effector mass	m_b	113.8 g
BioTac end-effector CoM	$\bar{r}^{\,\mathcal{F}}_{b}$	$[-3.66, 1.86, 21.0]$ mm

Table A.1: Calibration results. Reprinted from [2] © 2016 IEEE.

A.3. Summary

Calibration is incredibly important in the development of a new device. Sensor readings are meaningless without a means to express them in the correct frame of reference and at the appropriate scale. Often, these transformations are sensitive to small variations in manufacturing tolerances and must be adjusted to each particular device for greatest accuracy. In our case, we used a combination of estimates from CAD drawings, manufacturer-sourced calibration, and our own empirical measurements to align the readings from all the sensors.

This chapter explained the methods for each of the software-assisted calibrations that were performed in the course of preparing the Proton for effective data collection. The calibrations account for the manual placement of the reflective markers for Vicon tracking, the intrinsic mass distribution of the three end-effectors, and the imprecise mounting of the Bluefox camera. In addition, an engineering oversight was discovered quite late, and a software

solution implemented to minimize its impact. The numerical results, which are used in the postprocessing code described in the previous chapter, are also given here for reference.

APPENDIX B : SURFACE LIST

This is a list of all surfaces and data contained in the Proton Pack dataset at the time of publication. In the case of any changes, an updated list will always be included in the dataset itself. The surfaces are organized into sections using the Materials in Context [83] categories (we do not have materials in every category). Not all materials were recorded with every end-effector, due to the nature of the materials themselves. We avoided collecting data with the tooling ball when that hard metal end-effector would have damaged the surface (especially when using loaned materials!). Also, when the friction coefficient between the surface and the OptoForce or BioTac sensors was simply too high to gather data (e.g., the end-effector would just "skip" across the surface or threaten to rip off the fragile BioTac skin), those end-effectors were skipped. These are denoted with ✓ and ✗ in the tables below.

During some portions of data collection, we encountered technical issues with various sensors. Specifically, the Bluefox camera has an unreliable automatic white balance and exposure system (see Sec. 3.1.2 on page 32), while the Structure Sensor's driver was unreliable enough that we often deactivated it to avoid delaying data collection. Finally, in a few instances the microphone wiring was faulty. Rather than cut down the list of surfaces, we are still publishing some with incomplete data. These are all marked accordingly with footnotes in the tables.

Several surfaces in this list have the same description (first column). To aid in differentiating them, we also give the location or collection in which the surfaces were found (second column). As discussed in Chapter 5, some of the surface sources are:

- BOLT: material samples from the BOLT project [43]
- LMT: material samples from the LMT database [61] (scanned on location at TU Munich or transported to Penn)
- HaTT: material samples from the Penn Haptic Texture Toolkit [48]
- MPI (Michaela's collection): objects collected at MPI and scanned at Penn
- Princeton Material ConneXion: material samples from the rotating Material ConneXion exhibit in the Princeton University computer science library
- YCB: objects in the YCB set [68]

The other sources refer to building locations on the University of Pennsylvania campus.

B.1. Brick and Tile

Surface	Source	Tooling ball	OptoForce	BioTac
Beige tile	Towne/Skirkanich boundary	✓[2]	✓[1]	✓[1]
Brick	BOLT	✓[1]	✓	✓
Brick 1 (outside)	Towne basement east entrance	✓[2]	✓[1]	✓[1,2]
Brick-like tile	Chem cafe	✓[2]	✓[1]	✓[1]
Busy white tile	Towne/Skirkanich boundary	✓	✗	✗
Ceramique Dawn (Home Depot 183-480)	HaTT	✓[2]	✓[2]	✓[2]
Dark gray floor tile	LRSM	✓	✗	✗

Surface	Source	Tooling ball	OptoForce	BioTac
Floor Tile A	HaTT	✓[2]	✓[2]	✓[2]
Floor Tile B	HaTT	✓	✓	✓
Floor Tile C	HaTT	✓	✓[2]	✓[2]
Floor Tile D	HaTT	✓[2]	✓[2]	✓[2]
Floor Tile E	HaTT	✓[2]	✓[2]	✓
Floor Tile F	HaTT	✓[2]	✓[2]	✓[2]
Floor Tile G	HaTT	✓[2]	✓[2]	✓[2]
Glossy brick	Chem cafe	✓[2]	✓[1]	✓[1]
Gray floor tile	DRL W wing	✓[1]	✓[1]	✓
Gray floor tile	Penn bookstore	✓	✗	✗
Hallway floor 1	Hallway outside Towne 144	✓[2]	✓[1]	✓[1]
Hallway floor 2	Hallway outside Towne 144	✓[2]	✓[1]	✓[1]
Light gray floor tile	LRSM	✓	✗	✗
PMC 7330-03	Princeton Material ConneXion	✓	✓	✓
Patina (Home Depot 182-878)	HaTT	✓[2]	✗	✓[2]
Pink speckled floor tile	LRSM	✓	✗	✗
Porcelain tile	DRL W wing	✓	✓	✓
Red floor tile	DRL W wing	✓	✓[1]	✓[1]
Smooth brick	Vagelos patio	✓[2]	✓[1]	✓[1]
Stone tile (less busy)	Skirkanich 2nd floor	✓	✗	✗
Stone tile (more busy)	Skirkanich 3rd floor	✓	✗	✗
Stone tile 1	Towne sub-basement	✓[2]	✓[1]	✓[1]
Stone tile 2	Towne sub-basement	✓[2]	✓[1]	✓[1]

Surface	Source	Tooling ball	OptoForce	BioTac
Stone tile 3	Towne sub-basement	✓[2]	✓[1]	✓[1]
Stone tile A	HaTT	✓	✓[2]	✓[2]
Stone tile B	HaTT	✓[1]	✓[2]	✓[2]
Tile floor	Towne 144	✓	✓	✓
White floor tile	DRL W wing	✓[1]	✓[1]	✓
White floor tile	Penn bookstore	✓	✗	✗
White pebbled floor tile	DRL lobby	✓	✓	✓
White speckled floor tile	LRSM	✓	✗	✗
White tile	MPI (Michaela's collection)	✓[1]	✓	✓[1]

B.2. Carpet

Surface	Source	Tooling ball	OptoForce	BioTac
Blue rug	Penn bookstore	✓	✗	✗
Brown carpet	DRL lobby	✓	✓	✓
Carpet	DRL classroom	✓	✓	✓
Carpet	LMT	✓	✓	✗
Carpet	LMT	✗	✗	✓
Carpet A	HaTT	✓	✓[2]	✓[2]
Carpet B	HaTT	✓[2]	✓[2]	✓[2]
Carpet C	HaTT	✓[2]	✓[2]	✓[2]
Carpet D	HaTT	✓	✓[2]	✓[2]
Checked rug	Penn bookstore	✓	✗	✗

Surface	Source	Tooling ball	OptoForce	BioTac
Gray carpet	DRL lobby	✓	✓	✓
Lab rug	Towne 144	✓	✓	✓
Orange carpet	Singh center conference room	✓[2]	✓[1]	✓[1]
Pavilion Desert Stone	HaTT	✓[2]	✓[2]	✓[2]
Perfection Smoke Blue	HaTT	✓[2]	✗	✗
Red rug	Penn bookstore	✓	✗	✗
Tile rug	College Hall foyer	✓[1]	✓	✓
Tile rug	LMT	✗	✗	✓
Turquoise rug	Alex's apartment	✓	✓	✓
Wall to wall carpeting	LRSM	✓	✗	✗
Wall-to-wall carpeting	Alex's apartment	✓[2]	✓[1]	✓

B.3. Fabric

Surface	Source	Tooling ball	OptoForce	BioTac
Blanket	HaTT	✗	✓	✓
Blue biking jersey	Penn bookstore	✓	✓	✓
Blue floor cloth	LMT	✓	✓	✓
Blue office chair	Towne 144	✓[2]	✓[1]	✓[1]
CD Sleeve	HaTT	✓[2]	✓[2]	✓[2]
Canvas A	HaTT	✓[2]	✓[2]	✓[2]
Canvas B	HaTT	✓	✓[2]	✓[2]
Canvas C	HaTT	✓[2]	✓	✓[2]

Surface	Source	Tooling ball	OptoForce	BioTac
Canvas tote bag	Penn bookstore	✗	✓	✗
Chair with flower pattern	DRL lobby	✓[1]	✓[1]	✓
Chair with spiral pattern	DRL lobby	✓	✓	✓
Cloth 407	BOLT	✓	✓	✓
Cloth covered foam	BOLT	✓	✓	✓
Denim	HaTT	✓[2]	✓[2]	✓[2]
Felt	HaTT	✓	✓[2]	✓[2]
Flannel	HaTT	✓[2]	✓[2]	✓[2]
Fleece	HaTT	✓[2]	✓[2]	✓[2]
Floor cloth 3	LMT	✓	✓	✓
Flower patterned fabric bench	Hayden Hall 1st floor lounge	✓[2]	✓[2]	✓[2]
Gray office chair	Towne 144	✓[2]	✓[1]	✓[1]
Green couch	LRSM	✓	✗	✗
Green fibrous stuff	LMT	✓	✓[1]	✓
Green office chair	Towne 144	✓[2]	✓[1]	✓[1]
Grey jacket	Penn bookstore	✓	✓	✓
Jeans	LMT	✓	✗	✗
Köper white	LMT	✓	✗	✗
Microfiber Cloth	HaTT	✓[2]	✓[2]	✓[2]
Multicolored felt	LMT	✓	✗	✗
Nylon Bag	HaTT	✓[2]	✓[2]	✓[2]
Nylon Strap	HaTT	✓	✓	✓
Orange foam	BOLT	✓	✓	✓

Surface	Source	Tooling ball	OptoForce	BioTac
PMC 5343-22	Princeton Material ConneXion	✓	✓	✓
PMC 6106-05	Princeton Material ConneXion	✓	✓	✓
PMC 6452-05	Princeton Material ConneXion	✗	✓	✓[1]
PMC 7429-01	Princeton Material ConneXion	✓	✓	✓
PMC 7507-01	Princeton Material ConneXion	✓	✓	✓
Pennant (back)	Penn bookstore	✓	✓	✓
Pennant (front)	Penn bookstore	✓	✗	✗
Pink foam	HaTT	✓[2]	✗	✗
Pink perforated material	LMT	✓	✓	✓
Red felt	LMT	✓	✓	✓
Sheepskin	LMT	✓	✓[1]	✓
Silk A	HaTT	✓	✓[2]	✓[2]
Stiff woven fabric	LMT	✓	✗	✗
Striped fabric chair	Hayden Hall 1st floor lounge	✓[2]	✓[2]	✓[2]
Styrofoam under Velcro loops	Towne 144	✓	✓	✓
Styrofoam under nylon	Towne 144	✓[1]	✓[1]	✓
Textured Cloth	HaTT	✓	✓[2]	✓[2]
Towel	HaTT	✓	✓	✓
Velours-Lederimitat tan	LMT	✓	✗	✗
Velvet	LMT	✓	✗	✗
White felt	LMT	✓	✗	✗
Woven backing	LMT	✓[2]	✓	✓
Yellow felt	BOLT	✓	✓	✓

B.4. Metal

Surface	Source	Tooling ball	OptoForce	BioTac
Aluminum	MPI (Michaela's collection)	✓	✓	✓
Aluminum 1	LMT	✗	✗	✓
Aluminum foil	MPI (Michaela's collection)	✓[2]	✓	✓[2]
Aluminum plate	LMT	✗	✗	✓
Aluminum siding	Singh center lobby	✓[2]	✓[1]	✓[1]
Copper	LMT	✗	✗	✓
Corrugated aluminum	LMT	✓	✗	✗
Door kick plate (reflective metal)	Towne 144	✓	✓	✓
Fine aluminum mesh	LMT	✓	✓	✓
PMC 5946-03	Princeton Material ConneXion	✓	✓	✓
PMC 6800-01	Princeton Material ConneXion	✗	✓	✓
PMC 7560-01	Princeton Material ConneXion	✗	✓	✓
Tool cabinet	Towne 144	✓	✓	✓
Tuna can	YCB	✓	✓	✗
Whiteboard	HaTT	✓[2]	✓[2]	✓[2]

B.5. Painted surfaces

Surface	Source	Tooling ball	OptoForce	BioTac
Cork tile (back)	MPI (Michaela's collection)	✓[2]	✓	✓[2]
Lab desk	Towne 144	✓	✓	✓

Surface	Source	Tooling ball	OptoForce	BioTac
Metal handrail	Singh center stairway	✓[2]	✓[1]	✗
Metal newspaper dispenser	34th &Walnut NE corner	✓	✗	✗
Oscilloscope casing (rough painted metal)	Towne 144	✓	✓	✓
PMC 5842-03	Princeton Material ConneXion	✓	✓	✓[1]
Painted Wood	HaTT	✓	✓[2]	✓[2]
Painted metal floor divider	Fisher 3rd floor landing	✓	✓[1,2]	✓[1,2]
Phantom demo desk	Towne 144	✓	✓	✓
Plaster banister	Skirkanich elevator landing	✓[2]	✓[1]	✓[1]
Radiator top (painted wood)	Hayden Hall entryway	✓[2]	✓[2]	✓[1,2]
Red metal plate	YCB	✓	✓	✓[1]
Scrabble box	Alex's apartment	✓	✗	✓
Windowsill	DRL classroom	✓[1]	✓	✓

B.6. Paper

Surface	Source	Tooling ball	OptoForce	BioTac
Brown cardboard box	Towne 144	✓	✓	✓
Brown tissue paper	MPI (Michaela's collection)	✓[2]	✗	✓[2]
Bubble envelope	HaTT	✓[2]	✗	✓[2]
Bubble mailer	Penn bookstore	✓	✓	✓
Cardboard	HaTT	✓[2]	✓[2]	✓[2]
Cardboard	MPI (Michaela's collection)	✓[2]	✗	✓[2]
Cardboard CD case	MPI (Michaela's collection)	✓[2]	✓	✓[2]

Surface	Source	Tooling ball	OptoForce	BioTac
Cardboard box (peg box box)	YCB	✓[1]	✓[1]	✓
Coffee filter	MPI (Michaela's collection)	✓[1]	✗	✓[2]
Dot Paper	HaTT	✓[2]	✓[2]	✓[2]
Folder	HaTT	✓	✓[2]	✓[2]
Glossy hard book cover (grid pattern)	Towne 144	✗	✓	✓
Glossy soft book cover (blue)	Towne 144	✗	✓	✓
Glossy soft book cover (dark blue)	Towne 144	✗	✓	✗
Glossy soft book cover (dark green)	Towne 144	✗	✓	✓
Glossy soft book cover (green)	Towne 144	✗	✓	✓
Glossy soft book cover (red)	Towne 144	✗	✓	✓
Glossy soft book cover (turquoise)	Towne 144	✗	✓	✓
Green plastic folder	MPI (Michaela's collection)	✓[2]	✗	✓[2]
Greeting card	HaTT	✓[2]	✗	✓[2]
Mailing envelope	MPI (Michaela's collection)	✓	✓	✓
Masking tape	HaTT	✓[2]	✓[2]	✓[2]
PMC 7229-02	Princeton Material ConneXion	✗	✓	✓
PMC 7513-02	Princeton Material ConneXion	✗	✓	✓
Paper Bag	HaTT	✓	✓[2]	✓[2]
Playing Card	HaTT	✓	✓	✓
Red tissue paper	LMT	✗	✓	✓
Resume Paper	HaTT	✓[2]	✓[2]	✓[2]
Rough Paper Plate 1	HaTT	✓[2]	✓[2]	✓[2]
Rough Paper Plate 2	HaTT	✓[1]	✓[2]	✓[2]

Surface	Source	Tooling ball	OptoForce	BioTac
Rough backing	MPI (Michaela's collection)	✓[2]	✗	✓[2]
Rough paper book cover (yellow)	Towne 144	✗	✓	✓
Stiff black cardboard	MPI (Michaela's collection)	✓[2]	✗	✓[1,2]
Sugar box	YCB	✓	✓	✓
Texmet perfoiert	MPI (Michaela's collection)	✓[2]	✗	✓[2]
Textured Paper	HaTT	✓	✓[2]	✓[2]
Tyvek	MPI (Michaela's collection)	✓	✗	✓[2]
Wax paper	MPI (Michaela's collection)	✓[2]	✗	✓[2]
White cardboard box	Towne 144	✓	✓	✓
Woven paper	MPI (Michaela's collection)	✓[2]	✗	✓[2]
"How to bake"	Penn bookstore	✗	✓	✓

B.7. Plastic

Surface	Source	Tooling ball	OptoForce	BioTac
ABS	HaTT	✓	✓[2]	✓[2]
Acrylic	HaTT	✓	✓	✓
Balloon	LMT	✓	✗	✗
Blue checkerboard	LMT	✓	✓	✓
Bubble wrap	HaTT	✓[2]	✗	✓[2]
Chair desk (imitation wood)	Hayden Hall classroom	✓[2]	✓[2]	✓[1,2]
Desk	DRL classroom	✓	✓	✓
Dotted rubber	LMT	✓	✓	✓

Surface	Source	Tooling ball	OptoForce	BioTac
Green film	LMT	✓	✓	✓
Imitation granite table	LRSM	✓	✗	✗
Olive green plastic	LMT	✗	✗	✓
Orange table	Singh center study alcove	✓[2]	✓[1]	✓[1]
Paper slicer	Hayden Hall mailroom	✓[2]	✓[2]	✓[2]
Perfboard	LMT	✓	✓	✓
Plastic table	LRSM	✓	✗	✗
Plastic wood table	Hayden Hall seminar room	✓[2]	✓[2]	✓[2]
PMC 1524-05	Princeton Material ConneXion	✓	✓	✓
PMC 2252-08	Princeton Material ConneXion	✗	✓	✓
PMC 5724-04	Princeton Material ConneXion	✓	✓	✓
PMC 5830-09	Princeton Material ConneXion	✗	✓	✓
PMC 6973-13	Princeton Material ConneXion	✗	✓	✓
PMC 7376-01	Princeton Material ConneXion	✓	✓	✓
PMC 7693-01	Princeton Material ConneXion	✓	✗	✓
Printer	Towne 144	✓	✓	✓
Rough plastic table	Singh center roof deck	✓[2]	✓[1]	✓[1]
Rubber (back)	LMT	✓[1]	✗	✓
Rubber (front)	LMT	✓	✗	✓
Rubber plate	LMT	✓	✗	✗
Toy plastic airplane wing	YCB	✓	✓[1]	✗
White checkerboard	LMT	✓	✓	✓
Woven palm leaves	HaTT	✓[2]	✗	✗

B.8. Stone (polished and unpolished)

Surface	Source	Tooling ball	OptoForce	BioTac
Concrete floor 1	Towne sub-basement	✓[2]	✓[1]	✓[1]
Concrete floor 2	Towne sub-basement	✓[2]	✓[1]	✓[1]
Corrugated slate	LMT	✗	✓	✗
Dark stone banister	Annenberg (outside)	✓	✗	✗
Glossy marble bench	Annenberg (outside)	✓	✗	✗
Granite counter	Chem cafe	✓[2]	✓[1]	✓[1]
Granite curb	Towne south entrance	✓[2]	✓[1]	✓[1]
Marble banister	ARCH Cafe (outside)	✓	✗	✗
Marble banister	LMT	✓	✓	✓
Marble counter	34th &Walnut food court	✓	✗	✗
Marbled countertop	Accenture Cafe	✗	✓[1]	✓
PMC 5058-01	Princeton Material ConneXion	✓	✓	✓
PMC 5706-04	Princeton Material ConneXion	✗	✓	✓
PMC 6421-02	Princeton Material ConneXion	✓	✓	✓
PMC 7234-01	Princeton Material ConneXion	✓	✓	✓
PMC 7234-02	Princeton Material ConneXion	✗	✓	✓
PMC 7235-01	Princeton Material ConneXion	✗	✗	✓
PMC 7439-01	Princeton Material ConneXion	✓	✗	✗
PMC 7566-01	Princeton Material ConneXion	✗	✓	✓
Polished stone floor	Singh center lobby	✓[2]	✓[1]	✓[1]
Red clay panel	Fisher 3rd floor landing	✓[1]	✓[1]	✓[2]
Red granite railing	Vagelos patio	✓[2]	✓[1]	✓[1]

Surface	Source	Tooling ball	OptoForce	BioTac
Rough stone stairs	Singh center stairway	✓[2]	✓[1]	✓[1]
Shiny slate	LMT	✓	✓	✓
Slate	LMT	✓	✓	✓
Stone banister	Towne/Skirkanich boundary	✓[2]	✓[1]	✓[1]
Stone bench	Edgar Smith statue	✓[2]	✓[1]	✓[1]
Stone stairs	DRL W wing	✓	✓[1]	✓[1]
Stone stairs	Towne south entrance	✗	✓[1]	✗
Stone stairs (dense pebble inlay)	Skirkanich	✓[2]	✓[1]	✓[1]
Stone stairs (less dense pebble inlay)	Skirkanich	✓[2]	✓[1]	✓[1]

B.9. Wood

Surface	Source	Tooling ball	OptoForce	BioTac
Another wood table	Hayden Hall classroom	✓[2]	✓[2]	✓[2]
Antique Elm (Home Depot 169-597)	HaTT	✓[2]	✓[2]	✓[2]
Cork	BOLT	✓	✓	✓
Cork	HaTT	✓[2]	✓[2]	✓[2]
Cork	LMT	✓	✓	✓
Cork tile (front)	MPI (Michaela's collection)	✓[2]	✓[1]	✓[2]
Dark Walnut (Home Depot 172-739)	HaTT	✓[2]	✓[2]	✓[2]
Dark lacquered wood floor	College Hall foyer	✓	✓	✓
Dark wood bench	Singh center roof deck	✓[2]	✓[1]	✓[1]
Golden Maple	HaTT	✗	✓[2]	✓[2]

Surface	Source	Tooling ball	OptoForce	BioTac
IKEA chair	Alex's apartment	✓[1]	✓	✓
Iron Wood	HaTT	✗	✓[2]	✓[2]
Lacquered wood	LRSM	✓	✗	✗
Light lacquered wood floor	College Hall foyer	✓	✓	✓
Light wood table	Hayden Hall 2nd floor hallway	✓[2]	✓[2]	✓[1,2]
Notepad backing	BOLT	✗	✓	✓
PMC 5606-02	Princeton Material ConneXion	✗	✓	✓
PMC 7344-01	Princeton Material ConneXion	✗	✓	✓
PMC 7485-03	Princeton Material ConneXion	✓	✓	✓
PMC 7697-01	Princeton Material ConneXion	✗	✓	✓
Polished banded wood desk	Towne 144	✓	✓	✓
Polished hardwood floor	Alex's apartment	✓	✓	✓
Polished wood railing	34th &Walnut food court	✓	✗	✗
Round table	Towne 144	✓	✓[1]	✓[1]
Scratched wooden stairs	Vagelos	✓[2]	✓[1]	✓[1]
Slice of tree trunk	34th &Walnut food truck plaza	✓	✗	✗
Slightly darker wood table	Hayden Hall 2nd floor hallway	✓[2]	✓[2]	✓[2]
Smooth wooden table	Skirkanich elevator landing	✓[2]	✓[1]	✓[1]
Stained Wood	HaTT	✓	✓	✓
Teak (Home Depot 487-900)	HaTT	✓[2]	✓[2]	✓[2]
Wood	HaTT	✓	✓[2]	✓[2]
Wood	LMT	✓	✓	✓
Wood floorboard 1	Fisher 3rd floor landing	✓	✓[2]	✓[2]

Surface	Source	Tooling ball	OptoForce	BioTac
Wood floorboard 2	Fisher 3rd floor landing	✓	✓[2]	✓[1,2]
Wood picture frame backing	Towne 144	✓	✓	✓
Wood side table	Alex's apartment	✓	✓	✓
Wood stairs	DRL classroom	✓	✓	✓
Wood table	DRL classroom	✓	✓	✓
Wood table	Towne 144	✓	✓	✓
Wooden block	YCB	✓	✓	✗
Wooden peg board (back)	YCB	✓	✓	✗
Wooden stairs	Vagelos	✓[2]	✓[1]	✓[1]
Woven palm leaves	HaTT	✓[2]	✗	✓[2]

B.10. Miscellaneous

Surface	Source	Tooling ball	OptoForce	BioTac
Artificial grass	HaTT	✓[2]	✓[2]	✗
Binder	HaTT	✓	✓[2]	✓[2]
Bubbly foam	LMT	✓	✓	✗
Cotton	HaTT	✓	✓[2]	✓[2]
Duropal backing	LMT	✗	✓	✓
Fake stone wallpaper	LMT	✓	✓	✓
Foam brick	LMT	✗	✗	✓
Foam brick	MPI (Michaela's collection)	✓	✓	✓
Gelatin	BOLT	✓[1]	✓[1]	✓

Surface	Source	Tooling ball	OptoForce	BioTac
Gift Box	HaTT	✓[2]	✓[2]	✓[2]
Glitter Paper	HaTT	✓[2]	✓[2]	✓[2]
Gray foam pad	MPI (Michaela's collection)	✓	✓	✓
Gray leather (back)	LMT	✗	✓	✗
Gray leather (front)	LMT	✗	✓	✗
Hard book cover (blue)	Towne 144	✗	✓	✗
Hard book cover (light red)	Towne 144	✗	✓	✓
Hard sponge	BOLT	✓	✓	✓
Kitchen counter	LRSM	✓	✗	✗
Large Plastic Mesh	HaTT	✓	✓	✓
Layered foam	BOLT	✓	✓[1]	✓
Leather A (back)	HaTT	✓[2]	✓[2]	✓[2]
Leather A (front)	HaTT	✓[2]	✓[2]	✓[2]
Leather B (back)	HaTT	✓[2]	✓[2]	✓[2]
Leather B (front)	HaTT	✓[2]	✓[2]	✓[2]
Leather book cover (black)	Towne 144	✗	✓	✓
MDF	HaTT	✓	✓[2]	✓[2]
Maroon leather bench	Penn bookstore	✓	✗	✗
Mesh bag of wooden blocks	YCB	✓	✓	✗
Mesh chair	DRL classroom	✓	✓	✓
Mousepad	LMT	✓	✗	✗
Nitrile Glove	HaTT	✓[2]	✓	✓
Nylon Mesh	HaTT	✓	✓[2]	✓[2]

Surface	Source	Tooling ball	OptoForce	BioTac
Orange couch	Singh center study alcove	✓[2]	✓[1]	✗
PMC 5191-08	Princeton Material ConneXion	✓	✓	✓
PMC 5276-05	Princeton Material ConneXion	✓	✓	✓
PMC 5713-02	Princeton Material ConneXion	✓	✓	✓
PMC 5920-06	Princeton Material ConneXion	✓	✓	✓
PMC 6560-09	Princeton Material ConneXion	✓	✓	✓
PMC 6849-01	Princeton Material ConneXion	✓	✓	✗
PMC 6936-01	Princeton Material ConneXion	✗	✓	✓
PMC 7187-01	Princeton Material ConneXion	✓	✓	✓
PMC 7403-01	Princeton Material ConneXion	✓	✓	✓
PMC 7464-02	Princeton Material ConneXion	✗	✓[1]	✓
Portfolio Cover	HaTT	✓[2]	✓[2]	✓[2]
Ramp floor	Hayden Hall 1st floor lounge	✓[1,2]	✓[1,2]	✓[2]
Rug slip	HaTT	✓[2]	✓	✓
Scouring pad	YCB	✓	✓	✓[1]
Small Plastic Mesh	HaTT	✓	✓[2]	✓[2]
Smooth glitter paper	LMT	✓	✓	✓
Soft glitter paper	LMT	✓	✓	✓[1]
Soft sponge	BOLT	✓	✓	✓
Styrofoam	HaTT	✓[2]	✓[2]	✓[2]
Styrofoam brick	MPI (Michaela's collection)	✓	✓	✓
Tissue pack	BOLT	✓	✓	✓
Toolbox topper	Towne 144	✓	✓	✓

Surface	Source	Tooling ball	OptoForce	BioTac
Velcro Loops	HaTT	✓[2]	✓[2]	✓[2]
Vinyl A	HaTT	✓	✓[2]	✓[2]
White drawer liner	MPI (Michaela's collection)	✓[2]	✓	✓[2]
Whiteboard eraser (back)	BOLT	✓	✓	✓
Whiteboard eraser (board side)	BOLT	✓[1]	✓	✓
"Domino"	Penn bookstore	✗	✓	✓
"Morley"	Penn bookstore	✗	✓	✓
"Notations by Tschumi"	Penn bookstore	✗	✓	✓
"Sketchbook"	Penn bookstore	✗	✓	✓

[1]Images not available because of poor lighting conditions.
[2]Structure Sensor was turned off because of driver issues.

APPENDIX C : TERMINOLOGY

This is a list of all acronyms used in the document, their definitions, and the page where they were first introduced.

ABS acrylonitrile butadiene styrene (a type of 3D-printable plastic) 55

ADC analog-to-digital converter (electronic component) 25

CAD computer-aided design (engineering software) 49

CNN convolutional neural network (machine learning algorithm) 112

CoM center-of-mass (physics) 161

CSV comma-separated value (data file format) 32

DCT discrete cosine transform (signal processing algorithm) 74

EKF extended Kalman filter (state tracking algorithm) 68

ESKF error-state Kalman filter (state tracking algorithm) 69

FAST features from accelerated segment test (computer vision algorithm) 111

FSR force-sensing resistor (sensor) 8

GPU graphics processing unit (computer hardware) 112

I2C inter-integrated circuit (device communication protocol) 24

ILSVRC ImageNet Large Scale Visual Recognition Challenge (competition) 112

IRB Institutional Review Board (research ethics oversight committee) 114

IMU inertial measurement unit (electronic component) 8

KTH-TIPS2 Textures under varying Illumination, Pose, and Scale (material image database from the Kungliga Tekniska Högskolan (Stockholm Royal Institute of Technology)) 123

LED light-emitting diode (electronic component) 26

LMT Lehrstuhl für Medientechnik Texture (published dataset) 77

MAE mean absolute error (statistical measure) . . . 130

MDF medium-density fiberboard (stiff prototyping material, easy to laser-cut) 14

MFCC melfrequency cepstral coefficient (audio signal representation) . . . 123

MINC Materials In Context (material image database) . . . 112

NLP natural language processing (computer science discipline) . . . 111

OF optical flow (computer vision technique) . . . 68

PCB printed circuit board (electronic assembly) . . . 28

PNG portable network graphics (image file format) . . . 32

PNP Perspective-n-Point (computer vision technique) . . . 68

PVDF polyvinyledine fluoride (force sensitive film) . . . 11

RANSAC RANdom SAmple Consensus (statistical technique) . . . 160

RBF radial basis function (type of SVM/SVR kernel) . . . 57

RGB red-green-blue (camera type) . . . 17

RGBD red-green-blue-depth (camera type) . . . 17

RMS root-mean-square (statistical measure) . . . 69

RMSE root-mean-square error (statistical measure) . . . 128

SIFT scale-invariant feature transform (computer vision algorithm) . . . 111

SfM structure from motion (computer vision algorithm) . . . 111

SLAM simultaneous localization and mapping (robot navigation algorithm) . . . 69

SURF speeded-up robust features (computer vision algorithm) . . . 111

SPI serial peripheral interface (device communication protocol) . . . 27

SVD singular value decomposition (linear algebra technique) . . . 163

SVM support vector machine (machine learning algorithm) . . . 52

SVR support vector regression (machine learning algorithm) . . . 129

YCB Yale-Columbia-Berkeley (physical object set) . . . 87

BIBLIOGRAPHY

[1] J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker, "Human-inspired robotic grasp control with tactile sensing," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1067–1079, 2011. 2

[2] A. Burka, S. Hu, S. Helgeson, S. Krishnan, Y. Gao, L. A. Hendricks, T. Darrell, and K. J. Kuchenbecker, "Proton: A visuo-haptic data acquisition system for robotic learning of surface properties," in *Proc. IEEE International Conference on Multisensor Fusion and Integration*, 2016. 4, 25, 48, 51, 57, 59, 60, 64, 67, 78, 80, 152, 157, 162, 165

[3] A. Burka, A. Rajvanshi, S. Allen, and K. J. Kuchenbecker, "Proton 2: Increasing the sensitivity and portability of a visuo-haptic surface interaction recorder," in *Proc. IEEE International Conference on Robotics and Automation*, 2017, under review. 4, 21, 51, 58, 61, 62, 64, 65, 66, 67, 69, 70, 71, 78, 80, 152

[4] M. Strese, C. Schuwerk, and E. Steinbach, "Surface classification using acceleration signals recorded during human freehand movement," in *Proc. IEEE World Haptics Conference*, 2015, pp. 214–219. 5, 22, 52, 53, 59, 63, 64, 72, 73, 77, 78, 81, 85, 123, 152

[5] M. Strese, C. Schuwerk, A. Iepure, and E. Steinbach, "Multimodal feature-based surface material classification," *IEEE Transactions on Haptics*, vol. 10, no. 2, pp. 226–239, 2016. 5, 72, 73, 74, 75, 76, 77, 78, 81, 85, 118, 123, 152

[6] A. Burka and K. J. Kuchenbecker, "Handling scan-time parameters in haptic surface classification," in *Proc. IEEE World Haptics Conference*, 2017, pp. 424–429. 5, 51, 79, 81, 82, 84, 152

[7] ——, "How much haptic surface data is enough?" Workshop paper presented at the AAAI Spring Symposium on Interactive Multi-Sensory Object Perception for Embodied Agents, March 2017, 2017. [Online]. Available: http://www.cs.utexas.edu/~jsinapov/AAAI-SSS-2017/paper/Burka_AAAI_SSS_2017.pdf 5, 153

[8] H. Culbertson, J. Unwin, B. E. Goodman, and K. J. Kuchenbecker, "Generating haptic texture models from unconstrained tool-surface interactions," in *Proc. IEEE World Haptics Conference*, 2013, pp. 295–300. 7, 12, 22, 63, 64

[9] T. Bhattacharjee, J. M. Rehg, and C. C. Kemp, "Haptic classification and recognition of objects using a tactile sensing forearm," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4090–4097. 7

[10] D. Silvera-Tawil, D. Rye, and M. Velonaki, "Artificial skin and tactile sensing for socially interactive robots: A review," *Robotics and Autonomous Systems*, vol. 63, pp. 230–243, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2014.09.008 7, 154

[11] M. Strese, Y. Boeck, and E. Steinbach, "Content-based surface material retrieval," in *Proc. IEEE World Haptics Conference*, 2017, pp. 352–357. 8, 152

[12] M. Strese and E. Steinbach, "Toward high-fidelity haptic interaction with virtual materials: A robotic material scanning, modelling and display system," in *IEEE Haptics Symposium*, 2018, pp. 247–254. 8

[13] Izantux, "File:straingaugevisualization.svg," Wikimedia Commons. [Online]. Available: https://commons.wikimedia.org/wiki/File:StrainGaugeVisualization.svg 9

[14] C. Chorley, C. Melhuish, T. Pipe, and J. Rossiter, "Development of a tactile sensor based on biologically inspired edge encoding," *Design*, pp. 1–6, 2009. [Online]. Available: http://www.brl.ac.uk/projects/InspiredEdgeEncoding.pdf 8

[15] W. Yuan, S. Dong, and E. Adelson, "GelSight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, p. 2762, 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/12/2762 8, 154

[16] J. Fishel, "The BioTac - multimodal tactile sensor," http://wiki.ros.org/BioTac, accessed: 2018-05-03. License: CC-BY 3.0. 10

[17] S. A. Stansfield, "Primitives, features, and exploratory procedures: Building a robot tactile perception system," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 3, 1986, pp. 1274–1279. 10

[18] S. J. Lederman and R. L. Klatzky, "Hand movements: a window into haptic object recognition." *Cognitive Psychology*, vol. 19, no. 3, pp. 342–368, 1987. 10

[19] J. M. Romano and K. J. Kuchenbecker, "Methods for robotic tool-mediated haptic surface recognition," in *Proc. IEEE Haptics Symposium*, 2014, pp. 49–56. 10, 11, 22, 52, 57, 59, 61, 63, 64, 72, 152

[20] J. Sinapov, V. Sukhoy, R. Sahai, and A. Stoytchev, "Vibrotactile recognition and categorization of surfaces by a humanoid robot," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 488–497, 2011. 10, 11

[21] Y. Mukaibo, H. Shirado, M. Konyo, and T. Maeno, "Development of a texture sensor emulating the tissue structure and perceptual mechanism of human fingers," in *Proc. IEEE International Conference on Robotics and Automation*, 2005, pp. 2565–2570. 11

[22] N. Jamali and C. Sammut, "Majority voting: material classification by tactile sensing using surface texture," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 508–521, 2011. 11

[23] J. A. Fishel and G. E. Loeb, "Bayesian exploration for intelligent identification of textures," *Frontiers in Neurorobotics*, vol. 6, 2012. 11

[24] V. Chu, I. McMahon, L. Riano, C. G. McDonald, Q. He, J. M. Perez-Tejada, M. Arrigo, T. Darrell, and K. J. Kuchenbecker, "Robotic learning of haptic adjectives through physical interaction," *Robotics and Autonomous Systems*, vol. 63, no. 3, pp. 279–292, 2015-01. 11

[25] R. L. Klatzky, S. J. Lederman, C. Hamilton, and M. Grindley, "Perceiving surface roughness through a probe: effects of applied force and probe diameter," in *Proc. ASME Dynamic Systems and Control Division*, vol. 2, 2000. 11

[26] S. J. Lederman, R. L. Klatzky, C. L. Hamilton, and G. I. Ramsay, "Perceiving roughness via a rigid probe: psychophysical effects of exploration speed and mode of touch," *Haptics-e, The electronic journal of haptics research*, vol. 1, no. 1, 1999. 11

[27] D. Xu, G. E. Loeb, and J. A. Fishel, "Tactile identification of objects using Bayesian exploration," in *Proc. IEEE International Conference on Robotics and Automation*, 2013, pp. 3056–3061. 11

[28] N. F. Lepora, U. Martinez-Hernandez, and T. J. Prescott, "Active touch for robust perception under position uncertainty," in *Proc. IEEE International Conference on Robotics and Automation*, 2013, pp. 3020–3025. 11

[29] W. McMahan and K. J. Kuchenbecker, "Spectral subtraction of robot motion noise for improved vibrotactile event detection," in P. Isakoski and J. Springare, editors, *Haptics: Perception, Devices, Mobility, and Communication: Proc. EuroHaptics, Part I, volume 7282 of Lecture Notes in Computer Science*, 2012, pp. 326–337. 11

[30] D. K. Pai and P. Rizun, "The WHaT: a wireless haptic texture sensor," in *Proc. IEEE Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2003, pp. 3–9. 12, 64

[31] S. Andrews and J. Lang, "Interactive scanning of haptic textures and surface compliance," in *Proc. IEEE International Conference on 3-D Digital Imaging and Modeling*, 2007, pp. 99–106. 12

[32] M. Fiala, "ARTag revision 1, a fiducial marker system using digital techniques," Technical Report NRC/ERB-1117, NRC Publication Number 47419, 2004. 12

[33] E. Battaglia, G. Grioli, M. G. Catalano, M. Santello, and A. Bicchi, "ThimbleSense: an individual-digit wearable tactile sensor for experimental grasp studies," in *Proc. IEEE International Conference on Robotics and Automation*, 2014, pp. 2728–2735. 12

[34] R. Höver, M. Harders, and G. Székely, "Data-driven haptic rendering of visco-elastic effects," in *Proc. IEEE Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2008, pp. 201–208. 12

[35] H. Culbertson, J. J. L. Delgado, and K. J. Kuchenbecker, "One hundred data-driven haptic texture models and open-source methods for rendering on 3D objects," in *Proc. IEEE Haptics Symposium*, 2014-02, pp. 319–325. 12

[36] H. Van Hoof, O. Kroemer, H. Ben Amor, and J. Peters, "Maximally informative interaction learning for scene exploration," in *IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 5152–5158. 13

[37] "The Structure sensor," https://structure.io, accessed: 2018-02-08. 14

[38] R. L. Klatzky, S. J. Lederman, C. Hamilton, M. Grindley, and R. H. Swendsen, "Feeling textures through a probe: effects of probe and surface geometry and exploratory factors," *Perception & Psychophysics*, vol. 65, no. 4, pp. 613–631, 2003. 18, 64

[39] H. Culbertson, J. Unwin, and K. J. Kuchenbecker, "Modeling and rendering realistic textures from unconstrained tool-surface interactions," *IEEE Transactions on Haptics*, vol. 7, no. 3, pp. 381–393, 2014. 22, 63, 64, 87

[40] J. D. Brown, C. E. O'Brien, S. C. Leung, K. R. Dumon, D. I. Lee, and K. J. Kuchenbecker, "Using contact forces and robot arm accelerations to automatically rate surgeon skill at peg transfer," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 9, pp. 2263–2275, 2017. 25, 34

[41] "OpenNI2 downloads and documentation | the Structure sensor," https://structure.io/openni, accessed: 2018-02-08. 32

[42] R. Kaestner and M. Hoepflinger, "liboptoforce," https://github.com/ethz-asl/liboptoforce, accessed: 2018-02-08. 33

[43] V. Chu, I. McMahon, L. Riano, C. G. McDonald, Q. He, J. M. Perez-Tejada, M. Arrigo, N. Fitter, J. C. Nappo, T. Darrell, and K. J. Kuchenbecker, "Using robotic exploratory procedures to learn the meaning of haptic adjectives," in *Proc. IEEE International Conference on Robotics and Automation*, 2013, pp. 3048–3055. 34, 112, 168

[44] V. Chu, L. Riano, I. McMahon, K. Qin, and C. McDonald, "IanTheEngineer/Penn-haptics-bolt: Penn haptics team's repository for BOLT related work," https://github.com/IanTheEngineer/Penn-haptics-bolt, accessed: 2018-02-08. 34

[45] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5. 36

[46] Y. Gao, L. A. Hendricks, K. J. Kuchenbecker, and T. Darrell, "Deep learning for tactile understanding from visual and haptic data," in *Proc. IEEE International Conference on Robotics and Automation*, 2016, pp. 536–543. 52, 112

[47] N. Landin, J. M. Romano, W. McMahan, and K. J. Kuchenbecker, "Dimensional reduction of high-frequency accelerations for haptic rendering," in Astrid M. L. Kappers, Jan B. F. van Erp, Wouter M. Bergmann Tiest, and Frans C. T. van der Helm, editors, *Haptics: Generating and Perceiving Tangible Sensations, Proc. EuroHaptics, Part II, volume 6191 of Lecture Notes in Computer Science*, 2010, pp. 79–86. 54, 60, 73, 106

[48] H. Culbertson, J. J. López Delgado, and K. J. Kuchenbecker, "One hundred data-driven haptic texture models and open-source methods for rendering on 3D objects," in *Proc. IEEE Haptics Symposium*, 2014, pp. 319–325. 55, 56, 62, 64, 80, 95, 152, 153, 168

[49] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, pp. 1207–1245, 2000. 57, 61

[50] C. G. McDonald and K. J. Kuchenbecker, "Dynamic simulation of tool-mediated texture interaction," in *Proc. IEEE World Haptics Conference*, 2013, pp. 307–312. 63

[51] R. L. Klatzky and S. J. Lederman, "Tactile roughness perception with a rigid link interposed between skin and surface," *Perception & Psychophysics*, vol. 61, no. 4, pp. 591–607, 1999. 63

[52] M. A. Lawrence, R. Kitada, R. L. Klatzky, and S. J. Lederman, "Haptic roughness perception of linear gratings via bare finger or rigid probe," *Perception*, vol. 36, no. 4, pp. 547–557, 2007. 64

[53] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407. 67

[54] K. Daniilidis and J. Shi, "Lecture 26 - pose from 3D point correspondences: The Procrustes problem," in *Robotics: Perception*. Coursera, 2016. 68

[55] ——, "Lecture 27 - pose from projective transformations," in *Robotics: Perception*. Coursera, 2016. 68

[56] ——, "Lecture 37 - 3D velocities from optical flow," in *Robotics: Perception*. Coursera, 2016. 68

[57] J. Solá, "Quaternion kinematics for the error-state KF," *Laboratoire d'Analyse et d'Architecture des Systèmes-Centre National de la Recherche Scientifique, Toulouse, France, Tech. Rep*, 2012. 69

[58] V. Chu, I. McMahon, L. Riano, C. G. McDonald, Q. He, J. Martinez Perez-Tejada, M. Arrigo, T. Darrell, and K. J. Kuchenbecker, "Robotic learning of haptic adjectives

through physical interaction," *Robotics and Autonomous Systems*, 2014-10. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0921889014002061 72

[59] M. Strese, Personal correspondence, 2016–2017. 73

[60] K. Wojcicki, "HTK MFCC matlab," https://www.mathworks.com/matlabcentral/fileexchange/32849-htk-mfcc-matlab, accessed: 2017-01-11. 73

[61] M. Strese, C. Schuwerk, R. Chaudhari, and E. Steinbach, "LMT haptic texture database," http://www.lmt.ei.tum.de/downloads/texture/, accessed: 2017-01-11. 77, 87, 95, 168

[62] H. Culbertson and K. J. Kuchenbecker, "Should haptic texture vibrations respond to user force and speed?" in *Proc. IEEE World Haptics Conference*, 2015, pp. 106–112. 83

[63] Y. LeCun, C. Cortes, and C. J. Burges, "MNIST handwritten digit database," *AT&T Labs*, vol. 2, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist 86

[64] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5206848 86, 111

[65] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012. 86, 112

[66] D. Katz, *The World of Touch*. Lawrence Erlbaum, 1925, trans. L. E. Krueger, 1989. 87

[67] J. Leitner, D. G. Dansereau, S. Shirazi, and P. Corke, "The need for dynamic & active datasets," in *CVPR Workshop on the Future of Datasets in Vision*, 2015. 87

[68] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols," *Submitted to Robotics and Automation Magazine (RAM), Special Issue on Replicable and Measurable Robotics Research*, 2015. [Online]. Available: http://arxiv.org/abs/1502.03143 88, 95, 153, 168

[69] M. Regoli, N. Jamali, G. Metta, and L. Natale, "Controlled tactile exploration and haptic object recognition," in *Proc. International Conference on Advanced Robotics*, 2017, pp. 47–54. 88

[70] B. Hollis, S. Patterson, and J. Trinkle, "Compressed learning for tactile object classification," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1616–1623, 2018. [Online]. Available: http://arxiv.org/abs/1609.07542 88

[71] A. Harrison and P. Newman, "TICSync: Knowing when things happened," in *Proc. IEEE International Conference on Robotics and Automation*, 2011, pp. 356–363. 89

[72] G. M. Beylerian, A. Dent, and A. Moryadas, *Material ConneXion: the global resource of new and innovative materials for architects, artists, and designers.* Wiley, 2005. 95, 153

[73] B. Armstrong-Hélouvry, P. Dupont, and C. Canudas De Wit, "A survey of models, analysis tools and compensations methods for the control of machines with friction," *Automatica*, vol. 30, no. 7, pp. 1083–1138, 1994. 96

[74] M. Bloodgood and K. Vijay-Shanker, "A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping," in *Proc. 13th Conference on Computational Natural Language Learning*, 2009, pp. 39–47. 97, 99

[75] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, and K. Kavukcuoglu, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7585, pp. 484–489, 2016. [Online]. Available: http://dx.doi.org/10.1038/nature16961 110

[76] R. Munroe, "Tasks," https://xkcd.com/1425/, 2014, accessed: 2018-04-26. 110

[77] L. Hill, "Urban legends in computer vision," http://www.lyndonhill.com/opinion-cvlegends.html, 2017, accessed: 2018-04-26. 110

[78] S. A. Papert, "The summer vision project," 1966. 110

[79] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. European Conference on Computer Vision*, 2006, pp. 430–443. 111

[80] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157. 111

[81] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, 2008. 111

[82] PaddlePaddle, "Image classification," in *Deep Learning 101*, 2016, ch. 3, accessed: 2018-04-26. License: https://creativecommons.org/licenses/by-sa/4.0/. [Online]. Available: http://paddlepaddle.org/docs/develop/book/03.image_classification/index.html 112

[83] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Material recognition in the wild with the Materials in Context database," *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 3479–3487, 2015. 112, 126, 167

[84] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 112

[85] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 112

[86] W. Yuan, S. Wang, S. Dong, and E. Adelson, "Connecting look and feel: Associating the visual and tactile properties of physical materials," in *Proc. of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 21–26. 113

[87] S. Okamoto, H. Nagano, and Y. Yamada, "Psychophysical dimensions of tactile perception of textures," *IEEE Transactions on Haptics*, vol. 6, no. 1, pp. 81–93, 2013. 114

[88] E. Baumgartner, C. B. Wiebel, and K. R. Gegenfurtner, "Visual and haptic representations of material properties," *Multisensory Research*, vol. 26, no. 5, pp. 429–455, 2013. 118

[89] V. Andrearczyk and P. F. Whelan, "Using filter banks in convolutional neural networks for texture classification," *Pattern Recognition Letters*, no. 84, pp. 63–69, 2016. 123, 125

[90] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM International Conference on Multimedia*, 2014, pp. 675–678. 124

[91] San, "Evaluating a regression model," Cross Validated, accessed: 2017-04-13. [Online]. Available: https://stats.stackexchange.com/q/221311 127

[92] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947. 128

[93] Z. Xue, X. Zeng, L. Koehl, and L. Shen, "Interpretation of fabric tactile perceptions through visual features for textile products," *Journal of Sensory Studies*, vol. 31, no. 2, pp. 143–162, 2016. 149

[94] H. Zhang, J. Xue, and K. Dana, "Deep TEN: Texture encoding network," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2896–2905. [Online]. Available: http://arxiv.org/abs/1612.02844 149

[95] Y. Tenzer, L. P. Jentoft, and R. D. Howe, "Inexpensive and easily customized tactile array sensors using MEMS barometers chips," *IEEE Robotics and Automation Magazine*, vol. 21, no. 3, pp. 89–95, 2014. 154

[96] J. E. Colgate, "Parameterizing the tactile gamut," 2018, Cross-Cutting Challenges keynote presentation at IEEE Haptics Symposium, San Francisco, CA. 154

[97] A. Jennings, "Sphere fit (least squared) version 1.1," https://www.mathworks.com/matlabcentral/fileexchange/34129-sphere-fit-least-squared, 2013, accessed: 2018-03-02. 160

[98] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Proc. Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606. 163