Real-time Predictions from Unlabeled High-Dimensional Sensory Data during Non-prehensile Manipulation

by

Daniel Michael Troniak

B.CS. Hons. Co-op, The University of Manitoba, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

October 2014

Abstract

Robots can be readily equipped with sensors that span a growing range of modalities and price-points. However, as sensors increase in number and variety, making the best use of the rich multi-modal sensory streams becomes increasingly challenging. In this thesis, we demonstrate the ability to make efficient and accurate task-relevant predictions from unlabeled streams of sensory data for a non-prehensile manipulation task. Specifically, we address the problem of making real-time predictions of the mass, friction coefficient, and compliance of a block during a topple-slide task, using an unlabeled mix of 1650 features composed of pose, velocity, force, torque, and tactile sensor data samples taken during the motion. Our framework employs a partial least squares (PLS) estimator as computed based on training data. Importantly, we show that the PLS predictions can be made significantly more accurate and robust to noise with the use of a feature selection heuristic, the *task variance ratio*, while using as few as 5% of the original sensory features. This aggressive feature selection further allows for reduced bandwidth when streaming sensory data and reduced computational costs of the predictions. We also demonstrate the ability to make online predictions based on the sensory information received to date. We compare PLS to other regression methods, such as principal components regression. Our methods are tested on a WAM manipulator equipped with either a spherical probe or a BarrettHand with arrays of tactile sensors.

Preface

The thesis is based on work conducted jointly between the Sensorimotor Systems and IMAGER laboratories with professors Dinesh K. Pai and Michiel van de Panne, and students Daniel Troniak and Chuan Zhu of the University of British Columbia. This section outlines the contributions of each of the above individuals. Students worked under guidance of the professors.

Software Troniak wrote the C++ software framework for the control, perception and user interface of the robot, and developed MATLAB scripts in support of the design and usage of the TVR algorithm. Zhu implemented a MATLAB framework for data analysis, processing and figure generation, including the final implementation of the TVR algorithm.

Experiments Troniak designed and built the environment for the robot, collected all data with the robot, maintained and configured the hardware, and designed and executed the robot motion. Pai supported the purchase and maintenance of the robotic hardware. Zhu analyzed preliminary data collected during various manipulations to help Troniak determine successful motion trajectories. Data analysis, discussion, and final figure generation was accomplished collaboratively.

Algorithms van de Panne suggested the TVR feature selection metric and proposed the usage of the PLS algorithm. Troniak validated the effectiveness of the TVR algorithm on data collected during manipulations. Zhu performed similar validation on the usage of the PLS algorithm, as well as designed the scheme for realtime predictions using PLS.

Writing Chapters 3, and 5 of the thesis are adapted from a paper submitted to the 2015 IEEE International Conference on Robotics and Automation (D Troniak, C Zhu, D Pai, M van de Panne, Real-time Predictions from High-dimensional Unlabeled Sensory Data during Non-prehensile Manipulation, ICRA 2015, submission #1575, 01 October 2014). Troniak performed and wrote the background literature review and drafted the remainder of the manuscript, which was then improved and prepared for submission by van de Panne.

Figures Figure 1.2 was generated by Troniak. Part of Figure 1.1 was designed by van de Panne and is present in the paper submitted to ICRA 2015. Figures from Chapter 5 were developed collaboratively and are also present in the paper. Figures borrowed from the literature include appropriate references within the caption.

Table of Contents

Appendices

List of Tables

List of Figures

Acknowledgements

This thesis would not have been possible if not for my supervisors Professor Dinesh Pai and Professor Michiel van de Panne, whose guidance, patience and trust allowed this project to meet its full potential. They are both phenomenal advisors, and together brought complementary perspectives to this work, enhancing both its depth and breadth. Individual thanks go to Professor Pai for securing the robotic system, for providing high-level direction and for donating much of his personal time while on official leave, and to Professor van de Panne for providing concrete instruction and for never failing to find ample time in his busy schedule for stimulating discussion.

I am grateful to my second-reader and mentor, Professor Jim Little who provided much needed moral support through the inevitable dark times and who patiently guided me through finalizing my thesis.

Thanks to Chuan Zhu, my "un-a-believa-da-ble" lab-mate, with whom I shared much elation (and dissapointment!), for providing encouragement and comedy in the late-hours before deadlines.

Special thanks to Professor van de Panne and all members of the MOCCA reading group for listening to my (oft mundane) talks on topics related to my interests. Special thanks to Professor Pai and the SMRT reading group for helping me guide topics in Neuroscience into this thesis. Special thanks to Professor Little and the Robuddies reading group for lending me their undivided attention at my thesis talk.

Last but certainly not least, I would like to thank my family, both in Canada and in Taiwan, for their tireless effort to keep me secure, well-rested (and well-fed!) during the process of writing this thesis.

To Jill and Jade.

Chapter 1

Introduction

We begin this chapter by introducing robotic manipulation using force and tactile sensors, and present the challenges in programming robots to perform manipulation tasks in the physical world. Following this, we state the specific problem we wish to address in this thesis as well as the motivations behind seeking its solution. The chapter concludes with main contributions and thesis structure.

1.1 Learning to Interact with the Real World

Our world is infinitely complex. By the second law of thermodynamics, any real-world system constantly changes as energy transfers between its composing particles of matter [77]. Thus, if one's goal is to interact with such a dynamic ever-changing world, one needs to continually infer relevant truths which exist in that world through some form of external sensing.

This type of approach to system identification and control is known as *model-free* [64] in the sense that one functions without needing explicit knowledge of physics.

Living organisms are known to take this approach [16]. A child does not learn to walk given absolute knowledge of the dynamics which govern his interaction with the environment. Rather, he must build an action-consequence model of the walking task through much trial and error.

By learning a model which maps sensory information to relevant properties of an environment, the entity also gains an ability to predict anomalies. Novel events are important as they present opportunities to gain deeper insight into the true underlying principles which define the environment not presently described by the current model.

Humans, who are among the most successful of living organisms, have the ability to both model their environment and efficiently perform inference on that model using their available senses. For a trained athlete to succeed in the game of basketball, she must learn to ignore irrelevant sensory data, such as the cheering of the crowd, and focus on more important information, such as the image of the basket on her retina. How can she identify relevant information with respect to her current task? We are interested in exploring how humans might accomplish this by enabling robots to possess such a skill.

1.2 Problem Statement

We wish to address the problem of supporting robotic manipulation by efficiently predicting properties of an environment using unlabeled sensory data. The ability to predict properties of an environment, such as the mass of an object on a table, is an important prelude to closing the loop and allowing robots to adapt to unexpected change.

One barrier to achieving this goal, however, is that the size of the input space increases with the number of sensor readings available to the system. This makes making predictions susceptible to Bellman's infamous curse of dimensionality. We therefore take steps to reduce the number of sensors considered when predicting properties of an environment.

To test our solution, we consider a block topple-slide task (see Figure 1.1) performed by a robot equipped with force, torque and tactile sensors. In this task, the end effector of the robot pushes on the side of a foam-encapsulated block, causing it to topple, and then pushes the block back to its starting configuration against a wall. A prescribed joint-angle trajectory to achieve this manipulation task is provided by a human expert through kinesthetic teaching. The given example trajectory is assumed to be robust in two ways: repeating the trajectory should cause repeated rotations of the block, and the same trajectory should remain successful when applied to blocks with variations in mass, friction, and compliance.

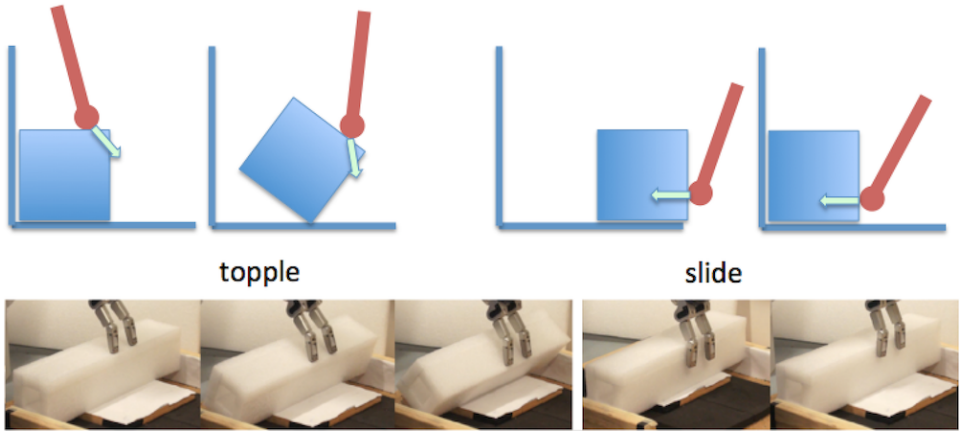


Figure 1.1: The topple-slide task.

1.3 Motivations

As sensing technologies become more affordable, robots will be able to concurrently make a wide range of measurements within the environments in which they operate. As can be seen in Figure 1.2, it is clear that sensors with the capacity to measure properties of an environment will obtain distinct readings as those environment properties change. Notice how the streams become disjoint during phases of robot-object contacts (during toppling and sliding) and recombine during phases of non-contact. It then becomes possible, given some form of ground-truth, for the robot to learn a model which explains how changes in its sensor readings relate to changes in its environment.

This releases the need for experts to model the relationships between sensory readings and environmental phenomena. For example, distinct force measurements may relate to the compliance of an object, which in turn could provide insight into some high-level feature such as its ripeness (if the object were a piece of fruit). We thereby achieve a highly scalable system capable of mapping arbitrary sensor readings (vision, olfactory, inertial, etc.) to arbitrary environment properties, given the availability of relevant ground-truth information on which to train the model-learning system.

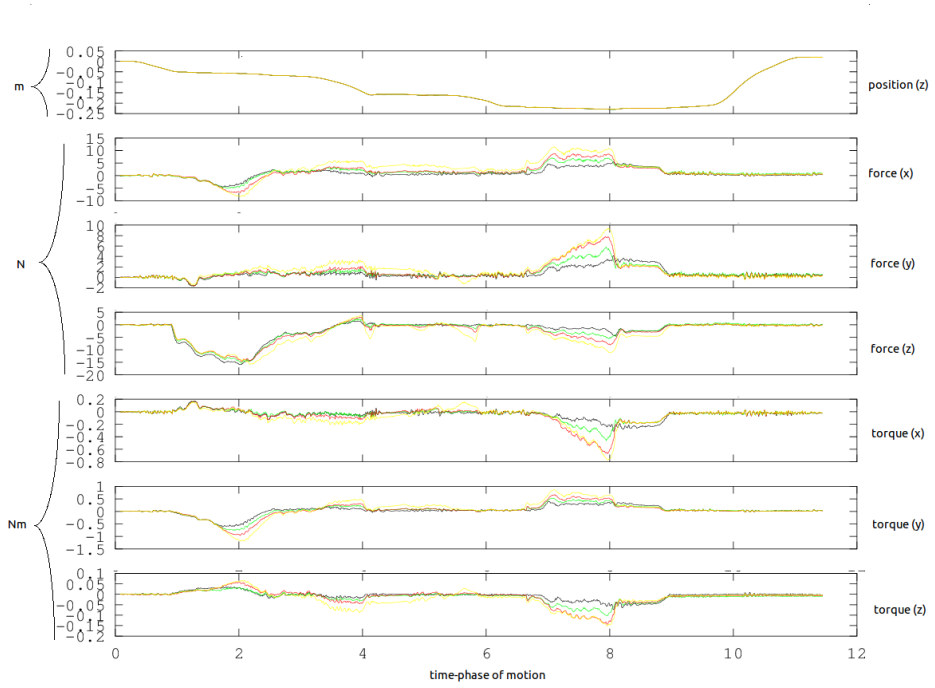


Figure 1.2: Example sensory streams during the topple-slide task in different environments (one colour for each environment).

1.4 Contributions

The contributions of this thesis are three-fold: (1) a unifying bridge between literature in robotics, physiology and neuroscience on the topic of exploiting force and tactile sensors for dexterous manipulation, (2) an unsupervised feature selection algorithm based on a new metric called the task variance ratio, which filters important sensor readings and motion-phases within high-dimensional sensory data streams during manipulation tasks and (3) a supervised learning algorithm with partial least squares (PLS) regression at its core, which builds statistical data-driven models that use important sensory data traces from a robot to predict properties defining its environment.

The feature selection algorithm allows the robot to distinguish between

sensors which provide task-critical information, and sensors which provide information that is noisy or irrelevant to the task. This ability to gauge the usefulness of sensor readings becomes important as the number of sensors in the system increases, and real-time processing of all sensors becomes impossible. Disregarding all but the most important sensors reduces algorithmic complexity to constant time, which guarantees support for real-time operation irrespective of the number of sensors physically available to the robot. The developed models are then used to efficiently predict environment properties when novel sensory data traces are recorded by the robot's sensors.

1.5 Thesis Structure

Following this introductory chapter, in Chapter 2 we provide some background on the topic of exploiting force and tactile sensors in support of physical manipulation, as presented in a sample of the literature from the robotics, neuroscience and physiology research communities. In Chapter 3, we more formally define the problem we wish to solve in this thesis and define the algorithms and data structures used in our approach. Chapter 4 serves as an overview to the robot control software developed to support experiments. In Chapter 5, we present details on experiments conducted to test the effectiveness of our approach on a physical robotic system, along with results of said experiments. Finally, conclusions are drawn and future work is provided in Chapter 6.

Chapter 2

Background

2.1 Robotic Manipulation

In the present study, we consider the general research question: how should robots be programmed to manipulate (e.g. grasp) objects? There exist two common approaches to this problem in the robotics literature. One approach is for a human expert to provide the robot with an analytical model of itself and its environment. Using this model, the robot achieves its manipulation goals in two phases: planning and execution. Planning is often performed in simulation using models of object geometry and robot/environment dynamics. Once the robot has found a feasible plan, it then executes that plan via physical robot actuation. One disadvantage to this approach is that due to imperfect precalculated dynamical models and robot calibration, executed manipulations can easily become unstable and fail [13].

Another approach is to remove the requirement of human-supplied models and have the models learned autonomously from data using statistical learning techniques. The key advantage of this type of *model-free* approach [30, 64] is that apriori object and dynamical models are not required to succeed in the task; the model is learned automatically from data. In this thesis, we define *model-free* [64] systems as those that function without given explicit knowledge of physics or geometry from human experts.

2.1.1 Model-free schemes

One example of such a learning technique is called the Self-Organizing Map (SOM), an architecture of Artificial Neural Networks (ANN) that spatially and uniformly organizes features automatically by input signals [39]. An

example where SOMs have been used successfully is in [65]: objects are grasped based on hand posture and tactile experience of previously successful grasps. Experience is represented as a low dimensional smooth manifold in hand posture space.

A similar system was devised in [54], where a SOM was used to map finger joint angles and tactile readings to object shape and size. The system could identify previously grasped objects as well as categorize new objects as being a particular shape and size.

The same authors obtained similar results with another algorithm inspired by biological spiking neurons, called a spiking neural network [53]. For this scheme, joint angle input is encoded into a series of spike trains which result in three feature outputs that are then used to recognize and classify grasped objects. In addition, similar objects tended to cluster in output feature space. The authors' system was able to recognize objects of different shapes as well as objects with the same shape but different size.

In [14], the authors present blind grasping: a novel approach to object grasping that does not require visual feedback or apriori 3D object models. Their scheme works from a database of one thousand stable grasps from the Columbia Grasp Database [25] using the model of a BarrettHand [71]. Corresponding tactile feedback during grasps of objects simulated in the Graspit! [45] simulator are also recorded. They proceed to create feature vectors comprising simulated tactile and robot kinematic data which they then use to train an SVM to classify grasps as being stable or unstable. In this way, the system was able to learn tactile feedback indicative of a stable grasp.

2.1.2 Model-based schemes

In *model-based* schemes, a human expert provides the robot apriori models which, for example, map sensory input signals to specific control policies. This approach grants the advantage of providing the robot access to the understanding of the task dynamics of the researcher. The disadvantage of this approach is that the supplied model may contain errors and is limited

by the domain expertise of the researcher.

In [20], the authors present sensor-based atomic controllers for a robotic hand/arm system to empty a box containing an undefined number of unknown objects. Manipulation primitives are defined that search, grasp and transport objects from the box to predefined locations. A finite state machine (FSM) is used to transition between motion primitives based on corresponding sensory feedback. The authors also compare a vision-plus-tactile-based version of their system to a purely tactile-based version. They found that while the version which incorporated vision was more efficient at completing the empty-the-box manipulation task, the tactile version was also successful. Vision was only crucial in determining if the box was empty; in the non-vision based system, a human moderator was required to tell the robot when it had finished its task. The authors in [20] also present an interesting scheme that compensates for errors in translation of the robotic hand. The hand repositions itself if there is force experienced by only one finger, denoting a single hand/object contact. The controller compensates by moving the hand in direction of single contact, which effectively repositions the manipulator above the object.

In [44], the authors take an analytical approach to the non-prehensile toppling task. This approach, while successful, assumes knowledge of the dynamics of the entire system and would therefore not generalize to operation outside of controlled factory environments where complete models of robot-object interaction dynamics were unavailable. Another analytical approach to a non-prehensile tumbling task, given apriori models of how the system reacts to the robot at each phase of the task, is studied in [58].

In [76], an analytical approach is applied to the non-prehensile task of manipulating an object with rolling contacts across a robotic finger tip using tactile sensor feedback. Their approach relies on accurate apriori kinematic and dynamic models of the robot and its environment.

2.1.3 Human-inspired schemes

In the human hand, there are a wide variety of receptors in the skin and muscles which in turn respond to a wide variety of stimuli. Sensed phenomena include skin stretch, skin curvature, vibration and muscle force and length. One baffling aspect of the human motor system however is that information bandwidths range from just a few Hz to possibly several hundred Hz. In terms of technological performance, this is horrendously slow. Information is also time varying, nonlinear, and its encoding scheme (known as pulse-frequency) obscures much of the raw inputs from the nerve endings. How these deficiencies are made up for, however, is a high degree of parallelism and redundancy [31].

It is also known that the human motor system executes manpulations as a series of discrete states that transition based on afferent signals [36]. Since this type of model is appropriate for execution on a computer, it has been quite popular to model the robotic grasping task as an FSM, which transitions between states based on tactile or other intrinsic contact input events [41, 61].

The authors in [41] present an approach to tactile-motor coordination of a robotic hand based on a neurological model of the human tactile-motor system. This model is implemented as a series of ANNs whose function and structure reflect discoveries in the human sensory areas specific to object grasping. A scheme based on SOMs was chosen to model these sensory areas, since they must process a high rate of combined tactile and somatosensory input. Use of the SOM controlled the volume of incoming inputs by making small, efficient adjustments to the model each time a new input vector became available.

The authors in [56] developed a human-inspired robotic grasp controller that gently picks up and sets down unknown objects. They employ pressure sensors and accelerometers to mimic SA-I, FA-I and FA-II tactile channels (see Section 2.3.2). An FSM is programmed to transition between six discrete states: (1) Close, (2) Load, (3) Lift and Hold, (4) Replace, (5) Unload, and (6) Open. Transitions are based entirely on tactile event cues. Their

controller also dynamically adapts its initial grasp force depending on tactile events such as slipping, and judges when to set down the object in light of detected contact events with the table.

In [61], a new tactile-based object manipulation strategy was proposed, called tactile servoing. Analogous to vision-based visual servoing, each state in the manipulation task sequence is characterized by tactile images detected via tactile sensor arrays on the robot hand. The authors' conclusion was that tactile sensors are useful in simple, direct and effective control of robots during manipulation tasks. The literature supports the fact that tactile data is processed much the same way that visual data is processed in the human brain [35, 59].

2.2 Sensory Information Processing

2.2.1 Force and tactile sensing for robotic manipulation

Force and tactile sensing can provide information about mechanical properties, such as compliance, coefficient of friction, and mass, which are not perceivable through other means (e.g. vision) [31]. Obtaining object properties via force and tactile sensing for the purposes of succeeding in manipulation tasks has been the subject of numerous studies [17, 27, 42].

The application of force and tactile sensors to many robotics problems affords new solutions that have previously been intractable via traditional, often computer vision-based methods [40]. In their 2005 review article, Tegin and Wikander [68] stress that, in contrast to the amount of literature on the application of vision-based solutions to robotics problems, literature on exploiting contact information (e.g. tactile) remains relatively rare. One reason may be simply due to the lack of availability of force and tactile sensors in comparison to cameras [31].

While vision is arguably the dominant sense in primates, including humans, there are certain scenarios in which vision fails, such as during object occlusion or when sensory resolution is too low for a given task. In such cases, more detailed and versatile contact information may compensate for

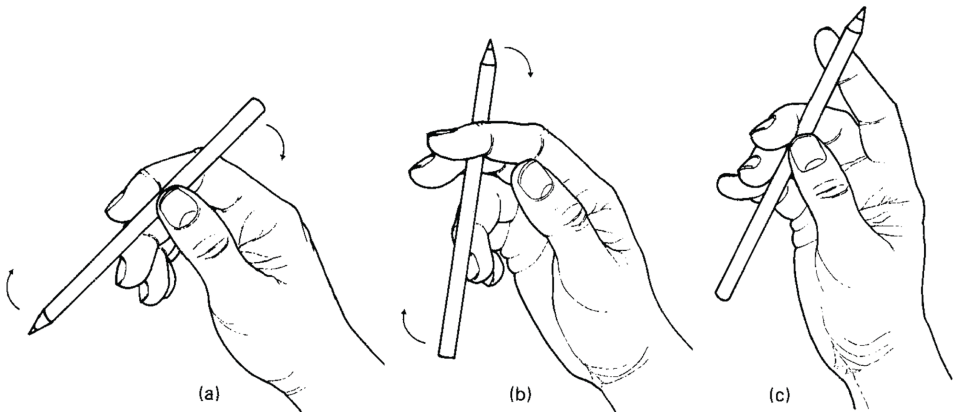


Figure 2.1: Schematic drawing of the pen-twirling task. Drawings reproduced from [18]. Use of the reproduction is by permission of the copyright owner, John Wiley and Sons.

these deficiencies.

In [11] the authors review techniques for processing and combining force and tactile information to develop abstract understanding of a given manipulation. An excerpt of these processing techniques is shown graphically in Figure 2.2: as raw sensory data travels from left to right, they are processed and combined to provide increasingly abstract understanding of a manipulation. The authors state that force and tactile sensors have potential to yield the following information: (1) object contact/no contact; (2) contact configuration (surface, edge, point, etc.) based on pressure-patterns; (3) object slip via vibrations in the grasped object, (4) properties (compliance, texture, friction, etc.) of an object via haptic exploration; and (5) feedback for control. Given the above information, a robot can more appropriately control the force and moment on an object to accomplish the desired manipulation task.

Example

Consider the following example: how might one accomplish the task of twirling a pen end-over-end between one's fingers, as demonstrated in Figure 2.1? The position and orientation of the object must somehow match

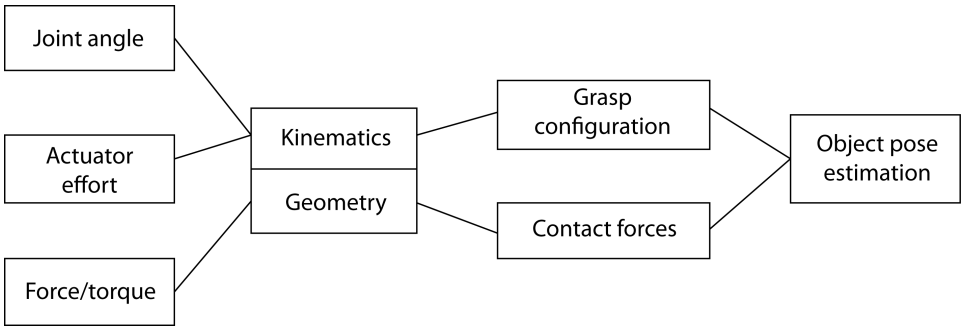


Figure 2.2: Force and tactile sensor processing to estimate object pose [11].

imposed forces to maintain stability. Successfully tracking the movement of the pen requires the knowledge of many variables, such as the configuration of one's hand, the locations and movements of contacts between the pen and one's fingers, the magnitudes of grasp forces, the contact conditions with respect to friction limits, etc. How is it that, with enough practice, one can control all of these parameters effortlessly, even in the absense of visual feedback?

A potential answer can be seen in Figure 2.2: we can combine the forward kinematic model of the hand together with current finger joint angles to determine the positions and orientations of the finger tips. When combined with force/torque measurements at the points of contact, it is possible to obtain local information of object shape, surface normal orientation, etc., which could then be combined to track the geometric pose of the object.

Tactile sensing

For robotic hands with tactile sensor arrays, such as the BarrettHand, curvature and shape information can be obtained by measuring the local curvature at each element of the sensor array [11]. From there, it is possible to extract features, such as corners and edges of the object by combining local shape information from multiple sensors. This task can be greatly enhanced if at least a partial model of the grasped object is available apriori, in which case the object can be statistically matched via surface or data

fitting methods [19].

The most common application of tactile information has been to classify and recognize objects from a known set based on calculated geometric information of the object from raw tactile data. Features, such as holes, edges and corners [11] and object surfaces [50] have been used and extracted from tactile array, force and/or joint sensor information. For example, Siegel [60] devised a way to extract object pose of a known object in a robot's grasp via joint angle and joint torque measurements.

Active sensing

Since force and tactile sensors provide only local object information, recognition and disambiguation often require the hand to actively explore multiple areas of the object surface. These types of strategies are referred to as active sensing. There exist many example applications of active sensing, such as tracing object contours, measuring compliance and determining lateral extent of object surfaces. In [47], the authors propose an active sensing strategy to edge-finding by exploring the surface of an object until contiguous segments of tactile array impressions are found. In [42], tactile sensors are used to discriminate shape and position of various textured cylindrical objects. In [17], grasp affordances are obtained through exploration of the pose space of manipulable objects.

Dynamic sensing

The ability to detect tactile events with respect to time (e.g. object slip) is important to many manipulation tasks such as lifting fragile objects. The challenge lies in detecting such events reliably in the presence of sensor noise. Highly sensitive tactile sensors capable of detecting minute events can be easily thrown off by e.g. vibrations from the robot actuators or by rapid robot acceleration. Robust dynamic event detection can be solved by comparing tactile sensor readings at and away from contact regions, or even more robustly via statistical pattern matching methods that detect the signature of particular dynamic events [73].

2.2.2 Anomaly detection in streaming data

Since high-dimensional data streams often exhibit considerable structure, information that does not fit within this structure is most likely an anomaly, or outlier, in comparison to the vast majority of other input data. An anomaly can be defined as an event or pattern which does not conform to some well-defined notion of normal phenomena. Detecting the existence of anomalies within data streams is an important topic within both the data mining and machine learning communities [1, 15, 38, 63, 75] and has far-reaching applications in such areas as fault detection, fraud detection, sensor-networks and image processing [6]. In [30], a model-free approach is taken to find anomalies in high-dimensional sensory streams. Data collected from the robot are first passed through a PCA-based feature extractor before building models of normal operation.

2.2.3 Dimensionality reduction

As the number of sensors available to a system increase, the computational, storage and transmission costs in inferring information from all available sensor readings also increase. Therefore, given a large number of sensor readings, it is important to develop a reduced set of measurements or derived features that can model desired information in a compact fashion. Dimensionality reduction techniques can be classified into two broad categories: (1) feature extraction and (2) feature selection. Most feature *extraction* techniques take an unsupervised [24, 28, 57, 69] or self-supervised [2, 12, 43, 62] learning approach. The result is a transformed, lower-dimensional set of features that more compactly describe the underlying structure of the data. In contrast, feature *selection* techniques, such as those based on feature similarity [46] and genetic algorithms [32], achieve dimensionality reduction by considering only a subset of input dimensions. In [51], the authors present a novel feature selection method comparing the variance of sensor readings to choose to encode either force or position information while recording user-demonstrated trajectories.

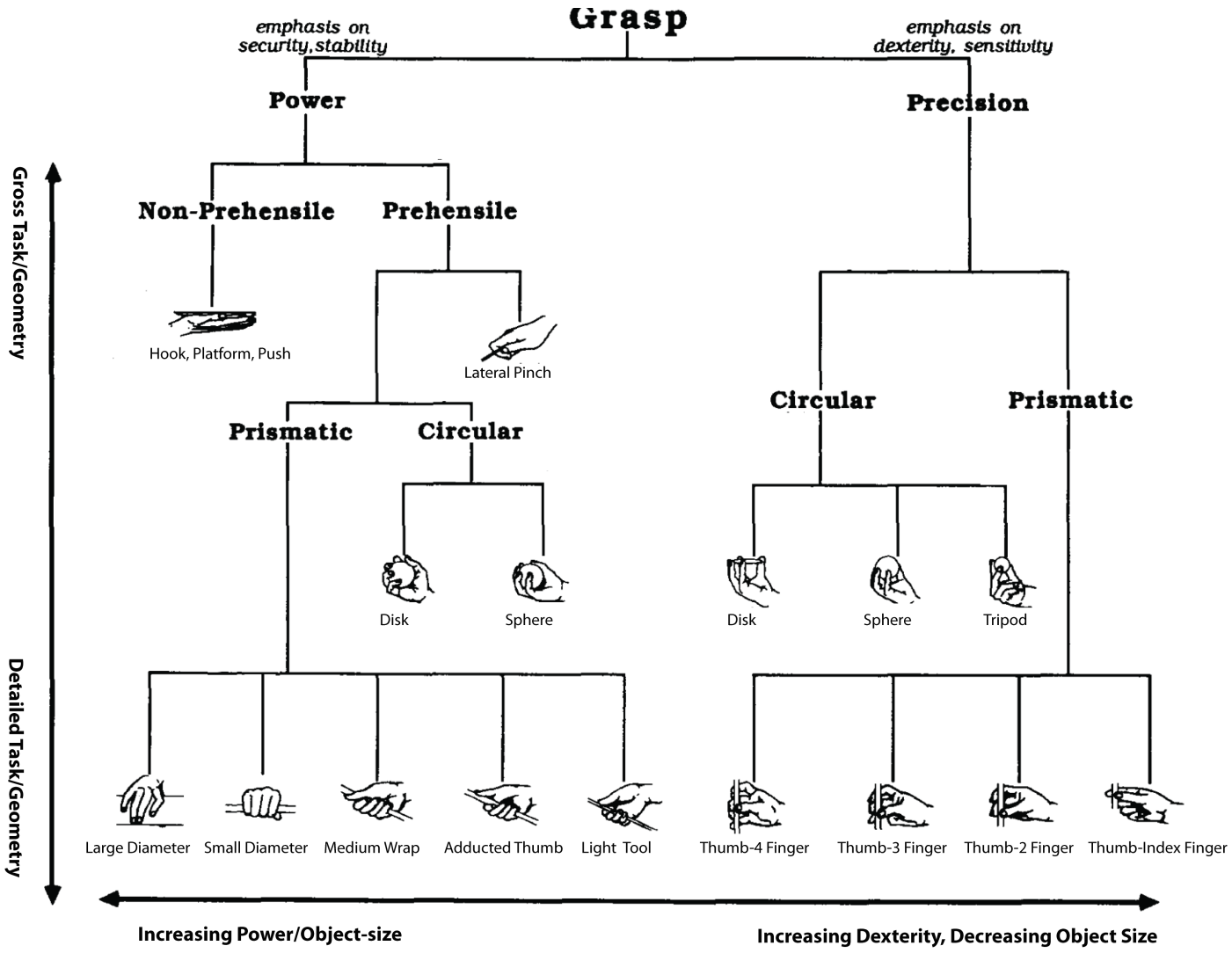


Figure 2.3: Dichotomy of human grasping, reproduced from [9]. ©1989 IEEE.

2.3 Neuroscience & Physiology

Studying the manipulation capabilities of humans and animals for the purpose of designing better robotic systems is a challenge. First, it is hard to discover the precise algorithms that our brains employ. Second, the mechanics of the human hand is highly complex and thus the algorithms our motor system employs may not be appropriate for the relatively simple mechanics of a robot. Nevertheless, studying human manipulation can provide insight into designing more efficient and effective robotic systems. In this section, we attempt to draw such insight by exploring the human motor system as presented in a sample of the neuroscience and physiology literature.

2.3.1 Object manipulation: definitions

In this section, we present some common vocabulary used by researchers in describing object manipulation tasks as performed by humans.

The power-precision dichotomy

Humans employ a wide variety of manipulation skills depending on the object being manipulated. When opening a jar, for example, a power-style grip is required to loosen the jar. Once the lid is loose and required torque is lessened, a lighter grip is adopted for speed and precision. This dichotomy of power/precision prehensile (i.e. grasping) activities was proposed by Napier in 1956 [49]. Figure 2.4 provides an example of these two patterns of activity in the manipulation task of tying a knot: power is required to hold the rope in place while precision is required to tie the knot. Cutkosky and Wright also propose a taxonomy of human grasps in [9], breaking down the dichotomy of power/precision even further (see Figure 2.3). Depending on the weight and size of the object as well as the desired dexterity of the hand, a human adopts a different style of grip.

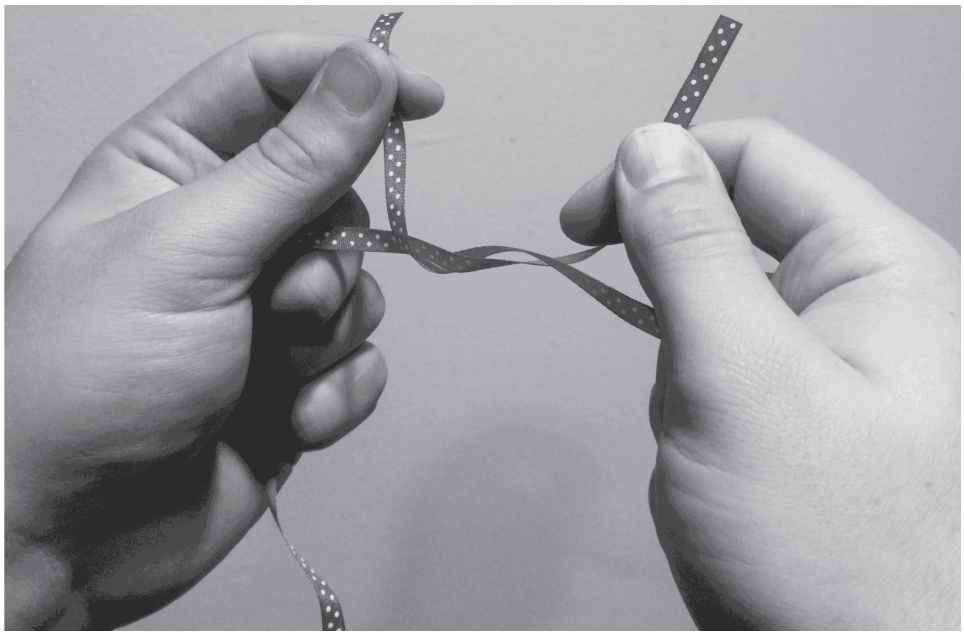


Figure 2.4: Tying a knot: manipulation task combining precision and power grips. Figure inspired by [49].

Analytical measures of grasp quality

The authors in [10] present common measures of grasp quality, which may be optimized or become part of the set of constraints with respect to a given manipulation task. An overview of these analytical grasp-quality measures is presented in Table 2.1. The set of ideal grasps of any object then exists within the space of grasps that satisfy all hard constraints and optimize important soft constraints with respect to the given task. For example, Nakamura et al. search for grasps that minimize internal forces (i.e. grasping effort), subject to constraints on force closure and manipulability [48]. According to physiological findings, humans tend to employ a similar scheme as proposed by Nakamura et al. where a certain frictional safety margin is maintained [55].

Human grasps have also been studied in terms of these analytical measures. For example, power grasps can be thought of as having higher compliance, stability and slip resistance than precision grasps. Power grasps also tend to have a connectivity of zero (since the fingers tend not to play a manipulating role). In contrast, precision grasps have high manipulability and connectivity (of at least three and often six) [10].

Metric	Description
Compliance	Inverse-stiffness of the object with respect to the hand
Connectivity	Number of DOFs between grasped object and the hand
Form closure	External forces are unable to unseat the grasped object
Force closure	Object held without slipping (a.k.a. frictional form closure)
Grasp isotropy	Fingers are able to accurately apply force/moment to object
Internal forces	Kinds of internal grasp forces hand may apply to the object
Manipulability	Fingers can impart arbitrary motions (i.e. connectivity = 6)
Slip resistance	Amount of force required before object starts to slip
Stability	Tendency of grasped object to return to a spatial equilibrium

Table 2.1: Common analytical measures that may be optimized or become a part of grasp constraints [10].

Force vs. form closure

A subtle yet important distinction must also be made between force closure and form closure. Form closure refers to grasping without the use of friction whereas force closure uses friction to keep objects seated in the hand. An object likely requiring form closure would be for example a wet bar of soap or a slinky.

Slipping vs. crushing threshold

While adequately large grip forces must be maintained to keep the object within a force closure grasp, exceedingly large forces are also not desirable as they impose unnecessary fatigue on the hand and may even crush fragile objects [33], [26]. Thus, grip force is constrained by both the slipping and crushing thresholds of objects (see Figure 2.5 for some examples).

The amount of force that subjects apply over and above the slipping threshold is known as the *safety margin*. The magnitude of the safety margin varies across subjects, and was found to be dependent on the dexterous manipulation skill of the subject in performing the given task [36].

When manipulating visually fragile objects, the initial force in human subjects is lighter and their action is slower when compared to manipulating visually non-fragile objects. Once contact with the object is made, tactile feedback complements the missing information with respect to the true fragility of the object. Subjects can then properly carry out the planned action [7]. Accurate predictions are crucial however due to the relatively slow response rate of corrective actions [34].

2.3.2 Force and tactile sensing

The elements of the human sense of touch can be broken up into two distinct categories: proprioceptive and tactile. Proprioceptive sensing refers to the perception of limb motion and forces using internal receptors, such as muscle spindles (responding to changes in muscle length), tendon organs (measuring muscle tension), and cutaneous afferents (reacting to skin deformations around the joints) [34]. Proprioceptive receptors within the joints of the

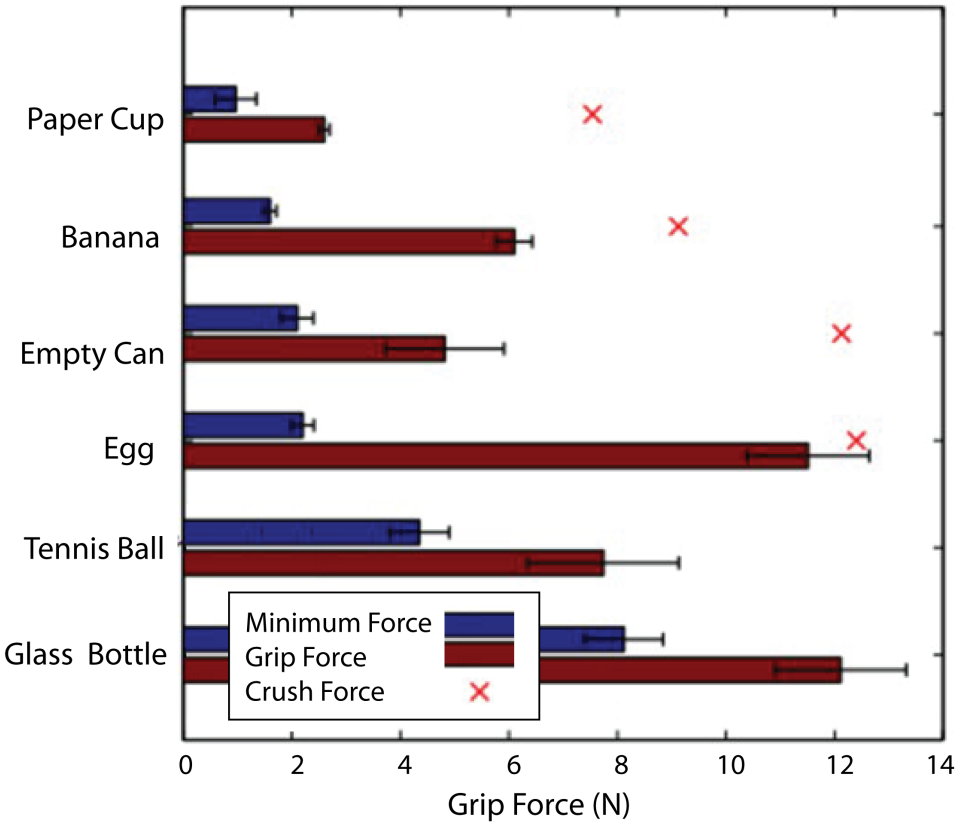


Figure 2.5: Example slipping and crushing thresholds of everyday objects. The difference between the required Minimum Force and observed Grip Force is known as the *safety margin*. Data obtained from [56].

hand are also present, which report joint angles, forces and torques [31]. Tactile sensing deals with the perception of contact information with receptors beneath the surface of the skin [74].

Actuation of the hand is imparted by muscles in the forearm through transmission of tension by tendons passing through the wrist. It has been shown that due to dynamics of transmission such as friction, backlash, compliance and inertia, accurate control of endpoint position and forces based on proprioceptive signals alone is difficult [37]. Thus, tactile afferents are essential for fine-grained mechanical measurements at contact locations [36].

Tactile afferents have received much attention in the physiology and neuroscience literature; a comprehensive summary of which may be found in [74] and, more recently in [34]. There are in total four specialized types of mechanoreceptive nerve endings within the skin of the human hand, each of which can be categorized as having large or small active areas (Type I and Type II respectively) and responding or not responding to static stimuli (SA for slowly adapting and FA for fast adapting, respectively). See Figure 2.6 for a description of each of these types. It has been calculated that a total of 17,000 specialized mechanoreceptors exist in the grasping surfaces of the human hand [34]. In addition, there are free nerve endings that are sensitive to thermal and pain stimuli [31].

2.3.3 High-level processes

In addition to low-level tactile and proprioceptive processing, the mammalian central nervous system performs many high-level processes such as prediction, planning and memory. These processes support, guide, and organize our more primitive manipulative functions to accomplish more complex manipulation tasks.

Prediction

In [33], the authors preclude that the magnitude of fingertip forces imposed on objects are determined by at least two high-level control processes: (1) anticipatory parameter control (APC) and (2) post-contact control. The

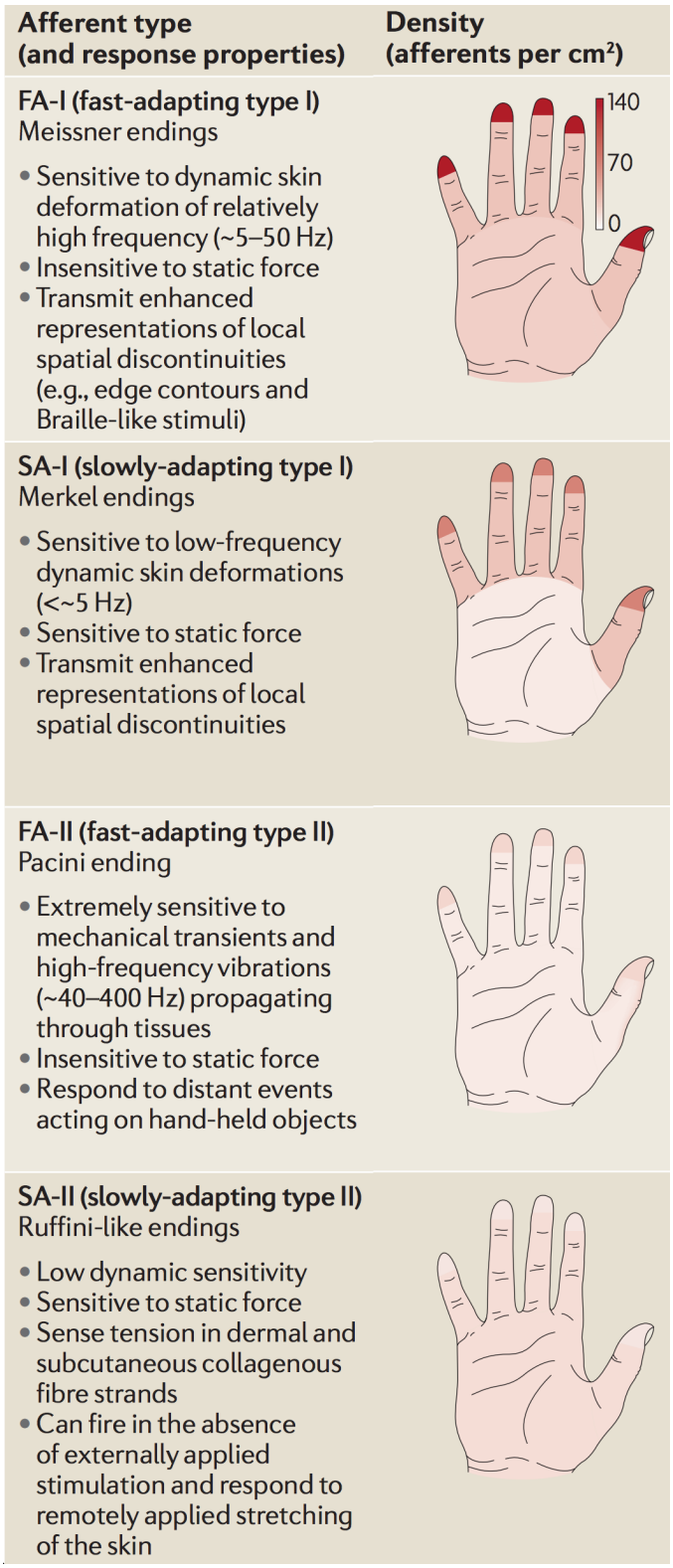


Figure 2.6: Characteristic of tactile afferents within human fingertip skin. Reproduced from [34]. Permission to reproduce granted by R.S. Johansson.

authors model APC as a feedforward controller that uses predictions of critical characteristics of the object (weight/friction/initial condition, etc.) based on the results of previous object manipulation experience. Following contact with the object, sensory information can be extracted to (1) modify motor commands automatically; (2) update sensory memories for APC; (3) inform central nervous system of the completion of subgoals of a task; and (4) trigger subsequent subgoals. The central nervous system monitors specific, expected events and produces control signals appropriate to each subgoal. In contrast to feedback controllers, this feedforward, sensor-driven control strategy predicts appropriate control output several steps in advance. Slips are avoided and force across digits is coordinated by independent control mechanisms based on local sensory information.

Planning

Planning plays an important role in anticipating future events as well. In [35], Johansson, et al. demonstrate the importance of eye-hand coordination during manipulation tasks. Subjects' gaze were tracked during a block-stacking task. It was found that their gaze played an important role in planning each pick-and-place action. The authors then further propose and demonstrate in [22] the direct matching hypothesis, which predicts that subjects will unconsciously produce eye movements when observing a familiar action as if they were performing the task themselves.

Supramodal processing

In a study conducted by Bicchi et al. [59], it was found that the V5/MT cortex (the same area in the brain that responds to optical flow) is activated during tactile-flow perception, i.e. when dynamic movement is detected via tactile afferents. This is consistent to other findings that there exists a supramodal, or multi-modal, organization of regions in the brain involved in both tactile-flow and optical-flow processing [21]. In another study by Bicchi et al. [4], it was found that certain experiments could fool the subjects' tactile flow processing in the brain through tactile illusions, much the same way

that optical-flow processing can be fooled, which is known as the *aperture problem*.

Action-phase control strategies

Findings by Johansson et al. indicate that, during certain manipulation tasks, the human motor system functions as a sort of state machine that transitions based on sensory predictions and sensory inputs. These states, or *action-phases*, are defined as sequences of specific sensory events that are each linked to subgoals of a given task [34].

Action-phase goals are evaluated by matching patterns in tactile afferent signals. For example, *grasp contact* – a required action-phase subgoal for many manipulation tasks – detects patterns in SA-I and FA-II afferent inputs. Combinations of certain afferents provide information such as contact timing, location, force intensity and direction. Contact location is defined as the spatial center of all afferents involved in the overall signal. Force intensity is characterized by the number of afferents involved as well as the firing rates of each. Patterns of activity in combinations of afferents give us the direction of the detected contact force.

Dexterity

Once a grasp is attained, adequate force within the friction cone of the object must be imposed to retain force closure. Dexterity is then defined as the ability to adapt the balance of grip and load forces to object surface properties [31]. Dexterous manipulation abilities are attributed mainly to tactile afferents since a loss in these abilities is experienced during digital anesthesia [36].

Object identification

Tactile afferents during initial contact also provide object surface property information, which is frequently combined with visual cues and/or sensory memories to develop abstract understanding. Reactions of FA-I, SA-I and

SA-II afferents to object surface are used to determine object surface properties. For example, FA-I afferents react more strongly to slippery surfaces [5].

Filtering noise

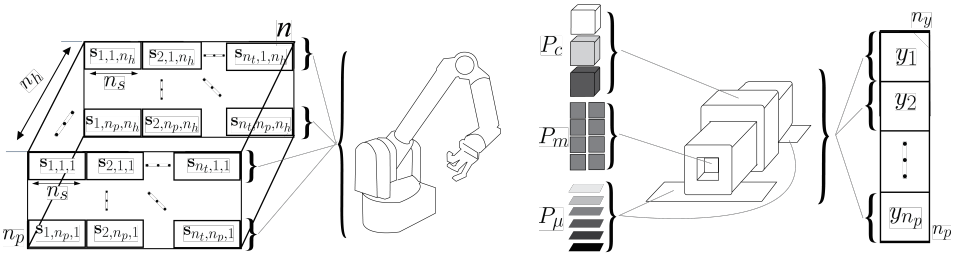
Robust processing of tactile afferent information is attributed to the brain's innate ability to detect coincidence: a phenomenon in which its central neurons receive synchronous input spikes from many distinct tactile afferents [29]. Therefore, noise in the environment, i.e. information unrelated to the current focus of attention, can be characterized by input spikes which do not arrive at the brain at the same precise moment in time as input deemed valuable to the current task.

Chapter 3

Predicting Environment Properties from Sensory Inputs

In this chapter, we first formally define the problem of predicting characteristics of an environment using high-dimensional force and tactile sensor data. We then present the prediction framework designed to solve this problem.

3.1 Problem Definition



(a) $\mathbf{X}$ (b) $\mathbf{Y}$

Figure 3.1: Collection of data matrix $\mathbf{D} = (\mathbf{X}, \mathbf{Y})$: (a) data $\mathbf{X}$ are collected across all trials and environments; (b) environmental quantities $\mathbf{Y}$ are provided by the human expert.

The manipulation task considered is shown in Figure 1.1. The end effector pushes down on a foam-encapsulated block, causing it to topple, then pushes the block back to its starting position against the wall. A prescribed joint-space trajectory to achieve this manipulation task is provided by a hu-

man expert via kinesthetic teaching. Upon replay, the robot tracks the given trajectory using standard computed-torque control. We assume the trajectory is robust in two ways: (1) repeating the trajectory causes repeated rotations of the block, and (2) the same trajectory remains successful in toppling blocks for all mass, friction, and compliance values. The use of a prescribed trajectory also implies an approximate correspondence between the current elapsed time within a motion and the manipulation phase.

Each sensory sample collected at each timestep of the prescribed trajectory is defined by a sensory data stream:

$$\mathbf{s} \in \mathbb{R}^{n_s} : (\rho, \dot{\rho}, \tau, p, \omega, f, \alpha) \tag{3.1}$$

where $\rho, \dot{\rho}, \tau \in \mathbb{R}^7$ represent respectively the angles, velocities and torque measurements of each of the seven joints of the manipulator arm, $p \in \mathbb{R}^7$ gives the end effector's pose measurements (3D position and 4D quaternion), $\omega \in \mathbb{R}^6$ is the task wrench measured via the force-torque sensor mounted to the wrist of the robot, $f \in \mathbb{R}^4$ are torque measurements for the joints in the robot hand, and $\alpha \in \mathbb{R}^{72}$ are tactile sensor measurements on each of the three fingers, reshaped into a single vector.

A single execution of the manipulation task leads to the capture of a sensory stream, $\mathbf{x} \in \mathbb{R}^n$, which consists of the observations of n_s sensor readings each sampled at n_t points in time, and then stacked into a single vector; here, $n = n_s \times n_t$.

The environment properties to be predicted from the sensory stream data are given by $\mathbf{y} \in \mathbb{R}^3 : (m, \mu, c)$, where m is the object mass, μ the coefficient of friction between the object and its support surface, and c is the material compliance of the object.

In order to learn a predictive model $\mathbf{y} = f(\mathbf{x})$, training data is first gathered for n_p different combinations of the environment properties, i.e., variations of mass, compliance, and friction. For each setting, the task is repeated n_h times in each environment. The final dataset is thus defined by

the following data pairs:

$$\mathbf{D} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_{p,h}, \mathbf{y}_p) \mid p \in [1 \cdots n_p], h \in [1 \cdots n_h]\}, \tag{3.2}$$

shown graphically in Figure 3.1. This dataset is used to learn the predictive model that we now describe.

3.2 Prediction Framework

Given a new sensory stream, $\mathbf{x}_{new}$, which consists of n_t samples of $\mathbf{s}$, we wish to predict the environment properties $\mathbf{y}$. For our work, the prediction task is characterized by having a small number of observations, $p = 144$, and a large number features to use for the prediction, $n = 1650$[1]. Furthermore, many of the features are likely to be highly correlated. These characteristics are problematic for many common regression methods. In contrast, PLS is a good alternative for such problems [23]. In particular, it couples the dimension reduction and the regression model, making the dimension reduction dependent on the input, $\mathbf{X}$, *and* the output, $\mathbf{Y}$. While PLS is a popular tool in the biological sciences and elsewhere, its use in the context of robotic sensing remains rare.

An important remaining challenge with PLS, however, is that while it provides a principled estimation method, the method itself is not tailored for variable or feature selection [8]. Relatedly, it can be shown that the PLS estimator does not guarantee statistically-consistent predictions for problems like ours with a high feature-count to sample-count ratio, and that noise variables act to attenuate the predictions of the regression parameters [8]. We specifically address this by imposing an aggressive feature selection method before applying PLS. In our results, we demonstrate this to be highly effective in helping to achieve improved prediction performance. We note that other methods have also been recently proposed to address the limitations of PLS, such as imposing sparsity in the dimension reduction step of

[1] Note that our notation for p and n is the reverse of that used in the statistics literature, where n commonly represents the number of samples, and p represents the number of predictors, i.e., the feature count.

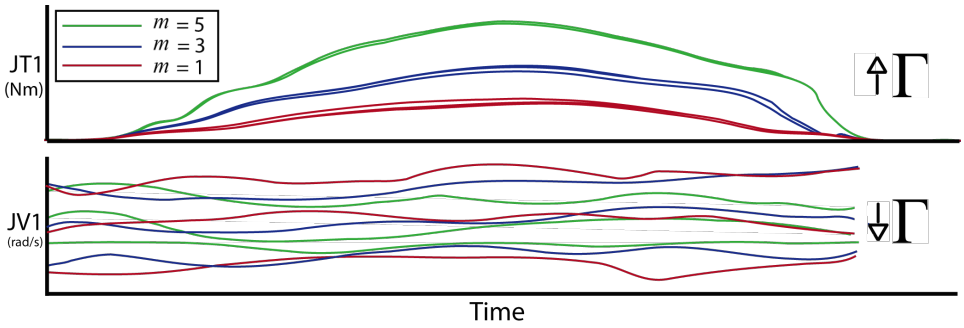


Figure 3.2: Intuition behind the task variance ratio (Γ) algorithm. We wish to select good sensors (at each time-phase) which exhibit low variance when the environment remains constant and high variance when the environment changes. Colours signify data streams collected in distinct environments.

PLS [8] or Supervised Principle Components [3]. As currently developed, these alternative methods are motivated-by and tested on gene expression problems.

With the above motivation in mind, our two-stage solution consists of (i) selecting the most *relevant* input features from $\mathbf{x}$, and (ii) using PLS to further learn a compact latent linear subspace that is well suited to predicting environment properties. We now discuss these two stages in further detail.

3.2.1 Feature selection via the task variance ratio

We define the *Task Variance Ratio* (Γ) vector as

$$\Gamma = \{\Gamma^i = \frac{Var_i^{enviro}}{Var_i^{trial}} \mid i \in [1 \cdots n]\} \tag{3.3}$$

where Var_i^{trial} models the variance of a given element of $\mathbf{X}$ across all trials, and Var_i^{enviro} models the variance of the same element across all environments. Specifically, for feature i:

$$Var_i^{trial} = \frac{\sum_{p,h}(x_{i,p,h}^2 - \mu_{i,p}^2)}{n_p n_h - 1} \tag{3.4}$$

and

$$Var_i^{enviro} = \frac{\sum_{p,h}(x_{i,p,h}^2 - \mu_i^2)}{n_p n_h - 1}, \tag{3.5}$$

where i, p, h are the indices for features, environment properties, and repeated trial number, respectively;

$$\mu_{i,p} = \sum_h x_{i,p,h}/n_h;$$

and

$$\mu_i = \sum_{p,h} x_{i,p,h}/n_p n_h.$$

A large value of Γ_i indicates a good feature, as it implies that variation occurs as changes to the environment take effect, while observable noise between repeated trials in the same environment is relatively small.

The feature selection is then implemented using a simple threshold function to produce a reduced input matrix $\mathbf{X}^*$:

$$\mathbf{X}^* = \mathbf{X}_p \cdot diag(\Gamma > \Gamma_{\min}), \tag{3.6}$$

where

$$\mathbf{X}_p = \frac{1}{n_h} \sum_{h=1}^{n_h} \mathbf{x}_{p,h} \mid p \in [1 \cdots n_p], \tag{3.7}$$

$diag(v)$ produces a square matrix with the elements of v across the diagonal, and Γ_{min} is chosen such that the desired number of elements of Γ are selected. The resulting reduced dataset is given by

$$\mathbf{D}^* = (\mathbf{X}^*, \mathbf{Y}) = \{(\mathbf{x}^*{}_p, \mathbf{y}_p) \mid p \in [1 \cdots n_p]\}. \tag{3.8}$$

In this way, we identify features in $\mathbf{X}$ that exhibit small variation across repeated trials when the environment is kept constant and exhibit large variations as the environment changes (see Figure 3.2), which we approximate as the degree of relevance of the sensor reading to predicting environment properties.

3.2.2 Property prediction with partial least squares

The PLS algorithm provides us with an estimated weighting matrix $\beta \in \mathbb{R}^{c \times n_y}$, where c is a parameter denoting the number of components to factor.

β is calculated iteratively according to the following algorithm: first, define

$$\begin{aligned} A_0 &= \mathbf{X}^{*\mathbf{T}}\mathbf{Y}, \\ M_0 &= \mathbf{X}^{*\mathbf{T}}\mathbf{X}^*, \quad \text{and} \\ C_0 &= I, \end{aligned} \tag{3.9}$$

then iterate

$$\begin{aligned} q_j &= eigv1(A_{j-1}^T A_{j-1}) && q_j \rightarrow \text{dominant eigenvector} \\ w_j &= A_{j-1} q_j \\ c_j &= w_j^T M_{j-1} w_j \\ w_j &= \frac{w_j}{\sqrt{c_j}} && \text{store into column j of W} \\ r_j &= M_{j-1} w_j && \text{store into column j of R} \\ q_j &= A_{j-1}^T w_j && \text{store into column j of Q} \\ v_j &= \eta C_j p_j && \eta \rightarrow \text{normalizing constant} \\ C_j &= C_{j-1} - v_j v_j^T \\ A_j &= C_j A_{j-1} \end{aligned} \tag{3.10}$$

for all $j \in [1 \cdots c]$. With R, Q and W assembled, we now compute:

$$\beta = W Q^T \tag{3.11}$$

Finally, we use β at runtime to predict environment properties:

$$\hat{\mathbf{y}} = \beta \cdot \mathbf{x}^*_{\mathbf{new}} \tag{3.12}$$

where $\mathbf{x}^*_{\mathbf{new}}$ represents the reduced version (i.e. following Γ feature selection) of a new unlabeled sensory data stream collected during a repetition of the motion in an unknown environment.

Chapter 4

Control Software

In this chapter, we provide an overview of the software deployed to control the motion of the robot and collect sensor readings in support of experiments.

4.1 System Overview

To support experiments, a sensor processing and robot control framework was written in C++ leveraging the libbarrett API provided by Barrett Technology (MA, USA) (Section 4.2). Our framework was developed exclusively on the internal PC of the WAM robot [72], the details of which are presented in Table 4.1.

Motherboard	Aaeon PFM-540I
Processor	500 MHz AMD LX-800 x86-compatible
Memory	256 MB 200-pin DDR-333 SODIMM
Linux distribution	Ubuntu 9.10
Linux kernel	2.6.31.4
Realtime	Xenomai 2.5
Ethernet	10/100 Base-T
CANbus	Peak PCAN-PC/104, 2 ports

Table 4.1: Specifications of the WAM Internal PC/104 [67] used for framework development and robot control in support of experiments.

4.2 Libbarrett API

The API used to communicate with the robot is called libbarrett: a C++ library from Barrett Technology Inc. [66]. Our framework was tested using

version 1.1.0 of the API. The libbarrett API provides abstract control of the WAM and BarrettHand and allows them to be controlled in tandem. Sample programs that perform simple control and sensor monitoring routines are provided, upon which our controller in the current study is based.

Libbarrett provides three high-level constructs to interface with the robot: (1) the WAM object (2) the Hand object and (3) the ProductManager object. The WAM and Hand objects provide high-level control of the WAM and attached BarrettHand respectively. The ProductManager provides access to the WAM's optional components, such as attached tools (e.g. BarrettHand) and Force/Torque sensor. The control software communicates with the Hand and WAM through high-level function calls to their respective libbarrett objects using a variety of pre-defined data structures. These datastructures are presented in Table 4.2.

Name	Type	Unit
Cartesian Position	cp_type	Meters
Joint Position	jp_type	Radians
Joint Velocity	jv_type	Radians/s
Joint Torque	jt_type	Radians/s

Table 4.2: Libbarrett data-types when communicating with the robot. The jp_type for the WAM is a seven-dimensional vector whereas for the BarrettHand, it is a four-dimensional vector – one entry for each finger and one for the spread.

4.3 WAM Control

Cartesian-space Control in Cartesian space is a convenient way to prototype motions and is sufficiently repeatable for tasks which require low accuracy. Accuracy of the Cartesian positioning of the WAM is advertised at two millimeters. In practise, however, this accuracy depends largely on the joint-angle configuration of the WAM at the beginning of the Cartesian-space move. In our experiments, position error could easily accumulate to reach as high as one centimeter, depending on the joint-angle position of

the robot at the beginning of the Cartesian-space move. These high errors may be due to imperfect inverse-kinematics currently available through the libbarrett API. Cartesian trajectories were not sufficiently repeatable in our experiments since we required a precise and highly repeatable motion (accurate to within 1 mm) to perform our task across various environments. Moreover, the workspace accessible by rotating the wrist is limited by the angular configuration of its attaching joint. This means that certain wrist orientations are not repeatable if the inverse-kinematics solution provides differing joint-space positioning around the wrist. These drawbacks prevent us from commanding the robot in Cartesian-space in our experiments.

Joint-space For accurate and repeatable sets of motions, the robot should be controlled directly in joint-space. The only issue with joint-space control is that the robot performs its task in Cartesian-space. Joint-angle trajectories that accomplish specific Cartesian-space tasks are difficult if not impossible to define manually. This necessitates kinesthetic teaching where the robot is trained to perform its task in Cartesian space by a human expert manually moving the robot tool to perform a task, while corresponding joint-space trajectories are recorded by the robot.

Kinesthetic Teach-and-Play The libbarrett API comes equipped with kinesthetic teaching functionality via the teach-and-play module. Teach-and-play records position information at the rate of 500Hz while a user physically moves the robot through the desired motions. The robot can record its trajectory in either Cartesian-space or joint-space. Again, due to imperfect inverse-kinematics, if a highly repeatable motion is required it is advisable to record trajectories in joint-space. It becomes possible to execute highly repeatable Cartesian-space trajectories if the relative displacement from a known starting joint-angle position is small (i.e. a workspace of approximately $10cm^3$ around a starting joint-angle position).

4.4 BarrettHand Control

In this section, we introduce the details on control of the BH8-280 BarrettHand through the libbarrett API, as used in our experiments.

Velocity Move The simplest form of control of each finger of the Hand is to specify a direction and rate of travel of each of the joints in the Hand. The joints will halt gracefully if they reach their limits or become obstructed before they reach these limits.

Trapezoidal Position Move If the Hand must be configured precisely (for e.g a pre-grasp posture), the user can specify desired joint-angles of each finger and spread. The spread of the Hand refers to the single degree of freedom rotation of the first and third fingers about the palm. Upon sending position commands to the hand, the proximal finger joint angles or the angle of spread move toward the goal position via a trapezoidal velocity profile. As with velocity control, the fingers will halt gracefully if the movement of the hand becomes obstructed.

High Control-rate Position Move High control-rate position moves provide an advanced alternative to simple trapezoidal moves. Joint-angles can be specified to reach a desired pose, however each of the joints travel at maximum velocity until they reach the desired pose or become obstructed. Care must be taken when sending these commands, as obstructions do not result in graceful halts and instead could cause damage to objects or the Hand itself. It is necessary to ensure a clear path for each of the fingers and spread of the Hand before sending these commands.

4.5 Realtime Systems

Control of the robot in realtime must be done through the libbarrett realtime systems API. Realtime control allows for complex and closed-loop motions since the output of each realtime system depends upon its inputs

at each step of the realtime control loop at the rate of 500Hz. Program 1 provides an example program which defines such a realtime system in the C++ programming language.

```
class WamSystem : public System {
public:     Input<jp_type> input;       // Obtain current joint-angles as input
public:     Output<jp_type> output;     // Provide updated joint-angles as output
protected:  Value* outputValue;         // Value that output reads
protected:  jp_type jp_offsets;         // Modifications to realtime motion feed
public:
    WamSystem(const string& sysName):   // All systems must define a name
        System(sysName), input(this), output(this, &outputValue){
            init_vec(&jp_offsets, 0);   // Initialize all offsets to 0
        }
    ~WamSystem(){ mandatoryCleanUp(); } // Mandatory destructor
protected:
    jp_type jp_out;                     // Declare local copy of output data
    virtual void operate() {
        const jp_type& jp_in = input.getValue();    // Pull data from the input
        jp_offsets[5] += 0.001;         // Increase 6th joint-angle of WAM slightly
        jp_out = jp_in + jp_offsets;    // Modify wam joints by relative offsets
        outputValue->setData(&jp_out);  // Push data to subsequent system
    }
};
```

Program 1: Example libbarrett realtime system written in C++ that controls the 6th joint of the WAM to increase indefinitely. Namespace references removed for brevity.

Chapter 5

Experiments and Results

In this chapter, we describe the experimental setup, provide details on the experimental procedure, and finally present and discuss environment property prediction results obtained by the prediction framework.

5.1 Setup

5.1.1 Actuation and sensing

Experiments are conducted using a 7 DOF Barrett WAM robot arm with attached 4 DOF Barrett BH-280 Hand, built by Barrett Technology (MA, USA). A 6-axis force-torque sensor is mounted to the wrist of the arm. The robot hand is equipped with tactile arrays on the fingers and palm. Joint torque sensors are embedded in each of the 3 fingers. Position control of the arm occurs at 500Hz and all sensors are sampled at 125 Hz, which is reduced to 2.5 Hz during preprocessing (see section 5.2.2).

5.1.2 Kinesthetic teach-and-play

Example trajectories are demonstrated to the robot via a kinesthetic teach-and-play interface. The system records pose estimates of the arm at the rate of 500 Hz and the result is saved for future playback.

5.1.3 Software architecture

Our real-time control framework runs on top of the *libbarrett* real-time systems library [66], and is used during demonstration and autonomous execution. We also use it to record and play back data streams that are time-synchronized with the motion. See Chapter 4 for further details.

5.1.4 Experimental testbed

The block used for the experiment is a rectangular prism made of medium-density polyethylene foam, with length 48.5 cm, width 10.5 cm and height 10.5 cm. Two parallel walls of length 28.5 cm and width 6.5 cm are used to prevent the block from sliding sideways out of the workspace. The distance between the walls is 49 cm. As shown in Figure 1.1, a wall is used to limit the final sliding motion and leave the block in its original location, ready to be toppled again. The walls are lined with paper to decrease the coefficient of friction between the block and the walls for smoother operation.

In our experiments, different environments are defined by the Cartesian product of three sets of environment property values for $P = \{P_m, P_\mu, P_c\}$, yielding a total of $8 \times 6 \times 3 = 144$ different environments. These values are shown in Table 5.1.

P_m (g)	425, 650, 875, 1100, 1325, 1550, 1775, 2000
P_μ	0.441, 0.505, 0.616, 0.768, 0.911, 1.136
P_c (mm/N)	0.294, 2.484, 0.978

Table 5.1: Mass, coefficient of Coulomb static friction and compliance property sets, as measured for a variety of blocks and surfaces.

5.1.5 The block topple-slide task

Toppling, as defined in [44], consists of two high-level phases: rolling and settling. In the rolling phase, the robot pushes the block up onto a toppling edge, which is perpendicular to the robot's movement, until the center of mass of the block is directly above the edge. During the settling phase, the block falls under gravity and lands on a new face before coming to rest.

As it is difficult to ensure the block's center of mass is above the edge following the rolling phase, the prescribed motion is developed so as to have the robot maintain contact with the block throughout the settling phase, to the extent that this is possible.

Once the block has settled, the robot then proceeds to slide the block across the surface of the table until the block has come to a stop back at its initial pose. Figure 1.1 depicts the topple-slide task with a sequence of images.

5.2 Procedure

5.2.1 Learn the task trajectory

A human expert demonstrates the topple-slide trajectory (Figure 1.1) via kinesthetic teaching. The robot is fixed to the table so as to not introduce additional variance in the recorded data due to base motion. The demonstrating user performs the task in about 6 s. The motion is then manually tuned so that the reference trajectory succeeds for the topple-slide task for a variety of combinations of block mass, coefficient of friction, and compliance (see section 5.1.1).

5.2.2 Record sensory dataset

The prescribed motion is repeated over a series of trials $h \in [1 \cdots n_h]$ for each property set $p \in P$, yielding the complete raw sensory data set D. We use $n_h = 20$ repeated trials. Before training our model, we pre-process the data as follows. The sensory data is resampled to 5 Hz after applying a 200 ms mean box filter.

We whiten each data set to support meaningful comparisons between sensors – by shifting data collected from each sensor to have zero mean and a variance of one – across all trials $h \in [1 \cdots n_h]$ and property sets $p \in P$. In our experiments we use $n_s = 110$ sensors across $n_t = 18$ time samples. This yields a complete input vector of size $n = 1980$ for each manipulation trial.

5.2.3 Feature selection

Following the equations in section 3.2.1, we compute the Γ for each element in $\mathbf{x}$. We select Γ_{min} so as to select $0.1 \times n$ features.

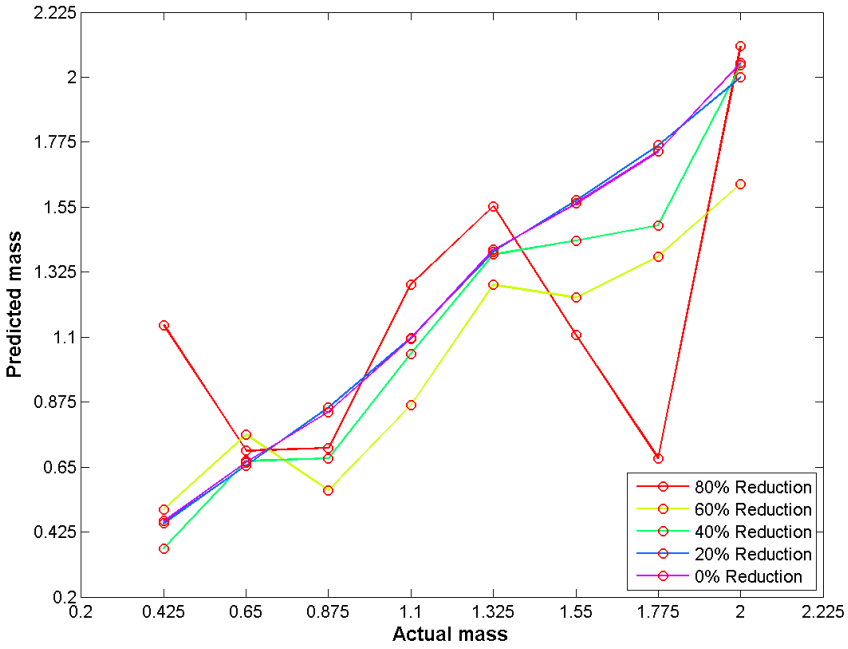


Figure 5.1: Effect of varying degrees of PLS dimensionality reduction on mass estimation performance. A 20% reduction is achievable with trivial loss in estimation quality.

5.2.4 Partial least squares modeling

Following PLS, we obtain the β coefficients. We can make the representation more compact by further choosing only the β coefficients of largest magnitude. In practice, we are able to make a further reduction of around 40% without any significant impact on the prediction accuracy, as shown in Figure 5.1. The first row uses all features and no PLS reduction. The second row uses a reduced set of 5% selected features without PLS reduction. The third row uses 5% selected features, followed by 40% PLS reduction.

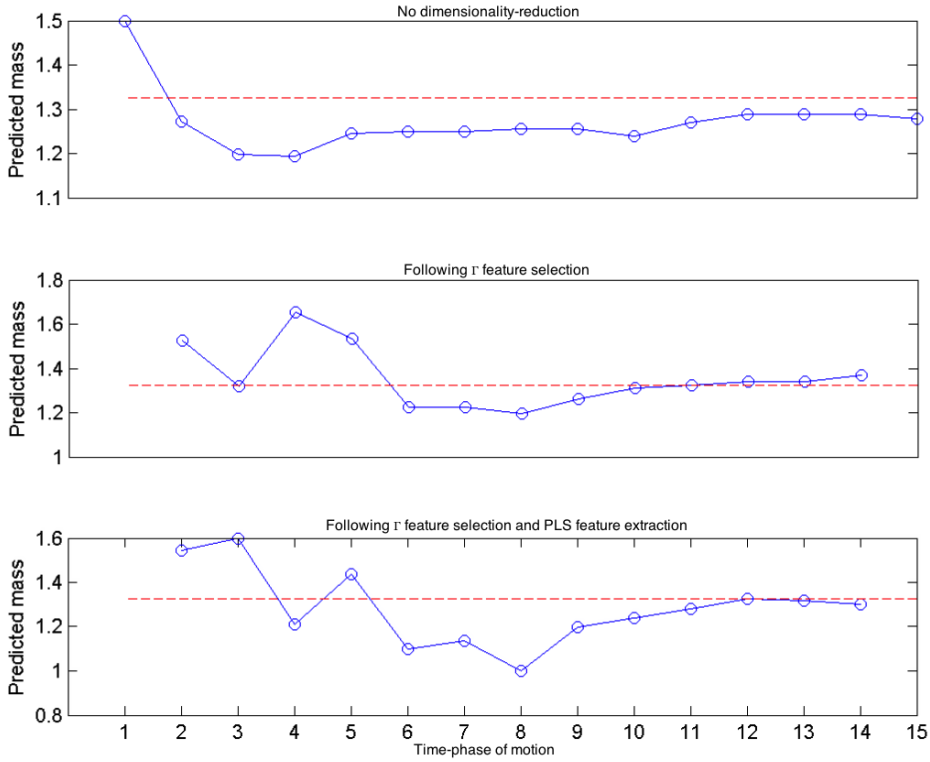


Figure 5.2: Online estimation of the mass from sensory data using varying degrees of dimensionality reduction. The estimated mass of the block are shown at various blue points throughout the motion. The dotted red horizontal line denotes the actual mass of the block.

5.2.5 Online prediction

We start by parsing the entire motion into a series of key time-phases, t^*. We define each $t^* \in T$ as a time-phase wherein at least K sensors have received a Γ larger than a certain value. In practice, we choose a minimum Γ so that $K = 0.1 \times n_s$. By training separate models in this fashion, we are able to make predictions as soon as the robot enters any phase of the motion involving selected features. Figures 5.2 and 5.3 demonstrate the prediction performance on-board the robot as it executes the task.

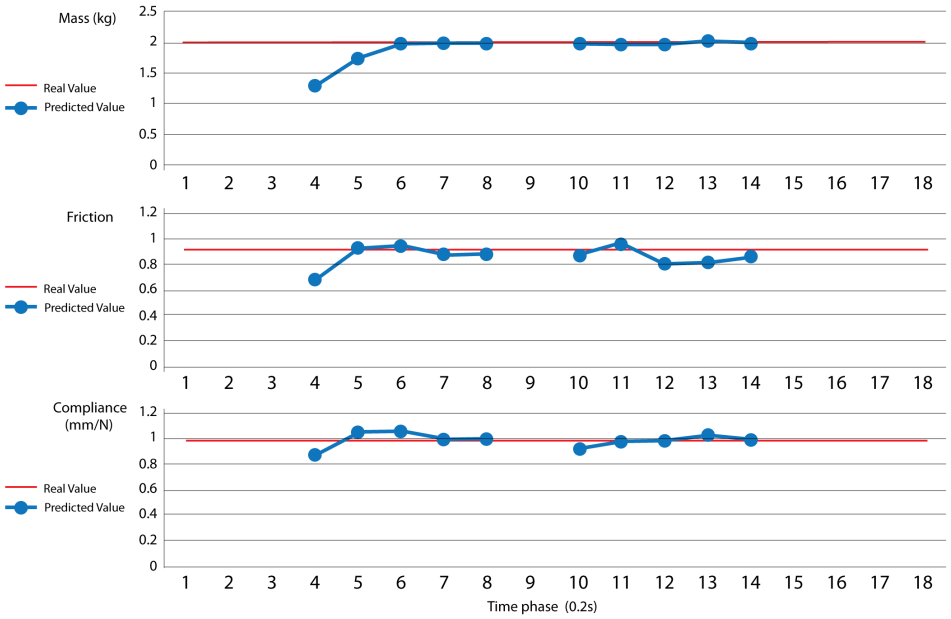


Figure 5.3: Online estimation of mass, friction and compliance from sensory data following Γ feature selection and PLS feature extraction. Time-phases 1 through 3, 9, and 15 through 18 are ignored since sensor readings during these time-phases do not provide any information with respect to distinguishing environment properties, i.e. their respective Γ values are below Γ_{min}.

5.2.6 Calculating environment properties

In order for the robot to discover a mapping between sensor readings and environment properties, a unique numerical approximation of the underlying property must be calculated. See Figure 5.4 for a graphical overview of how the coefficient of friction and compliance are calculated and Figure 5.5 for a photo of each of the environmental properties.

Mass The mass of each unit is approximated using an off-the-shelf kitchen scale.

Friction coefficient We first place the foam block atop a surface lined with the frictional material we are measuring. We then gradually incline the surface until the block begins to slide. We capture the inclination at the point of sliding as θ. We finally approximate the coefficient of static friction of the frictional material to be

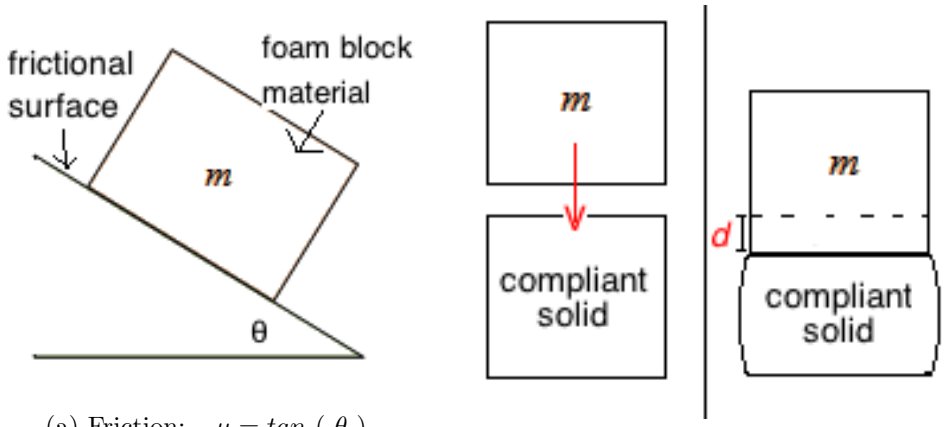
$$\mu = tan\ (\ \theta\ ),$$

which we also assume to be a fair approximation to the coefficient of kinetic friction.

Compliance As an estimate of the compliance of each type of foam, we set a rigid solid of known mass m atop a solid block of the foam we are measuring. The dimensions of the rigid solid and the foam solid are identical. We then measure the compressional displacement, d, of the top of the foam solid using standard calipers. Finally, we approximate the compliance of the foam to be

$$c = d\ /\ (\ m\ \cdot\ g\ ),$$

where g is acceleration due to gravity.



(a) Friction: $\mu = tan\ (\ \theta\ )$

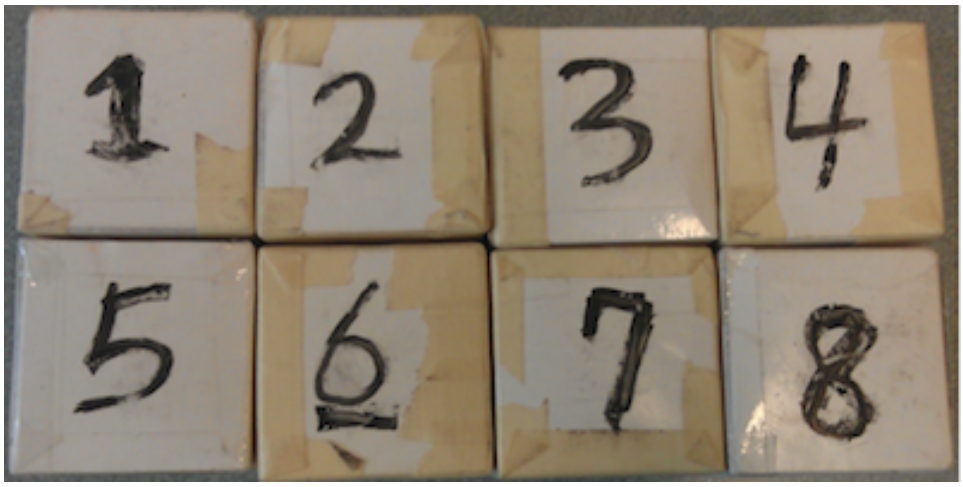
(b) Compliance: $c = d\ /\ (\ m \cdot g\ )$

Figure 5.4: Calculating numerical representations of environment properties (a) coefficient of friction, μ, and (b) compliance, c.

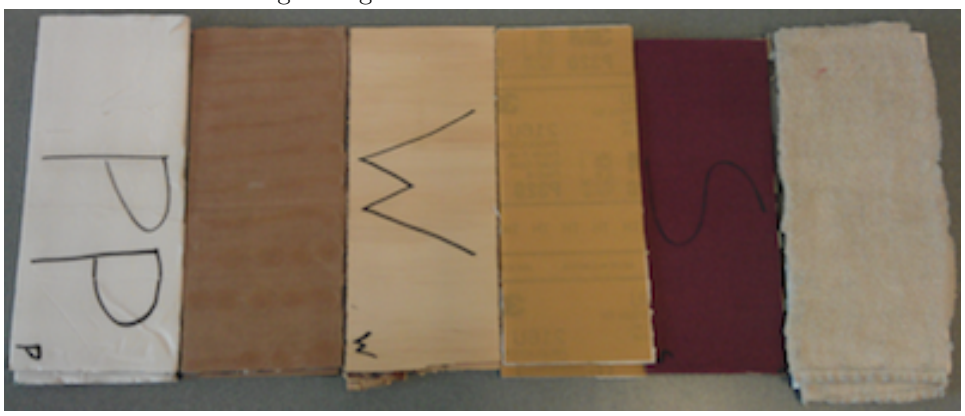
5.3 Results

In what follows below, we comment on topple-slide experiments carried out with the robot hand, as well as with the spherical probe. We also encourage the reader to watch the supplemental video associated with this thesis.

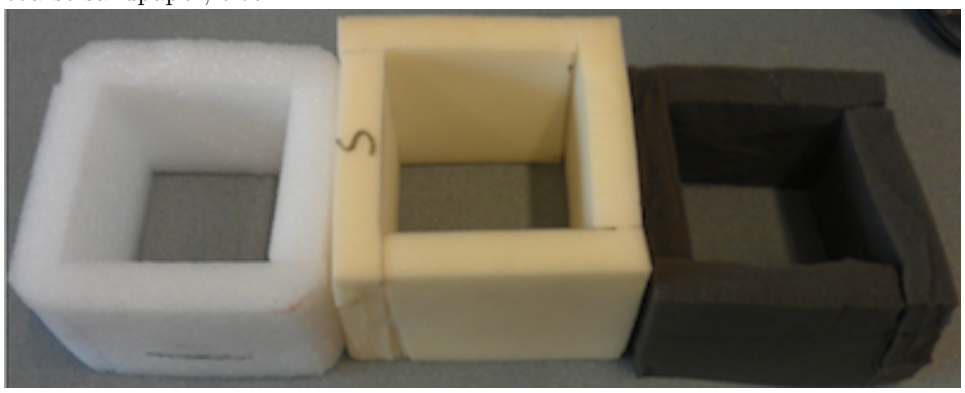
Γ selection helps focus attention on specific sensors and motion-phases that are particularly likely to provide information useful to predicting environment properties. Figure 5.6 illustrates the selected features for the topple-slide task as executed by the robot arm with either the Hand or spherical probe as end effectors (see Figure 5.7). Notice how clusters of $\mathbf{x}^*$ can be interpreted as defining important sensory events in the task sequence, which the robot should pay most attention to. The yellow shaded region corresponds to the topple phase and the blue shaded region corresponds to the slide phase. The motion phases where the arm is not in contact with the object are identified as being unimportant, as are the phases that mark the beginning and end of both of the topple and sliding phases. In terms of sensors, the joint velocities, jv_n, are generally unimportant, with the exception of joint 6. Joints 2, 4, and 6 provide task-relevant information in their sensed torques and positions. Similar results are also obtained with the full



(a) Eight units of 0.225kg mass which are used to vary the mass of the manipulated foam block from 0.445kg to 2kg.



(b) Six surface frictions (from left to right): paper, plastic, wood, fine-sandpaper, coarse-sandpaper, cloth.



(c) Three levels of compliance (from left to right): ethafoam, seafoam, greyfoam.

Figure 5.5: Environment properties used in experiments.

.

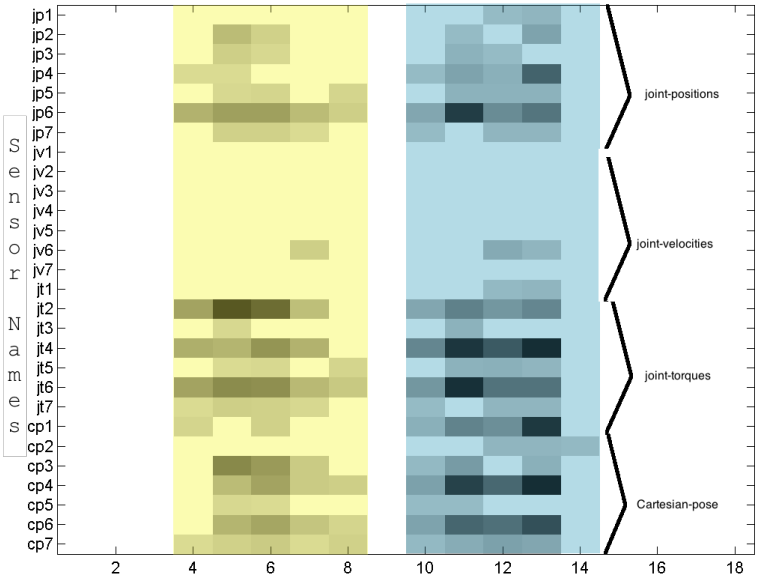
hand attached to the robot arm, in which case there are over a hundred sensors sampled across 18 time phases of the motion. With the hand in place, the key sensors are determined as being the task wrench ω, as measured by the force-torque sensor, the fingertip torques, f, and the fingertip tactile readings, a.

It may be noted that a simple contact/no-contact feature identifier might yield similar segmentations of the overall task in this case. However, this would require an explicit model that extracts contact information from sensory inputs. Our method identifies key points in the motion without any manually-tuned sensory features.

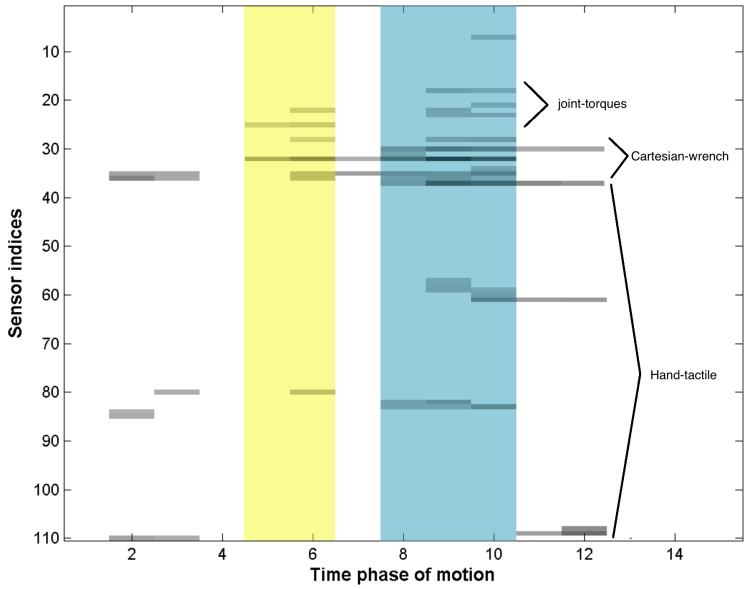
To determine the impact of the Γ feature selection, we compare mass predictions obtained using the inclusion of all features, i.e., no feature selection, and those obtained when Γ feature selection is used to select a subset of 5% of the the original features. This is applied to the manipulation task as executed using the spherical probe. In both cases, a non-reduced partial least squares model is constructed and leave-one-out-cross validation (LOOCV) test is considered for performance evaluation. As shown in Table 5.2, the result produced using the significantly reduced subset of input features is in most tests better than that obtained when using all the features and accuracy improves to within 1 measurement unit ($\pm$ 112.5 g) for all tests. Also, as can be seen in Table 5.3, applying up to 20% Γ feature selection to the incoming datastreams enables real-time operation in terms of both data-transfer bandwidth and prediction runtime. Tradeoff calculations assume 1Mbit CANBus, 16MHz dedicated processing speed and a real-time control loop frequency of 500Hz. FLOPs are calculated using standard inner-product vector multiplication complexity of $2n - 1$ for each of the three property predictions.

If desired, a fixed subset of the largest computed partial least squares coefficients can be used for the final prediction, instead of the full set, β. In practice, a 40% reduction in the number of coefficients yields only a minimal reduction in the quality of the prediction.

To validate our choice of partial least squares (PLS), we compare the results against three other methods: principal component regression (PCR),

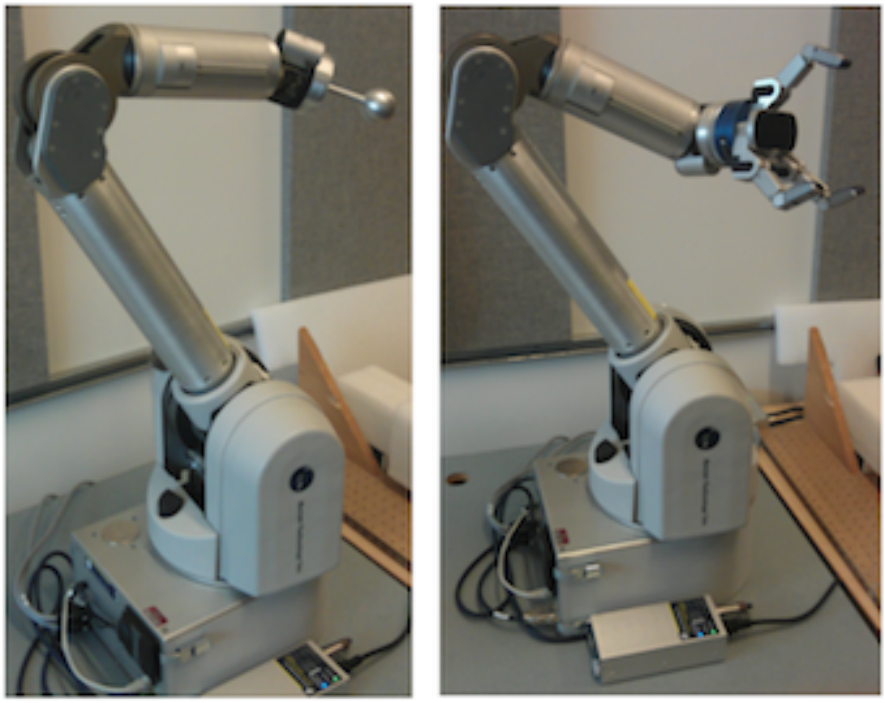


(a) Joint-velocities are not selected due to high noise.



(b) Joint-torques, Cartesian-wrench and Hand-tactile selected for providing most task-relevant information to the system.

Figure 5.6: Visualization of features selected according to the task variance ratio, Γ, of data collected using (a) robot with spherical probe and (b) robot with BarrettHand. Colour is added to signify task-phases. Gaps between task-phases signify an absence of task-relevant information within corresponding time-phases.



(a) (b)

Figure 5.7: WAM robot with (a) spherical probe and (b) BarrettHand as end-effectors.

	Prediction RMSE (g)	
Mass (g)	PLS + Γ	PLS Only
425	2.40	75.7
650	13.9	86.9
875	1.40	51.1
1100	15.8	40.8
1325	7.20	43.1
1550	0.20	37.2
1775	9.00	147
2000	11.0	7.20

Table 5.2: Effect of 5% Γ feature selection on LOOCV block mass prediction root mean squared error (RMSE).

least squares regression (LSR), and naive Bayes classification (NBC). For LSR, we regularize the solution using ridge regression. For NBC, a new sensory stream is treated as input to a classification problem, and the classifier is constructed using naive Bayes that assumes that all features in $\mathbf{x}$ are independent. Using the repeated trials for the given set of environment properties, a normal distribution is constructed for each element of $\mathbf{x}$, and the likelihood of a new value of $\mathbf{x}$ belonging to the same class is simply modeled as the product of the individual element likelihoods. The environment properties of the most likely class are then returned as the prediction. All four methods are evaluated using LOOCV, and are applied to $\mathbf{x}^*$, i.e., after Γ feature selection. The results for mass prediction show that PLS yields the best predictions, with respective mean errors for PLS, PCR, and LSR and NBC of 33.3, 56.4, and 84.9, and 282.6, with respective standard deviations of 4.2, 8.4, 14.6 and 83.2 as measured in *grams*.

Figure 5.2 illustrates online mass prediction results. The robot is able to make predictions at any key time-phase, $t^\star$, each characterized by a high Γ for many sensors. This is accomplished through building multiple models, each spanning the data from the start of the motion to some $t \in t^\star$.

These results are obtained for the case of training on data for $m = \{425, 650, 875, 1100, 1550, 1775, 2000\}$ as measured in g and is then tested using sensory data obtained for $m = 1335$ g. The result shows predictions

being made using increasingly fewer selected features or reduced PLS dimensions, as noted in the caption. Also, the predictions improve as the motion progresses and more selected features are observed.

Our framework is also robust to feature noise. To demonstrate this, we run experiments where we introduce large amounts of synthetic noise into the sensory data streams before feature selection and after data whitening (see Section 5.2.2). As shown in Table 5.4, LOOCV prediction RMSE increases smoothly as feature noise increases. Note that for even small amounts of additive noise ($\sigma^2 \approx 0.1$), PLS fails to produce meaningful results in the absence of Γ feature selection.

Γ data selection:	5%	10%	20%	40%	70%	100%
# FLOPs (approx.):	32	65	131	263	461	659
Runtime (ms):	0.25	0.5	1.0	2.1	3.6	5.1
Bandwidth (bit):	352	704	1408	2816	4928	7040
Maximum error (g):	69.5	112.7	96.3	111.4	107.9	188.3
Real-time satisfied?	T	T	T	T/F	F	F
Bandwidth satisfied?	T	T	T	F	F	F
Accuracy satisfied?	T	T/F	T	T/F	T	F

Table 5.3: Bandwidth/runtime/accuracy tradeoff following different amounts of Γ selection. Optimal tradeoff is achieved when between 5% and 20% of the data is selected using Γ.

5.4 Discussion

Our prediction framework uses unlabeled sensory data streams, collected during a manipulation task, to make reliable real-time predictions about environment properties that cannot be visually observed, i.e., mass, friction, and compliance, given the existence of relevant training examples. Sensors or motion phases that are observed to be noisy are readily discounted by our method. The results show that the task variance ratio, Γ, provides a

σ^2	RMSE (g)
0.0	9.4
0.5	55.1
1.0	160.7
1.5	172.8
2.0	255.5

Table 5.4: Effect of different levels of additive Gaussian noise $\mathcal{N}(0, \sigma^2)$ on the sensory input data for LOOCV mass prediction RMSE ($m = 1335$ g). Accuracy degrades smoothly as sensor noise increases.

simple means for feature selection, identifying important sensors and motion-phases supporting real-time predictions, and which furthermore improves the resulting partial least squares predictions.

While predicting environment properties from labeled training data could be an obvious application of linear regression, this is in practice problematic because the training data for our scenario consists of a relatively small sample size (low hundreds) embedded in a high dimensional space: $\mathbf{x}$ can contain thousands of sensory measurements. Furthermore, the large number of measurements required to make accurate predictions prohibits real-time operation.

Although PLS utilizes a dimension reduction technique by using a few latent factors, it cannot avoid the sample size issue since a it has been proven that a reasonable sample size relative to the number of parameters is required to estimate sample covariances consistently [8]. Thus, PLS works best under the conditions of large sample sizes and/or small numbers of input variables.

When combined with Γ feature selection, our results show PLS superior to other regression algorithms which do not leverage input and output correlations in their calculations. Unlike PCR, PLS uses $\mathbf{y}$ (in addition to $\mathbf{x}$) to construct its principal directions. Thus, its solution path is a nonlinear function of $\mathbf{y}$ [23]. In addition to outperforming the other benchmark prediction methods for the task, PLS also provides a further opportunity for dimensionality reduction if desired.

Due to the model-free nature of our approach, the prediction framework works for virtually any combination of sensor modalities, including tactile-pressure, Cartesian wrench and joint-torque, which enables easy experimentation to determine the optimal tradeoff between sensor usage and prediction accuracy.

One limitation of our approach is that Γ can be misleading, such as in the case of noise-free features that also exhibit significant non-linearities with respect to the properties being predicted. Another drawback is that the learned predictive model remains specific to the prescribed motion used for the task and the specific kinematics and dynamics of the robot and environment it trained in. The current remedy is to incorporate further training data from which to build the model when changes to the robot or its motion take effect. In future work, we intend to examine how the predictive model can be transferred to new settings [52].

Our framework can also be leveraged in multiple ways in order to detect anomalous events. Rapid changes in the predicted environment properties, such as object compliance, is a signal of an anomaly. Also, implicit in the computation of Var^{enviro} is a model of what value a sensory feature should have at a given point in the motion. This allows a motion anomaly to be signaled if a number of sensors each begin to signal anomalous values at a given point in time, or a sensor anomaly to be signaled if a single sensor begins to consistently produce anomalous readings.

Chapter 6

Conclusions and Future Work

In this thesis, we present the challenge of predicting properties of real-world environments using high-dimensional haptic sensory data from the perspectives of both biological and robotic systems.

6.1 Summary

We begin in Chapter 2 by presenting insights into the problem from the established robotics, neuroscience and physiology literature. Next, we define the prediction problem more formally in Chapter 3. In Chapter 4 we provide an overview on the software framework used to control the physical WAM/Hand system and collect data in support of experiments. We then introduce a model-free approach to the prediction of example environment properties – namely object mass, friction, and compliance – during the course of a non-prehensile topple-slide manipulation task in Chapter 5.

Given appropriate data from example manipulations with known environment properties, the method presented in this thesis extracts information from unlabeled sensory data collected over the course of a new manipulation. We demonstrate that our novel metric, known as the Task Variance Ratio (TVR), identifies important features, sensors and motion-phases. Using the TVR metric combined with the PLS regression method, we obtain accurate predictions in real-time using only 3% of the sensory input data from the robot.

6.2 Limitations and Future Directions

One significant limitation of the predictive framework is the need for a prescribed motion that can already succeed at the task despite variations in the environment properties that we seek to predict.

An important direction for future work will be to investigate the tight integration of prediction and adaptation into the framework. With knowledge (learned or provided) of how to adapt the topple-slide task for heavier or more compliant blocks, this could readily be used to enlarge the range of variations that can be coped with.

Surprise-and-adapt

We have devised a preliminary model-based motion adaptation approach to succeed in environments where the prescribed motion fails. Under the assumption that the robot has access to tactile pressure and/or fingertip torque sensors, we modify the orientation of the robot's wrist about the axis parallel to the manipulated block, so that the pressure readings at the fingertips track an appropriate profile provided by an expert.

To deal with sensor noise, we consider a history of sensor readings of size K_r. If sensor readings fall below a threshold for at least K_r timesteps, the orientation of the wrist increases – thus applying more pressure to the block. Similarly, if sensors consistently read above a threshold, the orientation of the wrist decreases – releasing pressure. See Appendix A for a pseudocode of this operation.

This scheme is successful for the topple-slide task when particularly small perturbations in the environment are experienced, such as a relatively small (within two units) change in object mass, but fails with large perturbations.

An interesting future direction is to devise a model-free approach in which the robot discovers for itself that a lack of pressure at its fingertips means that its wrist orientation should increase (in addition to other relevant mappings between sensor readings and adaptive motions). This knowledge would have to come from the data and would most likely require additional sensors and/or some form of supervision, e.g. from a human or a camera.

Time-synchronized motions

A related limitation is that our current sensory features are all time-indexed, i.e., there is an assumption that the current phase of the motion is tightly coupled to the current time. In future work, we would like to couple the phase estimate more tightly to the actual motion via available sensory observations.

Task-parameter generalization

A last key limitation is that because of the model-free nature of the current approach, the prediction procedures do not generalize well to changes in the task kinematics or dynamics. We aim to develop parameterized versions of the predictive model in order to allow for such generalization.

Leveraging physics-based simulation

We are are also interested in exploring how simulations with only qualitative accuracy might be used to identify suitable sensors and motion phases in advance. This could be used to inform the types of sensors and their placement, as well as provide insight with respect to relevant time-steps at which to record data. Initial results in this direction are promising and thus point to a new use for simulations that are not necessarily tightly calibrated to the true kinematics and dynamics of the plant.

Bibliography

[1] Charu C Aggarwal. A framework for diagnosing changes in evolving data streams. In *Management of Data, Proceedings of the 2003 ACM SIGMOD International Conference on*, pages 575–586, 2003.

[2] Anelia Angelova, Larry Matthies, Daniel M Helmick, and Pietro Perona. Dimensionality reduction using automatic supervision for vision-based terrain learning. In *Proc. Robotics: Science and Systems*, 2007.

[3] Eric Bair, Trevor Hastie, Debashis Paul, and Robert Tibshirani. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101(473), 2006.

[4] Antonio Bicchi, Davide Dente, and Enzo Pasquale Scilingo. Haptic illusions induced by tactile flow. In *EuroHaptics, Proceedings of*, pages 314–329, 2003.

[5] MKO Burstedt, Benoni B Edin, and Roland S Johansson. Coordination of fingertip forces during human manipulation can emerge from independent neural networks controlling each engaged digit. *Experimental Brain Research*, 117(1):67–79, 1997.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15:1–15:58, July 2009.

[7] Eris Chinellato. *Visual neuroscience of robotic grasping*. PhD thesis, Universitat Jaume I, 2008.

[8] Hyonho Chun and Sündüz Keleş. Sparse partial least squares regression for simultaneous dimension reduction and variable selection. *Jour-*

nal of the Royal Statistical Society: Series B (Statistical Methodology), 72(1):3–25, 2010.

[9] Mark R Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *Robotics and Automation, IEEE Transactions on*, 5(3):269–279, 1989.

[10] Mark R Cutkosky and Robert D Howe. Human grasp choice and robotic grasp analysis. In *Dextrous robot hands*, pages 5–31. Springer, 1990.

[11] Mark R. Cutkosky, Robert D. Howe, and William R. Provancher. Force and tactile sensors. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 455–476. Springer Berlin Heidelberg, 2008.

[12] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary R Bradski. Self-supervised monocular road detection in desert terrain. In *Proc. Robotics: Science and Systems*, volume 38, 2006.

[13] Hao Dang and Peter K Allen. Tactile experience-based robotic grasping. In *Workshop on Advances in Tactile Sensing and Touch based Human-Robot Interaction (HRI)*, 2012.

[14] Hao Dang, Jonathan Weisz, and Peter K Allen. Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics. In *Robotics and Automation (ICRA), Proceedings of the 2011 IEEE International Conference on*, pages 5917–5922, 2011.

[15] Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Interface of Statistics, Computing Science, and Applications, Proceedings of the Symposium on*, 2006.

[16] Peter Dayan and Nathaniel D Daw. Decision theory, reinforcement learning, and the brain. *Cognitive, Affective, & Behavioral Neuroscience*, 8(4):429–453, 2008.

[17] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krger, and J. Piater. Learning grasp affordance densities. *Paladyn, Journal of Behavioral Robotics*, 2(1):1–17, 2011.

[18] John M Elliott and KJ Connolly. A classification of manipulative hand movements. *Developmental Medicine & Child Neurology*, 26(3):283–296, 1984.

[19] Ronald S Fearing. Tactile sensing for shape interpretation. In *Dextrous robot hands*, pages 209–238. Springer, 1990.

[20] Javier Felip, Jose Bernabé, and Antonio Morales. Emptying the box using blind haptic manipulation primitives. In *Intelligent Robots and Systems (IROS), Proceedings of the 2011 IEEE/RSJ International Conference on*, 2011.

[21] J Randall Flanagan, Miles C Bowman, and Roland S Johansson. Control strategies in object manipulation tasks. *Current Opinion in Neurobiology*, 16(6):650–659, 2006.

[22] J Randall Flanagan and Roland S Johansson. Action plans used in action observation. *Nature*, 424(6950):769–771, 2003.

[23] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001.

[24] Zoubin Ghahramani, Geoffrey E Hinton, et al. The em algorithm for mixtures of factor analyzers. Technical report, Technical Report CRG-TR-96-1, University of Toronto, 1996.

[25] Corey Goldfeder, Matei Ciocarlie, Hao Dang, and Peter K Allen. The columbia grasp database. In *Robotics and Automation (ICRA), Proceedings of the 2009 IEEE International Conference on*, pages 1710–1716, 2009.

[26] Stacey L Gorniak, Vladimir M Zatsiorsky, and Mark L Latash. Manipulation of a fragile object. *Experimental brain research*, 202(2):413–430, 2010.

[27] G. Heidemann and M. Schopfer. Dynamic tactile sensing for object identification. In *Robotics and Automation (ICRA), Proceedings of the 2004 IEEE International Conference on*, volume 1, pages 813–818, april 2004.

[28] Albert Hein and Thomas Kirste. Unsupervised detection of motion primitives in very high dimensional sensor data. In *Proceedings of the 5th Workshop on Behaviour Monitoring and Interpretation, BMI*, 2010.

[29] JJ Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376(6535):33–36, 1995.

[30] Rachel Hornung, Holger Urbanek, Julian Klodmann, Christian Osendorfer, and Patrick van der Smagt. Model-free robot anomaly detection. In *Intelligent Robots and Systems (IROS), Proceedings of the 2014 IEEE/RSJ International Conference on*, pages 3676–3683, 2014.

[31] Robert D. Howe. Tactile sensing and control of robotic manipulation. *Advanced Robotics*, 8(3):245–261, 1993.

[32] Cheng-Lung Huang and Chieh-Jen Wang. A ga-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with applications*, 31(2):231–240, 2006.

[33] Roland S Johansson and Kelly J Cole. Grasp stability during manipulative actions. *Physiology and Pharmacology, Canadian journal of*, 72(5):511–524, 1994.

[34] Roland S Johansson and J Randall Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, 10(5):345–359, 2009.

[35] Roland S Johansson, Göran Westling, Anders Bäckström, and J Randall Flanagan. Eye-hand coordination in object manipulation. *Neuroscience*, 21(17):6917–6932, 2001.

[36] RS Johansson and G Westling. Roles of glabrous skin receptors and sensory memory in automatic control of precision grip when lifting rougher or more slippery objects. *Experimental Brain Research*, 56(3):550–564, 1984.

[37] Makoto Kaneko, M Wada, H Maekawa, and K Tanie. A new consideration on tendon-tension control system of robot hands. In *Robotics and Automation (ICRA), Proceedings of the 1991 IEEE International Conference on*, pages 1028–1033, 1991.

[38] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Very Large Data Bases, Proceedings of the 30th International Conference on*, volume 30, pages 180–191, 2004.

[39] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[40] Mark H Lee and Howard R Nicholls. Tactile sensing for mechatronicsa state of the art survey. *Mechatronics*, 9(1):1–31, 1999.

[41] Fabio Leoni, Massimo Guerrini, Cecilia Laschi, Davide Taddeucci, Paolo Dario, and Antonina Starita. Implementing robotic grasping tasks using a biological approach. In *Robotics and Automation (ICRA), Proceedings of the 1998 IEEE International Conference on*, volume 3, pages 2274–2280, 1998.

[42] Nathan F Lepora, Uriel Martinez-Hernandez, Hector Barron-Gonzalez, Mat Evans, Giorgio Metta, and Tony J Prescott. Embodied hyperacuity from bayesian perception: Shape and position discrimination with an icub fingertip sensor. In *Intelligent Robots and Systems (IROS), Proceedings of the 2012 IEEE/RSJ International Conference on*, pages 4638–4643, 2012.

[43] David Lieb, Andrew Lookingbill, and Sebastian Thrun. Adaptive road following using self-supervised learning and reverse optical flow. In *Proc. Robotics: Science and Systems*, pages 273–280, 2005.

[44] K.M. Lynch. Toppling manipulation. In *Robotics and Automation (ICRA), Proceedings of the 1999 IEEE International Conference on*, volume 4, pages 2551 –2557 vol.4, 1999.

[45] Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. *Robotics & Automation Magazine, IEEE*, 11(4):110–122, 2004.

[46] Pabitra Mitra, CA Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312, 2002.

[47] Chellappa Muthukrishnan, David Smith, Donald Myers, Jack Rebman, and Antti Koivo. Edge detection in tactile images. In *Robotics and Automation (ICRA), Proceedings of the 1987 IEEE International Conference on*, volume 4, pages 1500–1505, 1987.

[48] Yoshihiko Nakamura, Kiyoshi Nagai, and Tsuneo Yoshikawa. Mechanics of coordinative manipulation by multiple robotic mechanisms. In *Robotics and Automation (ICRA), Proceedings of the 1987 IEEE International Conference on*, volume 4, pages 991–998, 1987.

[49] John R Napier. The prehensile movements of the human hand. *Journal of bone and Joint surgery*, 38(4):902–913, 1956.

[50] Kenneth J Overton and Thomas Williams. Tactile sensation for robots. In *Artificial Intelligence (IJCAI), International Joint Conferences on*, pages 791–795, 1981.

[51] Lucia Pais, Keisuke Umezawa, Yoshihiko Nakamura, and Aude Billard. Learning robot skills through motion segmentation and constraints extraction. In *HRI Workshop on Collaborative Manipulation*, 2013.

[52] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[53] Sivalogeswaran Ratnasingam and T Martin McGinnity. A spiking neural network for tactile form based object recognition. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 880–885, 2011.

[54] Sivalogeswaran Ratnasingam and TM McGinnity. Object recognition based on tactile form perception. In *Robotic Intelligence In Informationally Structured Space (RiiSS), 2011 IEEE Workshop on*, pages 26–31, 2011.

[55] ND Ring and DB Welbourn. Paper 8: A self-adaptive gripping device: Its design and performance. In *Institution of Mechanical Engineers, Proceedings of the*, volume 183, pages 45–49, 1968.

[56] Joseph M Romano, Kaijen Hsiao, Günter Niemeyer, Sachin Chitta, and Katherine J Kuchenbecker. Human-inspired robotic grasp control with tactile sensing. *Robotics, IEEE Transactions on*, 27(6):1067–1079, 2011.

[57] Lawrence K Saul and Sam T Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *The Journal of Machine Learning Research*, 4:119–155, 2003.

[58] N Sawasaki, M Inaba, and H Inoue. Tumbling objects using a multi-fingered robot. In *Proceedings of the 20th International Symposium on Industrial Robots and Robot Exhibition*, pages 609–616, 1989.

[59] ClaudioGentili Scilingo, Lorenzo Sani, Vincenzo Positano, Filomena M Santarelli, Mario Guazzelli, James V Haxby, Luigi Landini, Antonio Bicchi, and Pietro Pietrini. Perception of visual and tactile flow activates common cortical areas in the human brain. In *EuroHaptics 2004, Proceedings of*, pages 290–292, 2004.

[60] David M Siegel. Finding the pose of an object in a hand. In *Robotics and Automation (ICRA), Proceedings of the 1991 IEEE International Conference on*, pages 406–411, 1991.

[61] Pavan Sikka, Hong Zhang, and Steve Sutphen. Tactile servo: Control of touch-driven robot motion. In *Experimental Robotics III*, pages 219–233. Springer, 1994.

[62] Boris Sofman, Ellie Lin, J Andrew Bagnell, John Cole, Nicolas Vandapel, and Anthony Stentz. Improving robot navigation through self-supervised online learning. *Journal of Field Robotics*, 23(11-12):1059–1075, 2006.

[63] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. Statistical change detection for multi-dimensional data. In *Knowledge Discovery and Data Mining, Proceedings of the 13th ACM SIGKDD International Conference on*, pages 667–676, 2007.

[64] J.C. Spall and John A. Cristion. Model-free control of nonlinear stochastic systems with discrete-time measurements. *Automatic Control, IEEE Transactions on*, 43(9):1198–1210, 1998.

[65] Jan Steffen, Robert Haschke, and Helge Ritter. Experience-based and tactile-driven dynamic grasp control. In *Intelligent Robots and Systems (IROS), Proceedings of the 2007 IEEE/RSJ International Conference on*, pages 2938–2943, 2007.

[66] Barrett Technology. Libbarrett. `http://support.barrett.com:8080/wiki/WAM/InternalPC104Setup`, 2011. Accessed: 2014-09-30.

[67] Barrett Technology. Wam internal pc/104 configuration. `http://barrett.com:8080/wiki/Libbarrett`, 2012. Accessed: 2014-09-30.

[68] Johan Tegin and Jan Wikander. Tactile sensing in intelligent robotic manipulation–a review. *Industrial Robot*, 32(1):64–70, 2005.

[69] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[70] Randall D Tobias et al. An introduction to partial least squares regression. In *SAS Users Group International (SUGI), Proceedings of the 20th*, pages 2–5, Orlando, FL, USA, 1995.

[71] William Townsend. The barretthand grasper–programmably flexible part handling and assembly. *Industrial Robot: An International Journal*, 27(3):181–188, 2000.

[72] William T Townsend and J Kenneth Salisbury. Mechanical design for whole-arm manipulation. In *Robots and Biological Systems: Towards a New Bionics?*, pages 153–164. Springer, 1993.

[73] Marc R Tremblay and Mark R Cutkosky. Estimating friction using incipient slip sensing during a manipulation task. In *Robotics and Automation (ICRA), Proceedings of the 1993 IEEE International Conference on*, pages 429–434, 1993.

[74] Å B Vallbo, RS Johansson, et al. Properties of cutaneous mechanoreceptors in the human hand related to touch sensation. *Human Neurobiology*, 3(1):3–14, 1984.

[75] Kenji Yamanishi and Jun-ichi Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Knowledge Discovery and Data Mining, Proceedings of the 8th ACM SIGKDD International Conference on*, pages 676–681, 2002.

[76] Yuan-Fei Zhang and Hong Liu. Tactile sensor based varying contact point manipulation strategy for dexterous robot hand manipulating unknown objects. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4756 –4761, Oct 2012.

[77] Dmitrii Nikolaevich Zubarev, P Gray, and PJ Shepherd. *Nonequilibrium statistical thermodynamics*. Consultants Bureau New York, 1974.

Appendix A

Surprise-and-adapt Pseudocode

```
void operate()  //repeat at 500Hz
{   //Consider a history of thresholded differences.
    if ( sensor_value - expected_value > threshold_p )
        history.append_front ( 1 )
    else if ( sensor_value - expected_value < threshold_n )
        history.append_front ( -1 )
    else
        history.append_front ( 0 )
    //'Surprise' iff at least K contiguous unexpected readings.
    if ( sum ( history [ 0 : K ] ) == K )   //too much pressure
        decrease_wrist_angle()
    if ( sum ( history [ 0 : K ] ) == -K )  //too little pressure
        increase_wrist_angle()
}
```

Program 2: Realtime *operate* method pseudocode for the surprise-and-adapt realtime system module. See Section 4.5 for general details on Libbarrett realtime systems development.