Robotic Grasp Control using Tactile Feedback

August John Bull
University of Utah

UUCS-20-009

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

7 May 2020

Abstract

As we move towards more autonomous robots, object interaction and manipulation remains difficult for robots. Grasping approaches vary from learning-based to traditional control means, each with their own challenges. Learning-based methods perform well on specific tasks, but struggle to move to other tasks. Approaches using control require a thorough predetermined model, which is not feasible for all tasks. Humans mastered dexterous manipulation and tool use through millions of years of evolution. What can we learn from human capabilities? We provide background in robotics and human inspiration necessary for a grasp controller using only tactile sensing on a robot. This controller encodes the whole pick-and-place operation, from grasp initiation to careful placement. We detail the implementation of the controller and provide insights to improvement by extracting events from sensory data. We present our work as a key step towards making dexterous robots.

ROBOTIC GRASP CONTROL

USING TACTILE FEEDBACK

by

August John Bull

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

In

Computer Science

Approved:

Tucker Hermans, PhD Thesis Faculty Supervisor	Ross Whitaker, PhD Director, School of Computing
Erin Parker, PhD Honors Faculty Advisor	Sylvia D. Torti, PhD Dean, Honors College

May 2020

Abstract

As we move towards more autonomous robots, object interaction and manipulation remains difficult for robots. Grasping approaches vary from learning-based to traditional control means, each with their own challenges. Learning-based methods perform well on specific tasks, but struggle to move to other tasks. Approaches using control require a thorough predetermined model, which is not feasible for all tasks. Humans mastered dexterous manipulation and tool use through millions of years of evolution. What can we learn from human capabilities? We provide background in robotics and human inspiration necessary for a grasp controller using only tactile sensing on a robot. This controller encodes the whole pick-and-place operation, from grasp initiation to careful placement. We detail the implementation of the controller and provide insights to improvement by extracting events from sensory data. We present our work as a key step towards making dexterous robots.

Contents

Chapter 1

Manipulation by Humans and Robots

Imagine the near future, where you sit for dinner at your favorite restaurant. A robot waiter brings your table drinks and begins handing them out. Tragedy strikes! The robot has set your drink down wrong, and risks spilling it! Your robot server is quick to act and corrects its error, like a veteran waitress, keeping your beverage in the cup. Robot grasping and manipulation has a rich background in heavily controlled situations, like when the robot can assume an object will not fall over after releasing it from its grasp. Alternatively, the complexity of manipulation is reduced by attaching a specific tool to a robot, or designing it around a specific task. However, this reduces the ability of the robot to perform other tasks well. Specificity is a compromise.

Robots are widespread in industry, from agriculture to hospitality [1]. Surgical robots promise precise procedures with teleoperation [2]. Assistive robots offer an alternative to retirement living for low-ability individuals who value their independence [3]. In the wake of the COVID-19 pandemic, robots provide a passable solution to the shortage of medical staff and reducing the risk of infecting health care workers.

Humans are able to use a range of tools due to our ability to manipulate them. Human-level dexterity allows for general tool use, making manipulation a highly valuable area of investigation. To achieve this in robots, several approaches may be taken with regards

to the robot's end-effector. Designers may opt for a specialized end-effector with various tools attached [4]. For a general-purpose robot, a robotic gripper is much better suited to general manipulation. Grippers come in various forms. Parallel jaw grippers are the most basic, offering simple control often with one motor [5]. Grippers with interphalangeal joints offer more versatility, but control each individual motor must be coordinated [6]. An underactuated robot hand has more joints than motors. This is achieved by having some pliable material or pivot act as the joint and have it change when the robot pose demands [7].

Humans have a number of nerve endings in the hand that give rich tactile information to the brain [8], [9]. This complex representation of grip force is key to the versatility of human grasping [8]. Additionally, human reflexes allow recovery of failed placements. Humans can reflexively regrasp an object if it unexpectedly stretches the muscles of the finger [10]. We believe this added layer of adaptability is key to making general-use robot systems. A similar reflex in a robot would allow it to stop objects falling from its grasp.

Research seeks to lessen the gap between human and robot capabilities by mimicking human senses and reactions [11]. This perspective on robot manipulation yields coded controllers that account for an object's malleability, or even breaking point. Mimicry requires perception. Visual information is useful for gaining location data of robots and target objects [12]–[14], but it contains no information on the stability or force of a grip. A camera cannot stop a robot from cracking an egg. We may use force-torque sensors in motors to estimate grip strength. We prefer robotic grippers come equipped with or the possibility to equip pressure sensors. These sensors give a direct representation of forces the robot applies to an object, and they come in a variety of robotic manipulators that vary in complexity. Robot manipulation is a difficult problem to solve, with entire university labs pursuing solutions involving optimization, learning and control. Determining how to successfully grasp an object is nontrivial. Finger position can vary widely, and there are often multiple feasible grasp types [15]. Beyond succesfully initiating a grasp is the question of

how to apply enough force to lift the object, but not so much that the object deforms or breaks.

This project seeks to widen the scope of how manipulators can pick up and replace objects by detecting grip strength and adjusting to loosen or strengthen as necessary. To do this we believe human inspiration is key due to the versatility and sensitivity of human grasps. This thesis provides an overview of techniques in robotic manipulation and compares them to human ability. It then introduces a method of actuating a robot hand based on tactile feedback to achieve careful grasp initiation and release. In implementing tactile feedback in the robot grasp model, we hope to take robotic manipulation closer to human levels of robustness.

1.1 Human Movement and Manipulation

Humans have many degrees of freedom in their hands between flexion and extension for each finger, adduction and abduction of the fingers, and opposition of the thumb. The advanced muscle and tendon structures of the hand make this level of expression possible [9]. The muscles of each finger span multiple joints, combining with the tendons to make a pulley-like system that allows for a number of finger poses independent of their orientation to the hand. We can keep a finger straight while flexing it relative to your palm thanks to extensor muscles on each interphalangeal joint. The muscles working in opposition, the flexor pollicis brevis and flexor pollicis longus, allow us to grasp objects and control what parts of the finger touch. These are only two sets of muscle agonist/antagonist pairs. The number of muscles and joints in the hand allow and facilitate immense dexterity. Combined with the palm, humans learn a myriad of grasps. Figure 1.1 provides examples of basic human grasp types.

More advanced than the musculoskeletal structure of the hand are the nerves that control it. The portion of the motor cortex dedicated to hand movement is roughly the size required

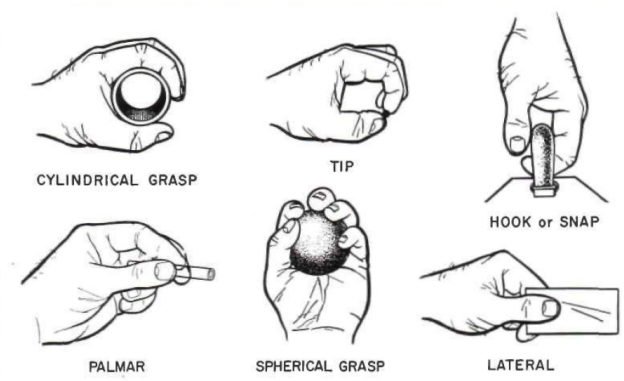


Figure 1.1: Basic human grasp types from Taylor and Schwarz [9, Fig. 3].

to control the arm, torso (trunk), and legs [9]. This increased demand makes sense when comparing the number of distinct muscle groups and number of movements possible in each structure.

Less obvious than the complexity of movements of the hand are the nerves that respond to tactile stimuli. The glabrous, or non-hairy skin on palms and fingertips has a high number of nerve endings, containing a variety of afferent nerves that send information to the brain. In particular there are four types of mechanoreceptors that work to provide a wide range of sensations: slow-adapting and fast-adapting nerve endings, each having two subtypes [8], [11]. These nerves vary in fidelity in both area and time, allowing humans to detect simple sensations like direct pressure to more advanced movements like movement parallel to the finger's surface.

The mechanics of human manipulation outside the anatomy are relatively simple. To grasp, we apply normal force to an object's surface such that the tangential force of friction is sufficient to overcome gravity. The deformation of the skin, fat and muscles of a human finger contributes to grasp capability as well: deformation allows for contact area to be compliant to the variations of a surface, ensuring good contact area.

1.2 Robot Movement

Moving a robot involves understanding its kinematics – the parameters that define the angles of the joints and orientations between them. In general each joint along an open-loop chain, such as an arm not grasping a fixed object, offers one degree of freedom, though underactuated joints are an exception to this. With 7-DOF robotic arms, a robot can reach nearly every point in its workspace [16]. Because some motors in robot systems may be able to turn infinitely, the vector space defined by the joint angles of the robot is often some subspace of $\mathbb{R}^d$, where d is the robot's degrees of freedom. The forward kinematics allow us to determine a robot's end-effector position based on the robot parameters including joint angles and link dimensions. This is rather easy as the transformations may be written symbolically, and simple multiplication of matrices for each link gets the desired result. Inverse kinematics is quite difficult in comparison. Because the transformation matrices vary with joint angles, computing an inverse transformation demands an inverse matrix calculation with a matrix varying over time. This is difficult to represent symbolically. When accounting for obstacles, conventional robotics considers mapping collisions into the joint space infeasible. To counteract this complexity, major methods involving robot movement use numerical methods to reduce the impact of high degrees of freedom on computation.

Motion planning involves taking a set of waypoints in the robot's joint space and commanding the robot to move to each during execution of a plan [17]. Determination of these waypoints requires only that the inverse kinematics be calculated for the desired beginning and end positions of the robot. Through forward kinematics we can know the robot's swept volume in real space. The initial and final configurations of the robot are often set by a human. The robot's decision process comes from the method it chooses to interpolate its motion. If obstacles and objects in real space are of concern and can be simulated, many optimal algorithms exist, and any-time motion planning solutions such as RRT* allow for a degree of optimality within a restricted timeframe. However, the time scale for replanning in case of failure for a 7-DOF arm can be as high as .2 seconds. This may seem like a short

time, but this is as fast as human conscious reaction time [18], and does not account for the time it takes for the robot to execute the new plan.

Data-backed machine learning is a well-studied method to decide on final configurations, especially for manipulation tasks [12], [14], [19]–[21]. Typically this involves collecting data on a simulated or real robot system to decide on what causes good and bad outcomes. In the case of supervised learning, what constitutes the value of an outcome is human-determined. Reinforcement learning leverages the long-term expected reward, or value, of actions an agent may take at a given point in time. By maximizing value, which we encode by modeling reward, a robot agent will, with practice, perform some desired task by forming a policy, which tells the agent which action to take to maximize value. This approach is particularly appealing in robotics for its ability to learn online. These methods often require large amounts of data to inform the maximization process, but using an actor-critic model, two functions (e.g., neural networks) operate in tandem. The "actor" determines what actions the robot should take and the "critic" decides how good or bad those actions are. Methods in this area have a lower data demand to reach optimal outcomes [22]. Neural networks offer the ability to learn features of the state space, reducing potential human bias towards what observable variables gain priority. Together with planning, learning sends robots closer to full automation.

A large body of work exists around grasping, but placement is a less explored area. Work focusing specifically on placement models location accurately, but small errors in locations for placement coming from learned methods create artifacts such as dropping the item in question [23]. This is undesirable for good manipulation. Humans can place objects gently, and this is often critical in areas we desire robots the most. For example, dropping glass vials in a medical lab costs a significant amount of money for materials lost and risks contamination of other materials in the lab.

In the moment-to-moment movements, robotic control is a fantastic solution to make small adjustments or account for outside stresses on a system. Control is a well-studied

engineering discipline and contributes greatly to automation. A controller uses a set of variables to determine the state of a system. Open-loop control adjusts the system based on its current state. A microwave timer is an example of open-loop control; the system turns off once enough uptime has elapsed. Feedback control explicitly incorporates outside variables and the system state in a cost function. Modeling cost gives a path to minimization through adjusting the system itself. Basic feedback control adjusts based on history, one or more steps in the past. Model predictive control adds a forecast of the system state to optimize at the current time and future times as well. Controllers can operate on the time scale of thousands of times per second, limited of course by the processing speed of the system and communication means. Most importantly, feedback control allows modeling gentle grasping and placement [11]. This in part achieves our goal of human mimicry. Humans have to detect when they make contact with a surface, whether through muscular sensation, tactile sensation, or some other means [8]. We pick feedback control as our preferred method for our desired robot system. Its ability to incorporate robot sensor data and fast computation time make it the best choice for a pick-and-place robot.

1.3 Robot Manipulation

Getting to an object involves a combination of planning and control. Fittingly, grasping involves planning and control on a smaller scale. Manipulation can be performed with or without a grasp and may incorporate learning similar to large robot movements [19], [24]. Often, manipulation tasks focus on moving location [23], but more precise movements like manipulating objects in-hand have backing research [25].

When building a robot to manipulate objects, one needs a gripper. There are several robotic grippers that incorporate tactile feedback, varying in degrees of freedom and sensory capabilities. Figure 1.2 provides gripper examples. A parallel jaw gripper like the one on the PR2 robot [5] has multiple sensors in each gripper pad, allowing localization

(a) PR2 Jaw Gripper (b) Allegro Hand (c) ReFlex TakkTile Hand

Figure 1.2: Examples of robotic manipulators.

of contact and measurement of grip strength. Its main limitation is its simplicity; a lack of a thumb or flexible digits means it can only pick up objects that it can hold with sheer force alone. Its simplicity makes it easier to model tasks, and it shows promise in general manipulation [3], [11]. A more complex gripper like the Allegro [6] has more degrees of freedom and has a wider range of possible grasps. The tradeoff with such a hand is the increased complexity of moving the hand. The motors in the Allegro give it an incredibly complex joint space. In addition, the only possible location for tactile sensors is on the fingertips, at least without severe modification.

Imitating humans yields a unique design result for robotic manipulators. The ReFlex TakkTile line of hands [7] provide tactile sensing in a gripper that can actively abduct, adduct, and flex its fingers. The tendon structures of the hand model a simplified human finger, mimicking a three-fingered hand with only two phalanges and the flexor pollicis longus on each digit. The downfall of these systems is there is no explicit representation of the pose of these underactuated joints, making simulation difficult. The underactuation serves as a benefit in the mechanics of a grasp, however. Because the motor pulls on the tendon to flex the finger, the interphalangeal joint keeps flexing after the first phalanx has hit the object. The fingers close around an object without explicit modeling of their angles. The hand fits the compliance of the object due to its imitation of human anatomy.

Advanced grippers give robots opportunity to mimic the expressiveness of human grasps. Robot grasps are considered in two forms: precision and power. Precision grasps use the fingertips, relying on finger pad pressure as the contact area. These grasps are well-suited

(a) Power grasp

(b) Precision grasp

Figure 1.3: Examples of robot grasp types [14, Fig. 6].

for dexterous tasks. Power grasps may incorporate any length of the finger and the palm. They grant stability when holding an object, so they are well-suited for high-movement tasks and heavy objects. Figure 1.3 gives examples of power and precision grasps. This classification nowhere near meets the expressiveness of human grasps, but nevertheless leads to successful grasps for robot manipulators [14]. Modeling grasp type transforms a robot's decision-making to be closer to our understanding of human grasps. Just as a human decides on how to grab a pen differently than a water glass, a robot must decide how to grab any object it targets.

1.4 Tactile Feedback in Humans and Robots

Human tactile sensation is key to manipulation. We detect slip and shock, allowing us to sense object properties and collisions in motion to even allow blind manipulation. The sensations a blind individual gains from a cane gives them a representation of objects around them. While humans have thousands of nerve endings in a given finger pad, tactile sensing on a robot is much more difficult, requiring specialized hardware and often with lower fidelity. For example, ReFlex TakkTile hands have a maximum of 14 sensors per finger, compared to the continuous coverage of nerves in human finger pads. To correct this disparity, filters on sensor readings discriminate important events [11]. Computation power

facilitates the explicit construction of a complex tactile representation where anatomy gives humans the same ability naturally.

Incorporating tactile and visual feedback in machine learning is effective on tasks similar to our project [11], [26]. Visual information allows learned estimation of geometric properties [13]. Learning and subsequent inference also facilitate dynamics estimation through tactile feedback [20], [27]. Tactile feedback has the advantage over purely visual information of approximating dynamics properties of an object. Through dynamics estimations, robots can estimate proper forces to apply to an object, similar to humans [8]. Machine learning allows robots to come closer to human levels of object knowledge from tactile information alone. Much like humans learn through experience, robots perform well on manipulation tasks given enough data.

Robot learning shows promise in extensions of grasping and placement. Through learning, robots may be able to correct mistakes or detect when the robot is not in compliance with its environment – for example, if a cup does not lie flat on some surface, but part of it still touches. Continuous feedback has been shown to be a fast and feasible approach when using only haptic data as well [28], [29]. Human tactile reflexes are the same regardless of sight, or even distraction [18]. By disentangling the visual and tactile representations of manipulation, isolated tactile information moves robots closer to human prehensile ability.

Chapter 2

Designing a Grasp Controller

We are ready to move forward in our design of a tactile-sensitive robot controller. Feedback control, or closed-loop control, is the natural choice for incorporating tactile sensory data. As mentioned in Section 1.2, feedback control uses a combination of observed variables from the system and the system state to decide on a course of action. Given a desired value of observed variables S_{des} and observed variables S_{obs}, we can operate the system by

$$u = p(S_{des}, S_{obs})$$

where p is a function dependent on system state and u is the adjustments to the system. The simplest form of this p is the calculation of an *error signal*, $S_{des} - S_{obs}$. When controlling a robot, the configuration adjustments force changes in the real-world dynamics, changing S_{obs} at the next measurement step. This process forms the titular "closed-loop" of feedback control. The goal of p is induction of a negative feedback loop to get the system closer to the desired variables. When p operates on the error signal, this means the system adjusts less if it is close to the desired variables. The system is stable when it reaches equilibrium. Once the system state changes, S_{des} changes, altering the behavior of the system.

As we are attempting to command a robot based on tactile feedback, closed-loop control provides direct incorporation of tactile information via the error signal, respecting our

inspiration of human grasps. It has a large body of work supporting its positive performance across manipulation tasks. Control also works well alongside robot learning, allowing extension of our work in the future.

2.1 Controller Goals

We wish to explicitly state our goals for the controller. Specific desired properties give each part of the controller motivation for its inclusion. Explicit goals also present criteria for comparing alternatives when comparing design options. Our goals for the controller are

1. **Stable.** Controllers are subject to chattering when controlling on a high-frequency signal, as sensor data can cause the error signal to fluctuate in such a way that the system is constantly over-compensating. To achieve stability, the controller must limit itself to small changes and tolerate a level of error or compensate for instability in the control signal.

2. **Tactile.** The sensation of human touch is key to prehensile capabilities. Taking humans as inspiration means using tactile data from robots. Moreover, we wish to solely use tactile data, as humans have the capability to manipulate without vision.

3. **Complete.** Our controller should span the entire task. Object manipulation is a nontrivial problem with many subtasks, so we should pursue a controller that adapts from the initiation of a grasp to releasing an object. This includes detecting events like the placement of an object on a surface.

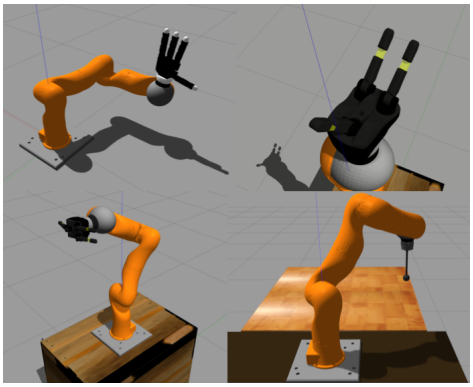
With our controller goals in mind, we can make conscious design choices that help us reach them. We will use feedback control as the backbone of our design. The exact design of the controller depends on the abilities of the system. The rest of the system depends on the hardware used and our state model for manipulation subtasks. We also require software to encode our model and communicate commands to the robot.

2.2 Robots and Software

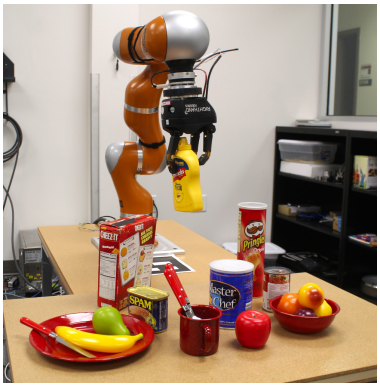
The KUKA LBR4+ arm is our choice for the base of our system. It is a 7-DOF robot arm, allowing for arbitrary position and orientation in its workspace, and its attachment head makes robotic grippers interchangeable. The ReFlex TakkTile robot manipulators are our choice for the project due to their human-inspired design and the compliant properties of the interphalangeal joints. Figure 2.1 depicts the combined setup. These hands have simple pressure sensors along each link in the fingers, allowing a degree of localization of pressure. The second version contains an internal measurement unit (IMU) in the palm and each fingertip with accelerometers and gyroscopes, allowing physical disturbance detection via linear acceleration [7]. The preshape motor that abducts and adducts the fingers will be useful in performing precision grasps as well. Lastly, each finger has a magnetic proximal encoder that measures the joint angle from 0 radians when open and at rest to π radians when closed.

The ReFlex hands work best in the real world. Simulating the 3D-printed deformable material in the hands is nontrivial, as is simulating the underactuated joints. The pressure sensors are also difficult to model as there are multiple along each finger, with limited surface area. Thus pressure values in simulation are prone to spiking without an accurate calibration of how much pressure it takes to maximize the sensor reading in the real world. Due to the COVID-19 pandemic, non-essential University of Utah labs must work remotely. As such, simulation is a more appealing option going forward, but due to the difficulties mentioned the transition is not yet complete.

To operate the robot, the creators of the ReFlex hands provide a package [30] for Robot Operating System (ROS), an open-source platform aimed towards robotics research [31]. ROS includes the `roscpp` and `rospy` APIs for C++ and Python, respectively. The just-in-time compilation and simple syntax of Python makes iterations on software development incredibly fast. These factors distinguish Python as the best choice for our controller software. We choose Ubuntu as the operating system under our controller as it is a requirement



(a) Simulated KUKA arm [32]



(b) Real-world KUKA arm and ReFlex hand

Figure 2.1: Desired robot and manipulator for tactile feedback control.

for ROS. With our software and hardware decided, we have the constraints under which we design the rest of the controller.

2.3 Grasping in Discrete Phases

Object manipulation is a nontrivial task, with many subtasks across a single instance of manipulation. Johansson and Flanagan provide a breakdown of these subtasks as phases which humans perform semi-consciously as part of manipulation [8]. Romano et al. refine these phases into a state machine for application to the PR2 robot [11]. Breaking manipulation into phases allows more precision at any given time by interpreting robot feedback differently based on the phase. More importantly, Johansson and Flanagan give a phase to placement. Using their model means we include object placement, a chief goal of our controller. A state machine model provides an outline of software design, guiding development. We outline the phases and their purposes in manipulation so we may design around human performance. Section 2.4 provides explanations of our changes from the PR2 controller to our own. The phases in order are

1. **Close.** The fingers must close around the object before any meaningful manipulation may be performed. Fast-adapting nerves detect when the fingers make contact and the next phase begins. This phase is referred to as **Reach** by Johansson and Flanagan.

2. **Load.** Fingers increase force on the object while slow-adapting nerves detect skin deformation. Application of force continues alongside lifting force until the object leaves the surface, which fast-adapting nerves detect.

3. **Lift.** The object is in motion towards some goal location. Fast-adapting nerves detect any collision with objects while in motion and slips the object may have in-grasp. Slow-adapting nerves continue to fire to provide the sensation of applying force.

4. **Hold.** The object is static and stable in-grasp.

5. **Replace.** The hand moves the object towards is goal placement, and fast-adapting nerves detect when collision is made with the desired surface.

6. **Unload.** The fingers loosen around the object, and slow-adapting nerve activity lessens as the fingers decrease normal force on the object. Eventually, the fingers break contact with the object.

Johansson and Flanagan describe 3 and 4 as separate phases, but we opt for the simplification from the PR2 controller that combines these. The robot may remain in motion and have many waypoints to which it wishes to move the object. Combining these phases turns them into one that simply needs to compensate for slips while the arm moves the manipulator and object to different locations. In addition, we add the phase from the PR2 called **Open**, which signals to the robot that it may freely return the gripper back to an open position.

2.4 Grasp Model

We must have a detailed approach to our grasp controller in order to achieve good performance. The PR2 controller from which we draw inspiration controls on E, the gripper

motor effort, by

$$E = K_P \cdot (x_g - x_{g,des}) + K_D \cdot (v_g - v_{g,des}) - \text{sign}(v_{g,des}) \cdot E_{fric}$$

where x_g and v_g are observed position and velocity variables of the gripper with desired counterparts $x_{g,des}$ and $v_{g,des}$. K_P, K_D, and E_{fric} are empirically tuned constants. We desire a simpler model for movement, ideally one that uses a simple error signal. To do this, we first define our observed variables of the system for our control law. In detailing the math of the controller, we denote the L_2 norm by $||x||_2 = |x|$ for compactness. Let

$$F = \begin{bmatrix} f_{11} & f_{12} & \dots & f_{1s} \\ f_{21} & f_{22} & \dots & f_{2s} \\ f_{31} & f_{32} & \dots & f_{3s} \end{bmatrix}$$

be the $3 \times s$ matrix of pressure values where f_{ij} is the j^{th} sensor reading on the i^{th} finger.

The first ReFlex TakkTile model has nine sensors per finger, and the second has fourteen, hence the variable number of columns in F. Figure 2.2 displays the sensor layout on each hand using RViz, a ROS visualization tool. These sensors are ordered proximal to distal, with the sensors on the side of the ReFlex TakkTile 2's fingers coming after the normal-oriented sensors. The first finger lies parallel to the second finger at rest, and the third finger is the thumb of the hand. We apply a low-pass filter to each force sensor reading over time to adjust for noise. For IMU readings, let $a = [a_x, a_y, a_z]$ be the linear acceleration in the coordinate system of the hand, where y points in the direction of the thumb and z is vertical through the center of the hand. We use a high-pass filter to get the acceleration disturbances $\tilde{a}$. The position of the fingers are each controlled by one motor. Using an error signal, channeling the error into a 3×1 matrix allows direct command of finger

adjustments. Define F_{obs} by

$$F_{obs} = \begin{bmatrix} \max(f_{11}, f_{12}, \ldots, f_{1s}) \\ \max(f_{21}, f_{22}, \ldots, f_{2s}) \\ \max(f_{31}, f_{32}, \ldots, f_{3s}) \end{bmatrix}.$$

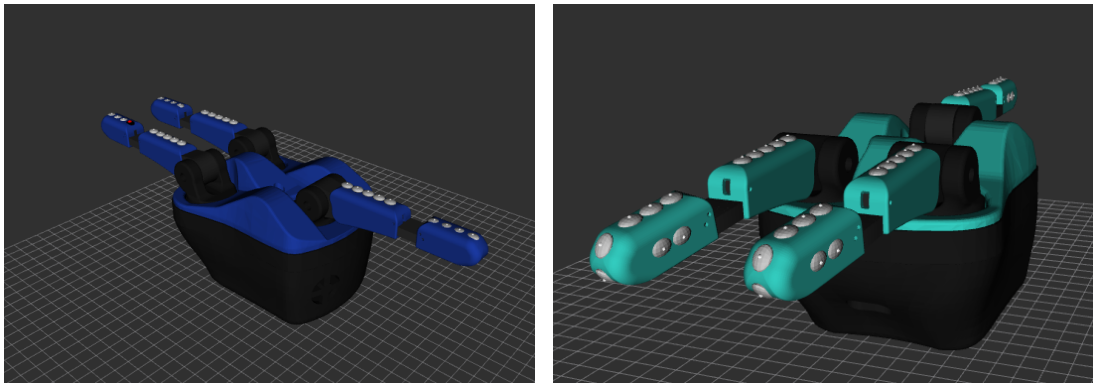
We constrain F_{des} to be a 3×1 vector in each controller phase to achieve the desired 3×1 form for the error signal. To adjust the signal's relative strength, we use the constant K_F as a form of gain to scale the error signal to the joint distance parameters. This is important as the tactile sensors range in value from 0 to 255, while the motor angle is defined in radians from 0 in rest position to π when the finger touches the palm. K_F acts as a scaling down to achieve reasonable values for motor adjustments. Additionally, to achieve stability we allow errors within some target window. We adjust the window with lower bound ΔF^- and upper bound ΔF^+. Let $F_{des,i}$ and $F_{obs,i}$ be the i^{th} components of their respective vectors. Then define

$$u = \begin{bmatrix} p_1(F_{des,1} - F_{obs,1}) \\ p_2(F_{des,2} - F_{obs,2}) \\ p_3(F_{des,3} - F_{obs,3}) \end{bmatrix}$$

with

$$p_i(F_{des,i} - F_{obs,i}) = \begin{cases} 0 & \Delta F^- < F_{des,i} - F_{obs,i} < \Delta F^+ \\ K_F \cdot (F_{des,i} - F_{obs,i}) & \text{otherwise.} \end{cases}$$

as our control function where ΔF^-, ΔF^+, and K_F are constants manually tuned for performance. In plain English, the control function subtracts the observed maximum forces from the desired and zeros them if they fall within a tolerance window. Note the maximum to gain F_{obs} makes it so only one sensor along a finger must fit in the tolerance window, effectively making the max operation act as a logical `OR` on the condition of window contain-



(a) ReFlex Takktile sensor layout (b) ReFlex TakkTile 2 sensor layout

Figure 2.2: Visualization of ReFlex TakkTile hands. Disks indicate sensor locations.

ment for all sensors along a finger. The tolerance window allows more undercompensation or overcompensation based on its defining constants. For example, having a relatively small ΔF^- compared to ΔF^+, we still operate proportional to the error when making our adjustments, but we allow for adjustments coming into the window to be above target slightly to stop chattering by repeated overcompensating adjustments around the target variable. Once the values for u are set, the system commands each finger angle change by the respective u component, positive or negative. Our phases change from Johansson and Flanagan's model and take large inspiration from the PR2 controller. Figure 2.3 provides a graphical depiction of the model.

1. **Close.** We initiate the closing of the fingers at velocity v_{close} around the object and stop each finger when they make contact. This is done by detecting when any sensor along a finger has passed some nontrivial threshold set in the code base.

2. **Load.** In this phase we set $F_{max} = \max(F_{hist})$ and $F_{des} = [F_{max}, F_{max}, F_{max}]^T$, where F_{hist} is a five-second history of finger sensor readings and max operates elementwise across the entire history. With F_{des} set we explicitly invoke our control law.

3. **Lift & Hold.** After stabilizing, we wait for a time t_{settle} before beginning control in the new phase. Rest time allows any potential deformations of the object or hand material to resolve. In this phase we use a first-order bandpass filter on F over time

and perform the same max to get $F_{obs,BP}$. We also track the high-frequency variations by running F under a high-pass filter to get $\tilde{F}$. We classify slip as when the magnitude of this vector surpasses a threshold $|F_{obs}|\varepsilon_{slip}$, and the mid-level disturbances are below the threshold ε_{BP}. This discrimination in a mid-range passband is to eliminate false positives from increasing grip force. Making the high-frequency disturbances proportional to $|F_{obs}|$ is an encoding of Weber's Law: a more intense sensation requires a higher change for the disturbance to be noticed. When we detect slip, we compensate by multiplying F_{des} by an increasing factor K_{slip}. As noted in the PR2 controller, a bandpass filter combined with our highpass filtered disturbances is effective for detecting signals similar to the fast-adapting nerves that detect sheer forces across a finger pad.

4. **Replace.** Through a ROS command, the user sends the `replace` command to the robot, initiating this phase. A placement is considered when the controller detects a slip or when $|\tilde{a}| > \varepsilon_{\tilde{a}}$, where $\varepsilon_{\tilde{a}}$ is another tuned constant. This rapid acceleration change signifies the robot's ability to sense when it has stopped.

5. **Unload.** We set our desired force at time t to $F_{des,t}$ set by

$$F_{des,t} = \frac{(t_{unload} - t)}{t_{unload}} \cdot F_{des}$$

   where $t \in [0, t_{unload}]$ is the system time since the beginning of the phase.

6. **Open.** The finger motors open with angular velocity v_{open} until they reach their rest positions.

Before the COVID-19 pandemic, we found working values that allowed good performance for our controller. We based performance on the consistency of phase switches and the stability of the fingers around an object. Table 2.1 lists the best found values of these constants. The discrete-time filters have varying effects based on their cutoff rates, which

we explore in Section 3.2.

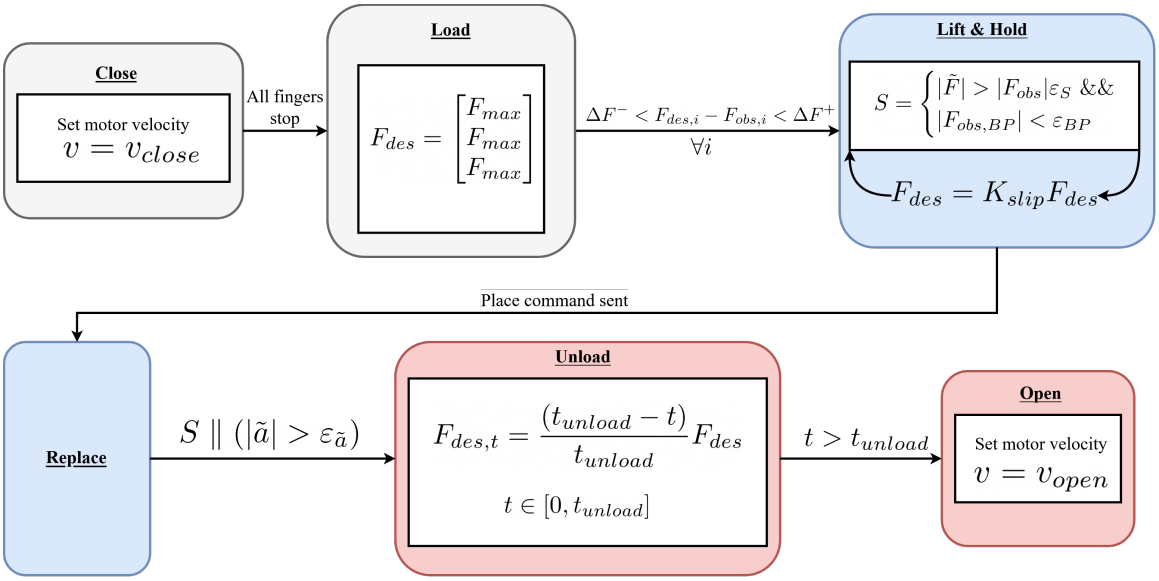


Figure 2.3: State machine of grasp model.

Constant	Value	Units
ΔF^-	-5	
ΔF^+	40	
K_F	0.0036	radians
K_{slip}	1.08	
ε_{slip}	1	
ε_{BP}	0.25	
$\varepsilon_{\tilde{a}}$	0.008	
t_{settle}	0.1	s
t_{unload}	0.2	s
v_{close}	1.2	radians$/s$
v_{open}	-1.5	radians$/s$

Table 2.1: Values of controller constants. A blank last column signifies the constant is unitless.

Chapter 3

Grasp Controller Implementation and Tuning

With our selection of a grasp model, robots, and software, we proceed to integrate the components of our design into a whole system. Synthesizing the system for our grasp controller requires software engineering skills to communicate with the embedded systems of the robots. Translating our model into code requires adapting it to the specific structures and models present in our software. A complete system requires we negotiate interactions between all of the parts of our design. Section 3.1 details the challenges in bringing our design together. Section 3.2 outlines experiments to test the impact of signal filters on the isolation of events and smoothing of sensor data, and Section 3.3 qualitatively discusses our experimental results.

3.1 Implementation Considerations

The software implementation alone presented many challenges. ROS uses an asynchronous event model to send message information. We build event handlers with thread-safe structures to allow safe modification and reading of the data from the controller software. This event model works to our advantage: after measuring the maximum rate of 20 Hz for in-

coming messages, we set our control loop rate to 20 Hz likewise, as our linear operations and filtering complete well within that time scale. The control loop operates as a recurring event: every 0.05 seconds, an event fires that runs one iteration of the control loop. During this iteration, the controller pulls sensor data, filtered where appropriate, from an intermediary controlling thread safety, reading of hand messages, and data filtering. The thread running the control iteration then interprets the data as demanded by the system state and sends the command to the hand. The code is available online and builds on the code from RightHand Robotics [33].

Research began on the ReFlex TakkTile hand. Due to a hardware failure, the sensors along the second finger would report maximum values even when nothing touched it. This failure, along with the additional features of the ReFlex TakkTile 2, motivated the move to the new hand model. The ReFlex TakkTile 2 by default does not send linear acceleration data in communication packets, instead opting to send quaternion measurements from the IMU. The firmware picked up this data by starting at the register address of the IMU's quaternion data. By changing this start address, we replaced the quaternion vector with linear acceleration data in the communication packets sent to ROS. The packet included enough space for four values of a quaternion (w,x,y,z). Altering the address turned the quaternion vector into (a_x,a_y,a_z,b) where we disregarded b entirely. From this we obtained a well-formed acceleration vector.

At the Learning Lab for Machine Automation (LL4MA) where this project took place, we had two of the newer TakkTile hand model. This redundancy proved invaluable when two failures occurred. One hand, Hand A, had a communications port failure and stopped sending data to the controller. The other hand, Hand B, had a finger lose the ability to gain power from the hand's control board, meaning the tactile sensors and proximal encoder on that finger sent back bad data. The failure caused the hand to think it was detecting less force and the joint angle was less than in reality. With two broken hands we made a new one by transplanting a working finger from the noncommunicative hand to the hand

with an insensitive finger. By the time we had working hardware the date was March 12, 2020. That same day, the University of Utah announced the move to online classes for the remainder of the semester [34]. In the following two weeks we managed to run the controller on the ReFlex TakkTile 2 a handful of times to verify our software. Like many engineering labs, the LL4MA moved to remote work for the safety of its students. For the time being, we put efforts with real robots on hold. We take this opportunity to look at the data we have and propose next steps for this project.

3.2 Sensor and Event Detection Experiments

Given the series of events surrounding this thesis, it is hard to see what can be gained from this project that is reliant on real-world systems and data. Fortunately, ROS allows recording of messages sent across its communications model, and with these recordings replay messages from robots from sessions past. We possess two such recordings where we recall the events that took place with the hand. With these, we observe the effects of filtering in two aspects: noise and placement detection via acceleration.

Experiment A: Eliminating Tactile Sensor Noise

Our first recording is of the ReFlex TakkTile hand repeatedly grasping a water bottle using explicit commands to close and open the hand. We used the default tactile stop thresholds for these commands. From this recorded run of the hand we extract the sensory data from the first finger and run it through a low-pass filter with the goal of eliminating sensor noise.

Our filter design is a first-order discrete-time Butterworth low-pass filter, where we vary the cutoff frequencies across trials. This filter allows for dropoff beyond the cutoff frequency, making exact frequency selection less crucial. More importantly, it has no ripple in the passband, leaving fewer filtering artifacts. A five-second history at a sampling rate of 20 Hz gives us a Nyquist frequency of 50 Hz, leaving plenty of room to find a suitable filter

cutoff. We run the filter with 1, 5, and 10 Hz cutoffs, as contacts worth noting are made for long periods of time. We compare filter effects on the signal noise over twenty seconds of runtime. During this runtime there are two grasp cycles of gripping and releasing the water bottle. Figure 3.1 shows the filter's effects on each of the nine sensors along the finger over time.

Experiment B: Isolating Quick Acceleration Changes

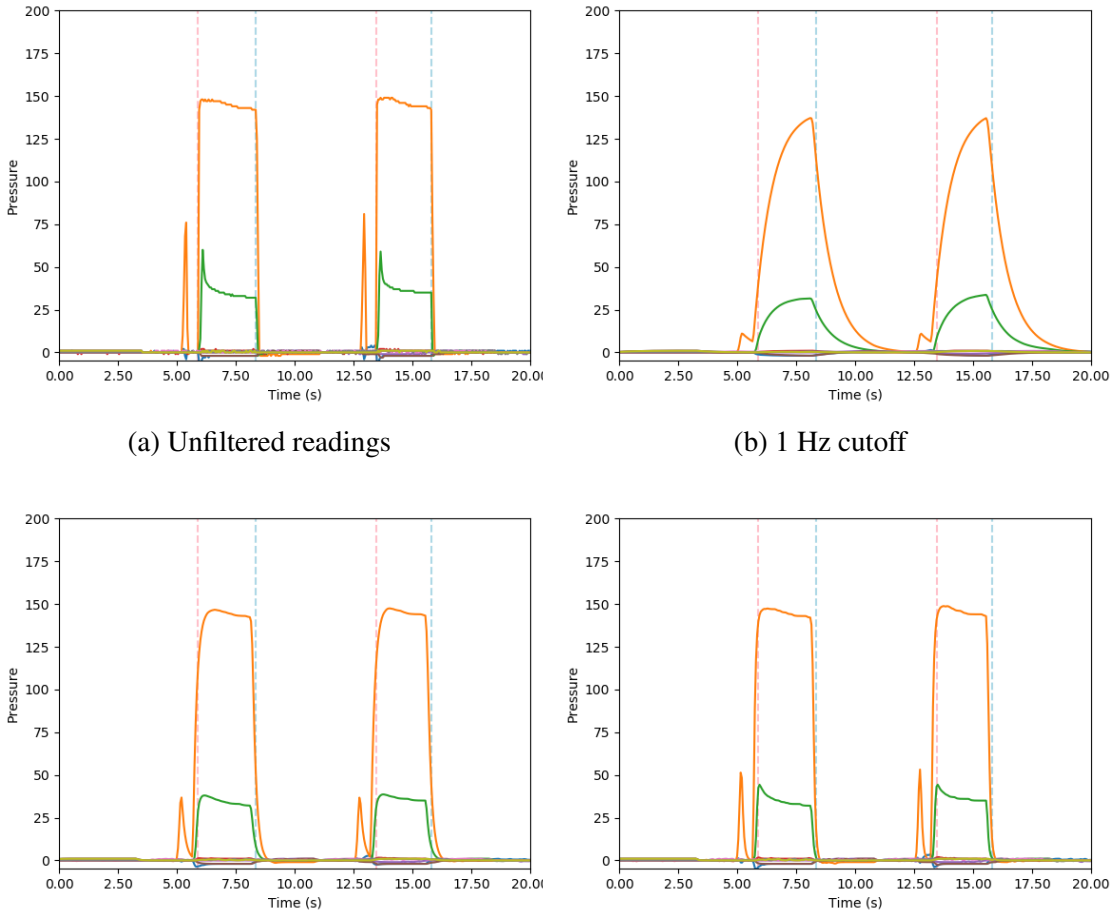
The second recording gained before the campus shutdown is a run of the ReFlex TakkTile 2 testing the IMU's accelerometer. The goal of this run was to examine if linear acceleration readings were coming through after our firmware change. We moved the hand on and off a table for a brief period of time. We isolate three significant changes in acceleration over seven seconds of the twenty-five second recording. Our chosen filter design is a first-order discrete-time Butterworth high-pass filter. In this experiment we choose cutoffs of 5 Hz, 10 Hz, 15 Hz, and 25 Hz and compare the magnitude of the filter response $\tilde{a}$ against the magnitude of the base signal a over time. Figure 3.2 depicts the results, with dotted gray lines marking significant events in the time period.

3.3 Results

Experiment A

The base readings give clear start and end points for the two grasp cycles. Two sensors responded chiefly to each grasp, signified by the orange and green lines. The chief response has an initial spike just before sustained contact, and the secondary responder spikes with the contact event to then settle as the hand maintains its grip.

Each cutoff frequency from our chosen set has a noticeable impact on the initial readings. The 1 Hz cutoff frequency is extreme and severely slows the ramp time of the contact



(a) Unfiltered readings

(b) 1 Hz cutoff

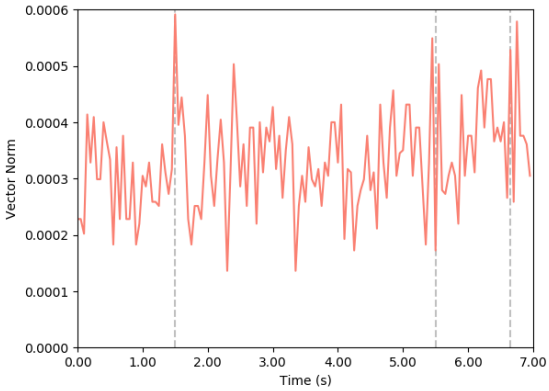
(c) 5 Hz cutoff

(d) 10 Hz cutoff

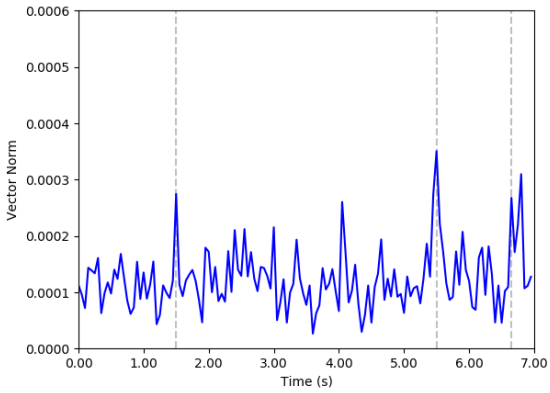
Figure 3.1: Comparisons between low-pass filter thresholds for reducing noise in pressure sensor readings. The pink dotted lines mark the beginning of contact, and the blue ones mark the release of the grip.

events. Setting the cutoff to 5 Hz makes for smooth readings while keeping the identity of the contact events intact. A sensor that spiked with the contact event lost the spike entirely under this filter, instead having a measured response to the grip initiation. The 10 Hz filter decreases the small spike before the main contact event, but still contains small variations compared to the other frequency cutoffs. Some of the spike from the less responsive contacting sensor still remains under this modality. For its noticeable but light-handed smoothing, 5 Hz appears to be the best of our choices for noise filtering.

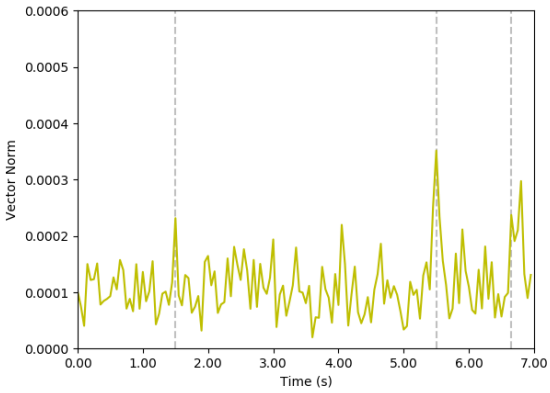
Experiment B



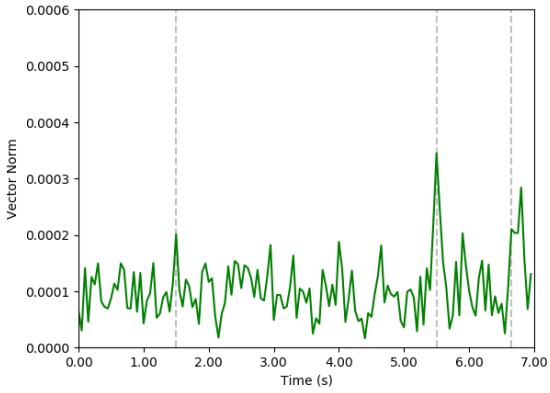
(a) Unfiltered acceleration



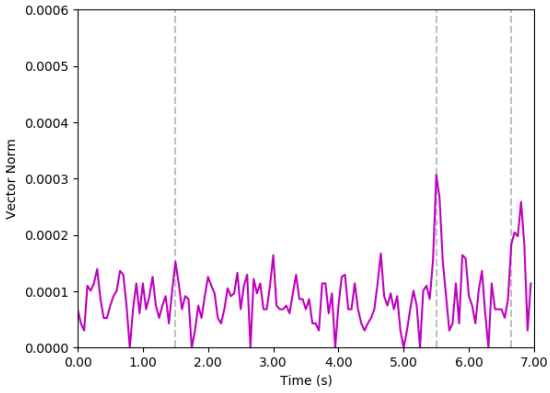
(b) 5 Hz cutoff



(c) 10 Hz cutoff



(d) 15 Hz cutoff



(e) 25 Hz cutoff

Figure 3.2: Comparisons between high-pass filter thresholds for isolating rapid changes in acceleration. The gray vertical lines reference significant changes in unfiltered acceleration readings across each trial.

The magnitude of the acceleration vector from the IMU varies rapidly due to its small scale. Any inspection at such a small scale is impacted by noise, but there are three distinct events we detect in the time frame chosen. The first is a large spike in the magnitude, potentially signifying a sudden stop, possibly against a rigid object. The two other events see rapid oscillations in the magnitude, even more so than the normal sensor noise. In each case, peaks bookend a sharp decline in acceleration magnitude. These rapid changes show a potential rebound against an object, although small. These final two events are particularly responsive under our high-pass filter design as a result of their increased oscillation.

The 5 Hz cutoff shows some sensitivity to the highlighted events, but the readings outside those events contain a high level of variation as well. The step to 10 Hz shows a more isolated response from the final two events, but less from the first. Third, the 15 Hz cutoff curbs many of the sharp but insignificant peaks present in the filters with lower cutoffs while eliciting fantastic response from the final two events. The first event is far less noticeable than in lower-cutoff filters, however it still has a degree of response. At the highest cutoff lies the 25 Hz filter. This filter does not have perceivable response gains over the 15 Hz filter in terms of the later two events and sees no distinguishable response from the first event. This filter approaches diminishing returns for filter cutoffs on this system. This makes sense, given the underlying sampling rate of 20 Hz from the robot.

The combined responses of the 10 Hz and 15 Hz filter show promise for different instances of fluctuating acceleration. Less extreme cutoffs isolate the single spike from our data decently, but allow a noticeable amount of oscillation from the raw measurements. Sensor noise is by definition high-frequency, so the best a high-pass filter does is reduce the noise's amplitude to focus on significant events. With a balance of noise cancellation and event preservation, the band of 10 to 15 Hz for a filter cutoff shows promise in isolating rapid changes in acceleration from the sensor data.

Chapter 4

Discussion

A tactile grasp controller is no simple piece of software. It requires knowledge from many engineering disciplines: mechanical, electrical, and computer science. The level of background required to approach this task is significant. Robotics is a specialized interdisciplinary field, and current research reflects that. Beyond the background knowledge of robotics alone is the human motivation. We compare machine learning classification systems with humans as the gold standard. Robotics holds many of the same principles as other machine learning specialties in this respect. Taking inspiration from humans, we adapted a grasp controller using only tactile information. This is a major step towards blind manipulation in robotics. Despite a global pandemic freezing lab work, we investigated what little data we had and focused the search on two key detection systems in the controller.

In future projects, more data will facilitate deeper analysis of the capabilities of a tactile controller. In the case of ROS, labelling robot session recordings with real-world events will be invaluable in parsing the data at a later time. A more refined search for filtering methods will benefit the controller in the long-term, as a more precise filter cutoff molds the filters to our desired effect. Hardware can be unpredictable, and the best salve for broken hardware is time, whether working on the system or ordering a new part. We will account for failure recovery time in the future for hardware and software and achieve a

fail-safe timeline for the project.

For the foreseeable future, we must be able to work from home. Simulation offers an opportunity to continue our work remotely, and the modeling difficulties of our robot expressed in Section 1.3 make this a challenge worthy of any quarantine duration. Upon verifying a functional controller, we can extend it to many other projects. Before even that, constants within the controller need tuning and refinement. Learning approaches can help achieve this through trials of different control values and searching for the optimal values through inference or reinforcement learning.

The Learning Lab for Machine Automation sees this project in the context of its larger grasping pipeline. The work of Lu, Van der Merwe, Sundaralingam and Hermans [14] complements our work quite well. We can use learning for the approach to the object and the controller to successfully grasp it from an optimal preshape. There are more ways we may incorporate learning. The example presented in the introduction of a robot waiter involves a situation where a robot could classify, based on tactile information, whether a grasp was going to fail. From there it could use a reinforcement learning-based policy to readjust the object so it sets properly. This grasp recovery also applies to fitting the compliance of a surface. In the case a robot needs to stably place an object on a slanted surface or on unstable material, the same work could apply to readjust the object to a stable equilibrium. There are many places for our controller in robotics. We look forward to seeing how we can push the field forward with tactile manipulation.

Bibliography

[1] S. H. Ivanov and C. Webster, "Adoption of robots, artificial intelligence and service automation by travel, tourism and hospitality companies – a cost-benefit analysis," in *Contemporary Tourism – Traditions and Innovations*, Sofia University, Oct. 2017, pp. 19–21.

[2] Intuitive Surgical, *About da Vinci Systems*, https://www.davincisurgery.com/da-vinci-systems/about-da-vinci-systems, Accessed: May 1, 2020, 2020.

[3] D. Park, Y. Hoshi, H. Mahajan, W. Rogers, and C. Kemp, "Toward active robot-assisted feeding with a general-purpose mobile manipulator: Design, evaluation, and lessons learned," Apr. 2019.

[4] DESTACO, *Automatic & Manual Robot Tool Changers*, https://www.destaco.com/tool-changers-effectors.html, Accessed: May 1, 2020, 2020.

[5] Willow Garage, *PR2 overview*, http://www.willowgarage.com/pages/pr2/overview, Accessed: May 1, 2020, 2015.

[6] Wonik Robotics, *Allegro Hand*, http://wiki.wonikrobotics.com/AllegroHandWiki/index.php/Allegro_Hand, Accessed: May 1, 2020, 2020.

[7] Righthand Labs, *ReFlex Robotic Gripper*, https://www.labs.righthandrobotics.com/reflexhand, Accessed: April 30, 2020, 2020.

[8] R. Johansson and J. Flanagan, "Coding and use of tactile signals from the fingertips in object manipulation tasks," *Nature Reviews Neuroscience*, vol. 10, pp. 345–359, May 2009. DOI: 10.1038/nrn2621.

[9] C. L. Taylor and R. J. Schwarz, "The anatomy and mechanics of the human hand," *Artificial Limbs: A Review of Current Developments*, vol. 2, no. 2, pp. 22–35, 1955, Accessed: May 1, 2020.

[10] M. M. TRAUB, J. C. ROTHWELL, and C. D. MARSDEN, "A GRAB REFLEX IN THE HUMAN HAND," *Brain*, vol. 103, no. 4, pp. 869–884, Dec. 1980, [Abstract]. Accessed: May 5, 2020, ISSN: 0006-8950. DOI: 10.1093/brain/103.4.869. eprint: https://academic.oup.com/brain/article-pdf/103/4/869/1030788/103-4-869.pdf. [Online]. Available: https://doi.org/10.1093/brain/103.4.869.

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

[12] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, Apr. 2016. arXiv: 1504.00702.

[13] M. V. der Merwe, Q. Lu, B. Sundaralingam, M. Matak, and T. Hermans, *Learning continuous 3d reconstructions for geometrically aware grasping*, 2019. arXiv: 1910.00983 [cs.RO].

[14] Q. Lu, M. V. der Merwe, B. Sundaralingam, and T. Hermans, *Multi-fingered grasp planning via inference in deep neural networks*, 2020. arXiv: 2001.09242 [cs.RO].

[15] M. R. Cutkosky *et al.*, "On grasp choice, grasp models, and the design of hands for manufacturing tasks.," *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 269–279, 1989.

[16] R. L. Williams II, *Baxter humanoid robot kinematics*, https://www.ohio.edu/mechanical-faculty/williams/html/pdf/BaxterKinematics.pdf, Internet Publication, Accessed: May 2, 2020, Apr. 2017.

[17] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006, Accessed: May 2, 2020.

[18] J. V. M. Hanson, D. Whitaker, and J. Heron, "Preferential processing of tactile events under conditions of divided attention," *Neuroreport*, vol. 20, no. 15, pp. 1392–1396, 2009, Accessed: May 1, 2020. DOI: 10.1097/WNR.0b013e3283319e25.

[19] O. Kroemer, S. Niekum, and G. D. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *CoRR*, vol. abs/1907.03146, 2019. arXiv: 1907.03146. [Online]. Available: http://arxiv.org/abs/1907.03146.

[20] O. Kroemer and J. Peters, "A comparison of autoregressive hidden markov models for multimodal manipulations with variable masses," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1101–1108, 2017.

[21] S. Leischnig, S. Luettgen, O. Kroemer, and J. Peters, "A comparison of contact distribution representations for learning to predict object interactions," in *Proceedings of International Conference on Humanoid Robots (Humanoids)*, Jan. 2015.

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018. arXiv: 1801.01290. [Online]. Available: http://arxiv.org/abs/1801.01290.

[23] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to place new objects in a scene," *The International Journal of Robotics Research (IJRR)*, 2012.

[24] M. Wilson and T. Hermans, *Learning to manipulate object collections using grounded state representations*, 2019. arXiv: 1909.07876 [cs.RO].

[25] S. Cruciani, B. Sundaralingam, K. Hang, V. Kumar, T. Hermans, and D. Kragic, "Benchmarking in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 588–595, Apr. 2020, ISSN: 2377-3774. DOI: 10.1109/lra.2020.2964160. [Online]. Available: http://dx.doi.org/10.1109/LRA.2020.2964160.

[26] S. Hoppe, Z. Lou, D. Hennes, and M. Toussaint, "Deep learning for manipulation with visual and haptic feedback," 2017. [Online]. Available: https://ipvs.informatik.uni-stuttgart.de/mlr/papers/17-hoppe-IROSws.pdf.

[27] B. Sundaralingam and T. Hermans, *In-hand object-dynamics inference using tactile fingertips*, 2020. arXiv: 2003.13165 [cs.RO].

[28] D. Park, A. Kapusta, J. Hawke, and C. C. Kemp, "Interleaving planning and control for efficient haptically-guided reaching in unknown environments," in *2014 IEEE-RAS International Conference on Humanoid Robots*, IEEE, Nov. 2014. DOI: 10.1109/humanoids.2014.7041456. [Online]. Available: https://doi.org/10.1109%5C%2Fhumanoids.2014.7041456.

[29] T. Bhattacharjee, A. Kapusta, J. Rehg, and C. Kemp, "Rapid categorization of object properties from incidental contact with a tactile sensing robot arm," *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015, pp. 219–226, Feb. 2015. DOI: 10.1109/HUMANOIDS.2013.7029979.

[30] RightHand Robotics, *Reflex-ros-pkg*, https://github.com/RightHandRobotics/reflex-ros-pkg, Accessed: May 5, 2020.

[31] Open Source Robotics Foundation, *Documentation - ROS Wiki*, https://wiki.ros.org, Accessed: April 30, 2020, 2018.

[32] A. Conkey, *Robot control in gazebo*, https://adamconkey.github.io/project/robot_control/, Accessed: May 4, 2020.

[33] A. Bull, *Reflex-ros-pkg at ll4ma-tactile-feedback*, https://github.com/BullAJ/reflex-ros-pkg, Accessed: May 5, 2020.

[34] University of Utah Communications, *Measures being taken to slow the spread of covid-19 to the university community*, https://attheu.utah.edu/facultystaff/measures-being-taken-to-slow-the-spread-of-covid-19-to-the-university-community/, Accessed: May 3, 2020, Mar. 2020.