An Exploration of Low-Cost Tactile Sensing in Robotic Manipulation

Jorge Cañardo Alastuey
University of Colorado at Boulder, jorgecanardo@gmail.com

Follow this and additional works at: https://scholar.colorado.edu/ecen_gradetds
Part of the Electrical and Computer Engineering Commons, and the Robotics Commons

Recommended Citation
https://scholar.colorado.edu/ecen_gradetds/156

This Thesis is brought to you for free and open access by Electrical, Computer & Energy Engineering at CU Scholar. It has been accepted for inclusion in Electrical, Computer & Energy Engineering Graduate Theses & Dissertations by an authorized administrator of CU Scholar. For more information, please contact cuscholaradmin@colorado.edu.
An exploration of low-cost tactile sensing in robotic manipulation

by

Jorge Cañardo Alastuey

M.S. Industrial Engineering, Polytechnic University of Catalonia, 2014
M.S. Mathematics, Polytechnic University of Catalonia, 2013

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Master of Science
Department of Electrical Engineering

2016
This thesis entitled:
An exploration of low-cost tactile sensing in robotic manipulation
written by Jorge Cañardo Alastuey
has been approved for the Department of Electrical Engineering

Prof. Nikolaus Correll

Prof. Christoffer Heckman

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.
Haptic sensing remains an open research area. Tactile feedback is acknowledged to be fundamental for human grasping, and yet robotics hardly uses it in practice.

In this work, we explore the manipulation benefits of a low-cost tactile sensor that combines proximity and force measurements. We implement a pick-and-place pipeline, and show that success rates increase in different tasks when adding haptic feedback. We study a modified version of the tactile sensor, which drops proximity sensing for a more accurate and wider range force measurement.
I want to thank my advisor, Dr. Nikolaus Correll, and the members of the Correll Lab for making it such an enjoyable work environment, full of knowledge and fun times. In particular, I’d like to acknowledge Radhen Patel and the whole Picknik team for the related work we’ve done together.

I must thank Mr. Pete Balsells, Dean Rob Davis and his team, and the Generalitat de Catalunya for the Balsells fellowship that has made this work possible. Previous funding from the Interdisciplinary Higher Education Center (Centre de Formació Interdisciplinària Superior, CFIS) was key to arrive here today.

My mom, dad and sisters have played a fundamental role, and their persistent support has led me here. Thanks to my family and friends, and thank you, dear reader.
Contents

Chapter

1 Introduction .................................. 1
   1.1 Motivation: Amazon Picking Challenge .............................. 1
       1.1.1 Insights ........................................... 3
   1.2 Related work ............................................. 4
   1.3 Outline and contributions of the thesis ............................ 6

2 Materials and methods ......................... 8
   2.1 Hardware .................................................. 8
       2.1.1 Tactile sensor ........................................ 9
   2.2 Software ................................................... 10
       2.2.1 Tactile information ................................... 11

3 Pick-and-place pipeline ......................... 13
   3.1 Grasping state machine ..................................... 13
   3.2 Calibration and perception .................................. 13
   3.3 Pick and place ............................................. 16

4 Experiments .................................... 18
   4.1 Results .................................................... 18
       4.1.1 Tower building task .............................. 19
Table 1.1 Classification of the four types of mechanoreceptors on the skin. They are classified according to their response speed and area that they cover. Nerve endings with small (resp. large) receptive fields are referred to as “Type I” (resp. “Type II”). If they respond to steady stimuli the nerve endings are “slowly adapting” (SA), or “fast adapting” (FA) if respond to dynamic stimuli.
Figures

Figure

1.1 On the left, part of the Picknik team (from left to right: Dave Coleman, Andy McEvoy, Steve McGuire), during a demo at the Boulder Is For Robots meetup, held at CU Boulder on April 15th 2015. On the right, preparing to compete at the APC during ICRA 2015, in Seattle. ......................................................... 2

1.2 Left: One of the many teams that used suction in the APC. Right: Sensor vendors at ICRA are aware of the possibilities of tactile sensing for robotics, but in practice few teams benefit from them. ......................................................... 3

1.3 Tactile receptors in the skin, from [49]. The SA-I mechanoreceptors correspond to Merkel cells, SA-II to the Ruffini endings, FA-I to the Meissner corpuscles, and FA-II to the Pacinian corpuscles. ......................................................... 5

2.1 A view of Baxter’s electric parallel gripper, schematically and in real life. There are 8 sensors embedded in PDMS in each finger. The IR receiver-emitter (VCNL 4010) are the black integrated circuits over the white board. ................................. 9

2.2 A view of Jaco, with an AR tag on its wrist. Jaco is a 6 DoF robot that weights 5.2 kg, and its maximum payload is 1.6 kg. It is built using carbon fiber, and the motros drive harmonic gears. ......................................................... 10

2.3 An illustration of the way the sensor works (from [38]). The infrared light emitted is reflected both at the PDMS-air interface, and by nearby objects. When an object presses against the PDMS, its width changes. ......................................................... 11
2.4 A high level overview of the communications in the pipeline. Most of the code and functionality lives in the ROS layer. The sensor readings, which are published via I²C and relayed via serial by an Arduino board, also end up being converted into ROS messages, so that they can be consumed by different nodes. The ROS layer can be seen with more details in figure 3.5.

3.1 Different phases of grasping, with their associated events (from [39]). Clockwise, from the top left: (1) Approach, (2) Alignment, (3) Contact, (4) Lift, (5) Shear/slip, (6) Disturbance, (7) Placement, and (8) Release.

3.2 State machine.

3.3 Calibration process, from left to right. Initially, the 3D point cloud given by the camera is given with respect to the camera frame, but the pose of that frame is unknown in the robot’s own base reference frame. Next, we acquire the AR tag’s pose both from the robot’s base (through the joint encoders), and from the camera’s reference frame. This enables calibration, but typically we move the arm to a new pose, and compute the transform from the robot’s base to the camera that minimizes the squared error over several calibration poses.

3.4 On the left, view from RViz, where the calibrated point cloud is superimposed to a model of the robot. Note the coordinate frames on each cubelet, computed by the perception node; as well as the AR tag, that enables eye-to-hand calibration. On the right, the view seen through the ASUS Xtion Pro at the same timestamp.
3.5 ROS nodes used to run the experiments. They are independent from another, so provided that they maintain the same interface (i.e., use the same message types and topic names), each component can be modified, or even entirely replaced without affecting the rest of the pipeline. This should make it easy for different people to work on the parts that interest them most without having to understand the full setup. The “finger sensor” node reads data from serial, and publishes it. The “artificial mechanoreceptors” node computes the FA and SA signals, which we visualize in real time use rqt_plot (not shown in the diagram). The manager nodes defines the experiment to be performed, encoded as a finite state machine. It also initializes the robot used, and employs tactile signals to drive some state transitions. Calibration and perception have been the most stable nodes throughout the development, as they are not part of our current research.

4.1 Actual pressure values when holding a 2-inch cube. The rightmost column shows the sensor values at the tip.

4.2 A successful attempt at building a tower of cubelets. From left to right, the initial disposition of the cubes, after stacking a cubelet, and after making it to three levels high. Reactive control was employed, and it was necessary to make it successful.

4.3 From [39]. Schematic drawing of the tower construction task on the left. If the gripper is not centered over the tower, the block might topple after placement. If the pre-grasp position is too low, the tower might fall. Finally, if the pre-grasp position is too high, the added block might fall. Right: Success rate to build towers of height two and three with and without tactile sensing.

4.4 From [38]. Left: Experimental setup. Middle: Schematic drawing of the experimental setup. Closing the gripper might topple the top building block unless the gripper is centered, here by moving left. Right: Success rates for grasp trials with increasing noise in the pre grasp position (drawn from a uniform random variable).
4.5 Signals during a contact event (from [39]), where the gripper closes against a cubelet. 23

4.6 FA-I signals during a placement event (from [39]), where a block held by the gripper touches the table. On the left, a single event. On the right, the overlap of seven such events, aligned using their cross-correlation. 23

5.1 Some of the finger boards, before the components are soldered to them. 25

5.2 Infrared light can travel through dust or smoke, and even some materials like the plastic bag that is clearly visible on the right side. Public domain pictures from NASA/IPAC (http://coolcosmos.ipac.caltech.edu/cosmic_kids/learn_ir/index.html). 26

5.3 On the left side, the 3D printed finger can be seen. It is connected to a multiplexing board, and it talks to the Arduino board via I²C. On the right, the force gauge used. 27

5.4 Force applied with the digital force gauge vs the raw sensor readings. Each color represents a different trial. To record each point, we maintained a constant force for a few seconds (≈ 100 samples), and the (small) error bars in x show their standard deviation. For the force, the error bars represent ±1 N. 29
Chapter 1

Introduction

The word *robot* [30] has been adopted by English and many other languages, unaltered from the Czech *robot*. Itself, it was coined by Josef Čapek from *robota*, meaning servitude, and it was first used by his brother Karel in 1921, for his science-fiction play “R.U.R. (Rossum’s Universal Robots)”.

Robotics, the study of robots, is a term coined by Isaac Asimov, and the field of robotics has progressed enormously from its science fiction days. However, humans can still easily outperform robots when dealing with unstructured environments and complex manipulation tasks.

Here, we focus on robotic grasping, and how tactile feedback can improve reliability in different tasks.

1.1 Motivation: Amazon Picking Challenge

Amazon announced its first Picking Challenge (APC) around the time of my arrival to the Correll lab. The APC is a competition to pick up items from warehouse shelves. The 2015 APC was to be held in Seattle, Washington, during the 2015 IEEE International Conference on Robotics and Automation (ICRA).

A few members from the Correll and ARPG labs took part in the competition. In terms of hardware, we used a Jaco arm mounted on a vertical gantry with its default gripper, a three-fingered hand (figure 1.1, right). In the software side, our stack was based on the Robot Operating System (ROS), and heavily used functionality from MoveIt!.
Over five months, our team (figure 1.1, left) built a pipeline that could recognize objects in the provided shelves, and plan collision-free trajectories to them. We developed a grasp generator that created grasp approaches, which were filtered according to their feasibility. We found that calibrating different cameras to our robotic arm was hard and not accurate enough to directly execute trajectories computed in a planning scene with virtual collision models. Even when calibration was good, the planned trajectories in the simulated collision world would sometimes fail due to the large number of uncertain measurements that were taking place. This turns out to be a well known problem [14].

As typical examples of the failure modes, the planner would generate trajectories that were theoretically valid (collision-free), but they would fail when executed. Sometimes, a grasped object would stick out more from the gripper than expected or detected by our vision system, maybe because it was transparent, and would thus hit the shelve and then fall on the ground. When grasping an object, the arm would reach for its centroid as computed by the perception pipeline. However, it wouldn’t perfectly align with reality, and a finger would push the object instead of enveloping it, so grasping would fail.
During testing, we had a fairly decent performance. Unfortunately, during the actual competition a calibration error during our second attempt caused the gripper to grab one side of the shelf instead of the object nearby. When the arm tried to pull, a low level hardware protection was activated to avoid mechanical damage. Software could not recover from that failure mode, and our final attempt was over with no successful grasps.

1.1.1 Insights

The work done to participate in the APC highlighted some of the problems that arise in robotics when working in unstructured environments, and the complexity of getting a robust and fully functional pipeline to test new and different grasping approaches.

It also motivates the need to add reactive control to grasping tasks. Most teams in the APC relied exclusively on vision to detect, classify and grasp the requested items. This was possible because most objects in the APC could actually be sucked, with no need for precise manipulation or grasping (figure 1.2, left).

Figure 1.2: Left: One of the many teams that used suction in the APC. Right: Sensor vendors at ICRA are aware of the possibilities of tactile sensing for robotics, but in practice few teams benefit from them.

However, vision is brittle. Some teams showcased this problem, since they had to incorporate
powerful lighting systems or even shades to their setups, in order to achieve consistent lighting that wouldn’t confuse their perception systems. We planned on adding a small vibrating motor to one of our RGB-D cameras to remove unwanted interference. Vision is also difficult to to use when working in small spaces, where the gripper itself may be blocking the camera view.

On the other hand, humans are surprisingly good at grasping and manipulating objects, and we are able to do so even if we close our eyes. Unlike robots, who have trouble using the same method to grasp an egg without crushing it and picking up a dense metal cube, humans can detect slippage and quickly increase the holding force. In the end, the tactile feedback from our hands allows us to essentially use the minimum force that is required to hold any object [28]. One single APC team employed tactile feedback during the competition [10], but some expensive tactile sensors are commercially available (figure 1.2, right).

Finally, a summary of lessons learned by all the teams [10] pointed out that “system integration and development remain fundamental challenges in robotics”. In this thesis, we deal with the system challenges and integration of tactile feedback for grasping.

1.2 Related work

Human’s ability to manipulate and grasp objects depends greatly on haptic cues, and less so on vision [28]. A diminished tactile sensitivity, such as in users of prosthetic hands, makes manipulation hard because of the lack of feedback about contact between the hand and the object [11]. In [28], the process of picking up an object is separated into various phases, namely reach, load, lift, hold, replace, unload, and release.

Specific sensory events are generated by mechanoreceptors (i.e., transducers between physical movement on the skin to neural signals) in between the previous phases. There are four types of mechanoreceptors in the human hand, which are classified depending on their response speed and the area that they cover (table 1.1 and figure 1.3). Each type of afferent nerve is stimulated by different transitions [42], and [11] argues that similar information is necessary for robots to manipulate objects, the same way our brains require it too.
Table 1.1: Classification of the four types of mechanoreceptors on the skin. They are classified according to their response speed and area that they cover. Nerve endings with small (resp. large) receptive fields are referred to as “Type I” (resp. “Type II”). If they respond to steady stimuli the nerve endings are “slowly adapting” (SA), or “fast adapting” (FA) if respond to dynamic stimuli.

Figure 1.3: Tactile receptors in the skin, from [49]. The SA-I mechanoreceptors correspond to Merkel cells, SA-II to the Ruffini endings, FA-I to the Meissner corpuscles, and FA-II to the Pacinian corpuscles.

In fact, tactile feedback has been shown to be particularly useful for in-hand manipulation [37, 7]. In [33], a control framework with a low-level joint controller, a higher-level haptic controller, and a high-level visual servoing controller, work together to provide manipulation and exploration of new objects. Several works have focused on the usage of tactile feedback for object detection [46], estimation of dynamic properties [43], extraction of generic tactile features [9] or increasing grasp robustness against pose uncertainty [14].

In [52], the authors argue that manipulation of unknown objects requires compliance and tactile sensing. A soft-robotics gripper that provides embedding pressure and strain sensing [18] is thus a good candidate to implement dexterous manipulation. A review on tactile sensors can be found in [48], §19.
Feedback control was already suggested to improve grasping in 1985 [3] using infrared sensors to detect distance. More recently, proximity sensors in the fingertips of a Barrett Hand and TUM-Rosie were used to perform reactive online grasping [27] and measure distance to objects [35], respectively. In general, distance sensors are helpful while approaching an object, either embedded in robotic hands [3, 27] or skin [36]. In fact, [21] uses both distance and force, by employing capacitive sensors that measure both proximity of conductive objects as well as pressure after contact. A multi-modal skin that incorporated optical proximity and capacitive pressure sensing was introduced in [36].

Here, we demonstrate a number of the above capabilities with a simple, low-cost array of combined pressure and distance sensors, thereby significantly improving Baxter’s capabilities. The grasping approach used at the APC followed the traditional approach of decomposing tasks in a planning and execution stage [14], where the generated grasps and collision-free trajectories are subject to inaccuracies due to eye-to-hand calibration errors. Those errors can be corrected via haptic feedback control, and we explore the possibilities.

1.3 Outline and contributions of the thesis

In chapter 2, we describe the hardware and software tools employed, with a focus on the tactile sensors and the tactile information that they provide. In chapter 3, we contribute a robust system that provides a grasping pipeline to allow quick and easy prototyping and testing of new approaches involving feedback control. To show its extendibility, we port it to run in two different robots, Baxter and Jaco. In chapter 4, we provide an experimental demonstration of the benefits provided by a tactile gripper that incorporates combined proximity and force sensors developed at the Correll lab [38]. In chapter 5, we present a modified version of the sensor that targets more accurate contact detection and wider dynamic range for force measurements, to be used in prosthetic applications. This version removes an unwanted dependency on external surface properties when only force data is desired. However, the experimental data does not follow a previous model developed for the unmodified sensor [38] that depended on the external surface properties. Thus,
we hypothesize an alternative explanation.
Chapter 2

Materials and methods

We present the hardware employed, with a focus on the tactile sensors used. We also give an overview of the software stack upon which we build our experiments and pick-and-place pipeline, which is detailed in §3, and the processing of the tactile signals from the sensors.

2.1 Hardware

The main robot used during this thesis was Baxter, from Rethink Robotics [20]. We mount a parallel electric gripper on its left arm, and an augmented reality tag on its right arm, which is used for calibration purposes.

The parallel gripper is a standard accessory for Baxter. However, it has been improved with a set of 8 combined proximity and force sensors in each finger (figure 2.1), that are described in the following section. Seven sensors point towards the other finger, and the last tip sensor points along the finger direction.

We also ported the software to run on Jaco, a more accurate, precise—and expensive—arm developed by the Canadian company Kinova to assist disabled patients in daily tasks. In this case, the AR tag is mounted on its wrist (figure 2.2).

To capture RGB-D data, we use an ASUS Xtion Pro camera.
Figure 2.1: A view of Baxter’s electric parallel gripper, schematically and in real life. There are 8 sensors embedded in PDMS in each finger. The IR receiver-emitter (VCNL 4010) are the black integrated circuits over the white board.

2.1.1 Tactile sensor

In order to improve grasping, we integrate information from a combined force and distance sensor. It uses distance measurements from an infrared emitter-receiver that is embedded in an elastomer, typically Polydimethylsiloxane (PDMS). This novel sensor was introduced in [38], but we will briefly describe it here for completeness.

The sensor can measure distances (0–10 cm) and forces (0–5 N). It consists of a small integrated proximity and ambient light sensor (VCNL 4010, Vishay Semiconductors). Eight of these sensors can arranged using an I\(^2\)C multiplexer (TCA9548A, Texas Instrument). Communications use an I\(^2\)C bus, and at 100 kHz of bus frequency, a strip of eight sensors can be read at 85 Hz. We read the sensors at 20 Hz.

The thin layer of PDMS (Dow Corning Sylgard 184) permits force measurements, by deforming elastically so that the effective distance between the object and the IR receiver decreases as a function of the force.

This sensor is both simple and inexpensive, and permits several modifications to measure other quantities (like shear, or more accurate force), as we will see in in chapter 5. There, we
modify it so that it can measure forces up to 50 N, an order of magnitude more over the results presented in [38]. That modified sensor is insensitive to proximity, but that is not a big burden since it is simple to package several sensors close together, modifying some but not all to only measure force.

2.2 Software

Baxter provides a Software Development Kit (SDK) which enables full control of the robot functionality from the Robot Operating System (ROS).

Despite its name, ROS [41] is actually a robotics framework that runs on top of more common operating systems, mainly on linux. It provides a communication layer over heterogeneous and distributed computing resources. A typical ROS system has nodes that can run in different host machines. These nodes can publish messages on specific topics, subscribe to other topics so that callbacks are executed upon message reception, and provide or request services. New message types are straightforward to define, but a number of useful types, like twists or IMU data, are easily accessible.

Each ROS node can be written in different languages, and I personally saw nodes written in C++, Python and Lisp during the First Amazon Picking Challenge. Even other languages can interoperate using wrappers. For example, the Julia language [4] has a package that wraps the Python ROS library. We extensively use the capability to easily mix and match programming
Figure 2.3: An illustration of the way the sensor works (from [38]). The infrared light emitted is reflected both at the PDMS-air interface, and by nearby objects. When an object presses against the PDMS, its width changes.

languages, writing nodes that had to perform extensive computations in C++, but keeping all the complex program logic in Python, using its powerful features to maintain a readable, extendable code base.

A wealth of tools have been developed around ROS, which is free and open source. We use RViz to visualize and interact with 3D point clouds and the state of our robot. We also plot signals in real time using rqt_plot.

We use an Arduino Uno to interface the sensors—which speak in I2C—and ROS nodes—which speak using ROS messages (figure 2.4). This board initializes the sensors, reads their output, and relays the information via serial. The Arduino language is a set of C/C++ functions that can be called from sketches (source code files). Those sketches require small modifications before being sent to a standard compiler like avr-g++, like automatic generation of function prototypes.

2.2.1 Tactile information

Pressure data can be processed to obtain signals equivalent to SA-I and FA-I [42]. SA-II signals, which respond to shear, can also be extracted from the spatial derivative of a pressure sensor array [24], but we do not study them here. Finally, FA-II can be extracted from acceleration data [42].

An estimate of the SA-I sensory channel, similar to the total finger force, can be obtained
Figure 2.4: A high level overview of the communications in the pipeline. Most of the code and functionality lives in the ROS layer. The sensor readings, which are published via I^2C and relayed via serial by an Arduino board, also end up being converted into ROS messages, so that they can be consumed by different nodes. The ROS layer can be seen with more details in figure 3.5.

by summing over the seven sensors that point towards the other finger [42]. For the left finger, it would be

$$F_l(t) = \sum_{i=1}^{7} f_{l,i}(t)$$
$$F_r(t) = \sum_{i=1}^{7} f_{r,i}(t)$$ (2.1)

for the right finger.

The FA-I channel can be estimated using a discrete-time first order Butterworth high-pass filter with a cut-off frequency of 5 Hz for a 20 Hz sampling rate of the sensor:

$$\tilde{F}(t) = \sum_{i=1}^{7} (h_f(t) * f_{l,i}(t) + h_f(t) * f_{r,i}(t)).$$ (2.2)

Like in [42], the FA-II channel is approximated by taking the magnitude of the filtered three-dimensional accelerometer data from Baxter’s wrist. The filter applied to each of the three Cartesian acceleration components is again a discrete-time Butterworth high-pass filter with a 33 Hz cut-off frequency, experimentally chosen for the 100 Hz sampling rate of the acceleration stream that Baxter provides:

$$\tilde{A}(t) = \sqrt{\sum_{i=x,y,z} (h_a(t) * a_i(t))^2}. $$ (2.3)
Chapter 3

Pick-and-place pipeline

We use a simple pick-and-place pipeline to evaluate the contribution of proximity and tactile information on task reliability and detection of grasp state transitions. The pipeline consists of several independent nodes that perform specific tasks, like eye-to-hand calibration or object detection, and a main managing node that specifies the experiment and coordinates every other node.

3.1 Grasping state machine

The grasp pipeline is implemented as a finite state machine. A visual representation of the different states is shown in figure 3.1, and a diagram of the phases during pick and place, from the initial camera-to-hand calibration to the release of a grasped object can be seen in figure 3.2.

3.2 Calibration and perception

We use an Augmented Reality (AR) tag\(^1\) rigidly mounted to one of Baxter’s arm to find the transformation between Baxter’s frame of reference and the 3D camera. A single arm pose to do so isn’t usually accurate enough, so we find the best calibration over two or three arm configurations (figure 3.3).

After obtaining a calibrated point cloud, we need to segment the objects to be picked up. We implement a simple perception pipeline that uses the Point Cloud Library (PCL). The table top is segmented out employing random sample consensus (RANSAC) [19]. Objects on top but

\(^1\) http://wiki.ros.org/ar_track_alvar
Figure 3.1: Different phases of grasping, with their associated events (from [39]). Clockwise, from the top left: (1) Approach, (2) Alignment, (3) Contact, (4) Lift, (5) Shear/slip, (6) Disturbance, (7) Placement, and (8) Release.

Figure 3.2: A high-level description of the different phases during a pick and place routine, including the transitions that are driven by either position or in-hand sensing. Although we can detect the events that characterize the transitions that could use the sensor, not all of the transitions are driven by the sensors: some are manually controlled.
Figure 3.3: Calibration process, from left to right. Initially, the 3D point cloud given by the camera is given with respect to the camera frame, but the pose of that frame is unknown in the robot’s own base reference frame. Next, we acquire the AR tag’s pose both from the robot’s base (through the joint encoders), and from the camera’s reference frame. This enables calibration, but typically we move the arm to a new pose, and compute the transform from the robot’s base to the camera that minimizes the squared error over several calibration poses.

Figure 3.4: On the left, view from RViz, where the calibrated point cloud is superimposed to a model of the robot. Note the coordinate frames on each cubelet, computed by the perception node; as well as the AR tag, that enables eye-to-hand calibration. On the right, the view seen through the ASUS Xtion Pro at the same timestamp.
close to the table are then segmented using Euclidean clustering. Albeit this basic approach cannot
differentiate objects that are touching each other, it provides the pose of simple cubic objects (the
poses can be seen in figure 3.4, on the left) with an accuracy that is comparable to state-of-the-art
approaches for object localization [34] and is limited by the resolution and accuracy of Kinect-like
RGB-D sensors.

3.3 Pick and place

For testing, an user can select an object to be picked up by entering its number (super-
imposed over each object), and the robot will position one of its arms in a pre-grasp pose using an
inverse kinematic solver and then execute the state machine shown in Figure 3.2. After successfully
grasping the object, the arm retracts back to its pre-grasp pose, and transitions to placing the
object. The user can now choose where to place the Cubelet, which could be anywhere on the table
or on top of an existing tower of Cubelets.

An overview of the different nodes involved in the pipeline can be seen in figure 3.5.
Figure 3.5: ROS nodes used to run the experiments. They are independent from another, so provided that they maintain the same interface (i.e., use the same message types and topic names), each component can be modified, or even entirely replaced without affecting the rest of the pipeline. This should make it easy for different people to work on the parts that interest them most without having to understand the full setup. The “finger sensor” node reads data from serial, and publishes it. The “artificial mechanoreceptors” node computes the FA and SA signals, which we visualize in real time use rqt_plot (not shown in the diagram). The manager nodes defines the experiment to be performed, encoded as a finite state machine. It also initializes the robot used, and employs tactile signals to drive some state transitions. Calibration and perception have been the most stable nodes throughout the development, as they are not part of our current research.
Chapter 4

Experiments

There exist many different tactile sensors [12], but they are not widely used, either in research or industrial robotic platforms. A single team, out of 24, used tactile sensors at the APC [10]. There are two main reasons for their lack of usage: tactile sensors are expensive, hard to manufacture, and the data they provide lacks easy ways to use it [13, 9]. There are exceptions in concrete cases, like slip detection [24] or tactile exploration [26].

The sensor introduced in [38] and briefly presented earlier in §2.1.1 is easy to build and inexpensive. In this chapter, we focus on using that sensor to improve grasping. Figure 4.1 shows some actual sensor values when holding a cubelet. We experimentally show that sensor data improves success, both during grasping and other tasks like building towers, and we study a number of events that take place during grasping. See [39] for some extra experiments and discussion.

4.1 Results

We design an experiment that uses proximity and force information to improve the success of building a tower of cubelets [47]. This tasks exercises the complete pick-and-place system, as it involves picking cubes from a table, and placing them on top of other cubes. We also record tactile signals, and show that there are very specific signals associated with different events.
4.1.1 Tower building task

We compare two versions of the tower construction task to see how each individual improvement benefits a more complex exercise.

The first one uses only RGB-D data, with no reactive control. Thus, the object poses are computed by the perception node, and the arm follows the grasping state machine using only position, moving via its own inverse kinematics solver.

The second version adds reactive control based on the tactile data. In particular, feedback is used during picking to align the gripper and the target cubelet. It is also fundamental during placing, to correct position errors in the existing tower.

All the experiments were run with the same hand-to-eye calibration, and the differences in performance are solely attributed to the tactile/proximity sensors. Every experiment starts with a set of cubelets distributed over the table (figure 4.2), and we consider it successful when a 3-levels high tower has been built, or failed when either grasping or placing any of the cubelets fails.

Maybe surprisingly, this task turns out to be quite difficult for an inaccurate system consisting of RGB-D sensing and the Baxter robot. A tower of two blocks worked only in two out of ten
Figure 4.2: A successful attempt at building a tower of cubelets. From left to right, the initial disposition of the cubes, after stacking a cubelet, and after making it to three levels high. Reactive control was employed, and it was necessary to make it successful trials. A typical failure mode is to try to execute a grasp from a position that is too high, due to the combined inaccuracy of the perception and motion systems, leading to a failed grasp. Another common failure would be a cube placement either too high or too low, so the cube either falls over the tower, or the gripper collides against the table or the existing tower. The results of the experiment are summarized in figure 4.3.

When the sensors are being used to provide feedback control during picking and placing,
the failure modes are quite different. Picking is essentially always successful: the finger sensors are capable of correcting the uncertainty in the poses provided by the vision system, and all the grasps are consistently flush against the table. Most failures happen when placing the cubelets. A common cause is due to the magnetic snaps of the cubelets. When the system is trying to center the following block on top, the magnetic forces between cubes may drag the existing tower, making it impossible to center the next block. A similar failure happens when the tower rotates a few degrees and then snaps, but the faces are no longer aligned. A higher success rate could probably be achieved with another type of blocks that do not snap together, like simple wood cubes.

### 4.1.2 Centering for grasping

During tower building, centering is not overly important as the cube will end up in the same position when the fingers close. However, in other tasks, like handling top-heavy objects or playing Jenga, it is necessary to ensure that both fingers make contact at the same time to avoid disturbing the pre-existing equilibrium.

To demonstrate how the finger sensors can help in these tasks, we use a simple proportional controller to center the object inside the gripper before closing it. This is an example of a task where a calibrated distance sensor is not much more helpful, and in fact we use the raw sensor values to perform the centering.

We start with a tower of 1-inch cubes from the Yale-CMU-Berkeley (YCB) Object and Model Set [6], and the robot has to remove cubes from it. The block position is hard coded, but random noise is added to it, and we observe the differences between keeping the centering controller enabled or not. Noise is drawn from a uniform random variable (from $\pm 0.1$ to $\pm 0.7$ cm). The results after trying to grasp cubes 20 times (with and without using the proximity sensors) are shown in figure 4.4 (right). Performance from the feedback-less approach drops with very little amounts of noise, but the centering controller makes it robust against up to $\pm 0.5$ cm of noise in the block position.
4.1.3 Tactile events

We observe that transitions in the state machine model of grasping show characteristic patterns on the tactile channels (FA-I, FA-II and SA-I). We would like to drive the state machine from the tactile signals, and here we collect data for two of these events. More events were analyzed in [39].

A contact event signals the transition from approaching the object to gripper-object contact. This is a high frequency event, thus seen on the FA-I channel (eq. 2.2) and plotted with the other tactile channels in figure 4.5.

A placement event corresponds to an in-hand object touching a table, like during object placement. The FA-I channel is shown in figure 4.6, where the signal peak at placement corresponds with a shear sensation, and the subsequent oscillations represent vibrations due to the arm pressing the object against the table.
Figure 4.5: Signals during a contact event (from [39]), where the gripper closes against a cubelet.

Figure 4.6: FA-I signals during a placement event (from [39]), where a block held by the gripper touches the table. On the left, a single event. On the right, the overlap of seven such events, aligned using their cross-correlation.
Chapter 5

Force calibration: the unexpected

The initial development of the combined proximity and distance sensors was presented in [38]. In chapter 4 we’ve shown those sensors can dramatically improve success rates in various tasks. There, we mainly exploit the distance readings to achieve a better performance, and we also observe that the filtered signals which mimic human nerve response can detect the events that drive the pick and place state machine. However, force readings suffer from a strong dependency on surface properties (those of the object that is touching the sensor’s PDMS), and it is not straightforward to differentiate the distance regime from the force regime.

To tackle the second issue, [38] proposes the detection of an inflection point in the signal values over time, just as contact separates the distance and the force regimes.

To avoid the first issue, we decide to cover the PDMS with a material opaque to IR radiation. Thus, the reading stops depending on the outside-object surface, and force can still be measured from the elastic changes in the PDMS thickness.

This adaptation emphasizes the flexibility of the developed sensors, which can also be easily adapted to measure shear by embedding two layers of lines in the PDMS, so that shear efforts on the PDMS surface change the relative position of the two grids, changing the amount of light reflected back.

In the following sections, we will motivate the specific application that required in-hand force sensing, detail our experimental process to find an adequate material to cover the sensor and the observed results.
5.1 Prosthetic hands

For people who use a prosthetic limb, and in particular prosthetic hands, providing them with a little bit of extra information is the missing step to properly integrate their usage.

Jacob Segil is developing a myoelectric prosthetic hand. Using a neural interface, this hand can transmit signals between nerves and the computers in the hand, which themselves control small motors to allow finger control and object grasping. Signals can also be relayed towards the patient’s brain, and we are working with him to use sensors based on [38] to detect that information.

Providing the prosthetic users with distance measurements from their finger tips, and hoping that the brain’s plasticity adapts to them, can be quite cool. In fact, maybe at some point in the future people will implant tiny chips to increase their sensing capabilities. Currently, however, force and contact are the two primordial metrics.

5.2 Removing object’s surface dependency

Thus, we adapt the sensor to measure only force, which also enables us to do more accurately. The elasticity of PDMS is the mechanism that we use, schematically drawn in figure 5.1.

A board to house the sensor (figure 5.1) was designed by John Klingner, embedded in a 3D printed finger designed to fit Jacob’s hand, and covered with molded PDMS. Each board contains two sensors, one is proximal and the other one is distal.

![Figure 5.1: Some of the finger boards, before the components are soldered to them.](image)

The myoelectric interfaces can’t transduce very nuanced values, so we aim to differentiate
and transmit three different force regimes (low, medium and high force), instead of sending exact force values. We target a maximum force of about 90 N.

We experiment, in an unstructured way, to find a material that works well as a PDMS coating. In [38], the force readings had some limitations that make them unfit. For example, the forces that could be measured were between 0 and 5 N, which is not enough. The cause is that the receiver would saturate, reading its maximum value at any larger forces. As discussed earlier, the readings also depend on the surface properties of the object in contact.

Guided by that previous work, we initially focused on “dark” materials. Observe that an object that is black in visible light is not necessarily opaque to infrared, as seen in figure 5.2

![Figure 5.2: Infrared light can travel through dust or smoke, and even some materials like the plastic bag that is clearly visible on the right side. Public domain pictures from NASA/IPAC (http://coolcosmos.ipac.caltech.edu/cosmic_kids/learn_ir/index.html).](image)

Therefore, we ended up testing several materials that we had easily available in the laboratory. For example, we covered the sensors with latex, vinyl, nitrile, several types of fabrics, and foams. We wanted to quickly test as many materials, discard the ones that didn’t work well, and refine the experiments when results showed promise. Thus, we conducted manual testing as follows: we would cover PDMS on the 3D printed finger in a stable way (keeping it in place with some thread), and manually press the rubber with our fingers while observing a real time graphical representation of the measured values.

Typically, the behaviour we observed was sensor saturation: either immediately after the coating material touched the PDMS, which means that it is reflecting most of the received IR, or with very light pressure. This is similar to what was described in [38], and like in that situation, it would only enable the measurement of relatively small forces (up to 5 N). One benefit is that
the coating would remove the sensor’s dependency on the surface properties of the object that is exerting the pressure.

Surprisingly, one of the tested materials exhibited a very different behaviour, not only quantitatively but also qualitatively. The material is nitrile rubber, extracted from a pair of “BlackNitrile Examination Gloves” manufactured by Dynarex Corporation. For this material, the sensor readings decreased as force was increased. That behaviour is the opposite of what we observe in every other material we tested, and the model proposed in [38] cannot explain it.

5.3 Calibration results

Given the surprising results, we refine the experiments for the nitrile coating. The finger, together with the interface board and Arduino board that connects it to a PC, can be seen in figure 5.3a. We use a digital force gauge (FGV-10XY, Nidec-Shimpo Corporation) seen in 5.3b, with a circular tip that has a 10 mm diameter, to compress either the proximal or distal rubber.

(a) Experimental setup. (b) Digital force gauge.

Figure 5.3: On the left side, the 3D printed finger can be seen. It is connected to a multiplexing board, and it talks to the Arduino board via I²C. On the right, the force gauge used.

To obtain the measurements, we press the tip of the force gauge against the PDMS, on top of either finger sensor. We observe a good separation between the two sensors, and we are able to excite one or the other independently. A linear model (figure 5.4) is good enough to model the relationship between force and raw sensor value ($R^2 = 0.9$ for both the proximal and distal sensors).
5.4 Possible explanations

We do not have a convincing explanation for this behaviour. In infrared spectroscopy, nitriles–organic compounds characterized by a $\text{C}�单=\text{N}$ functional group–have a medium-intensity, sharp absorption due to stretching between $\tilde{\nu} = 2200$ and $2300 \text{ cm}^{-1}$ [50]. The wavenumber, as used in infrared spectroscopy, is defined as

$$\tilde{\nu} = 1/\lambda,$$

and it is often given in units of $\text{cm}^{-1}$.

The integrated infrared emitter in the VCNL4010 has a peak wavelength of 890 nm, which corresponds to a wavenumber of 11200 $\text{cm}^{-1}$. This is part of the near-infrared region, where the absorption bands correspond to overtones [50] (multiples of the fundamental absorption wavenumbers). In our case, $11200 \approx 5 \cdot 2250$, but [50] also mentions that the overtone absorption bands are usually weak. We have been unable to find a spectrum for nitrile that arrives to the near-infrared, most likely because that region is less useful in different analysis, so they tend to stop at around 4000 $\text{cm}^{-1}$, far from the 12000 $\text{cm}^{-1}$ that we would need (a nominal nitrile spectra can be seen for example in chapter 12, at [53]).

At any rate, an absorption band located near the emitters peak wavelength could explain why the sensor doesn’t saturate as quickly, but not why the amount of light reflected to the IR receiver decreases as force increases. To explain that, we hypothesize two explanations.

One is that as pressure increases on the finger, the nitrile rubber suffers a mechanical deformation that changes its IR absorption profile, decreasing the amount of light reflected back. Changes in mechanical stress can be detected in fibrous materials, like wood [44], via IR spectroscopy, so it is reasonable to expect a polymeric material like nitrile to have a different response under stress. This hypothesis could be tested by performing near IR spectroscopy on two samples of nitrile, one of them under compression stress.

Another possibility is that nitrile scatters IR light, and as pressure is increased and the PDMS is compressed, scattering increases and less light is reflected back.
Figure 5.4: Force applied with the digital force gauge vs the raw sensor readings. Each color represents a different trial. To record each point, we maintained a constant force for a few seconds (≈ 100 samples), and the (small) error bars in $x$ show their standard deviation. For the force, the error bars represent ±1 N.
6.1 Machine learning for grasping

All the grasps performed for this thesis were programmed manually, in the sense that none of the experiments were performed independently by the robot after a training period.

Deep learning techniques have shown impressive capabilities to recognize patterns [?], and achieve state-of-the-art performance in image recognition [23], language parsing [2] or generation of image captions [29]. They have also been used to find good grasps for parallel grippers from RGB-D data [31]. Also, those techniques have been used to learn grasps in unsupervised environments, after tens or hundreds of thousands of trials [32, 40].

Reinforcement learning [51] deals with agents which need to take actions to maximize certain rewards. It was introduced to grasping as early as 1993 [17], and has been used to learn grasps from vision [45] and also from tactile information [7], and to learn manipulations in [52] with two force sensors in their fingertips. There, a Markov decision process (MDP) formalizes the manipulation task, which is particularly useful because the grasping state machine in figure 3.2 is the first step to establish an MDP. Reinforcement learning has also been used to grasp objects in cluttered environments [5], but the state of the system is given by a 3D point cloud, with no tactile information.
6.2 Detection of state transitions

In chapter 4 we have seen that some state transitions have very characteristic signatures in the tactile signals (FA-I, SA-I and FA-II). Currently, our pick-and-place state machine is not fully driven by those signals, but we believe that data is sufficient to completely drive it in the future.

To do so, we expect that a classification algorithm trained with labeled data should be able to autonomously follow the state machine, knowing when to repeat a step because it wasn’t successful. Following the advice in [1], three machine learning algorithms stand out and should be tried: random forests [25], XGBoost [8]–a scalable tree boosting system–and of course neural networks [22].

Recording correctly labeled data from the grasping transitions is fundamental to train a working classifier, but it can also be a tedious task. It is thus necessary that a user-friendly system be put in place to obtain the training dataset.

The input data is sequential, and different methods exist to deal with that [15]. We would favor the sliding window algorithm [15] as an initial step, given its simplicity. Its length is a hyperparameter that needs to be tuned, but it should be the same order of magnitude than the signal to be detected, typically a few tens of a second.

For the first two algorithms in particular, choosing the right features to feed either one is the first and most important step. A straightforward feature vector would be made with some of the first statistical moments (i.e., mean, skewness, kurtosis, etc), augmented with other features like range. It would be interesting to start with as many features as possible, and then reduce them based on their importance [16].

6.3 Calibrated force sensing

There is a clear trade-off in our modified sensor from chapter 5. We obtain force readings that are calibrated and also increase the reading range 10 times. To do so, we lose the multimodality of the sensor, since it can no longer measure distance. An important loss is the ability to reliably
measure contact, as the modified sensors need to press against the object to detect contact. The original multimodal sensors don’t need to disturb it.

An option to overcome this problem is to package the two sensor versions together, providing both proximity and calibrated force readings.
Chapter 7

Conclusion and future work

We have seen that tactile feedback is fundamental to enable higher quality grasps, and to perform complex tasks with higher success rates. We have shown that simple, cheap tactile sensors that provide distance and force readings can be used to improve grasping. Also, their readings (processed to mimic certain human nerve signals) identify a number of events, like contact or placement. We have presented a simple modification of the sensor that unexpectedly provides higher force sensitivity and $10 \times$ greater range.

Grasping is still a challenging problem, and it is likely that manually coded grasping techniques will be relegated by automatically learnt methods [40, 32]. However, integrating tactile feedback into those pipelines will make them work better, and is a very exciting area of future research. The pick-and-place pipeline presented here should be flexible enough to allow the grasping state machine to be driven using machine learning methods.

Moreover, expanding the range of measurements that the finger sensors can take (like shear forces) has some fabrication challenges, and using that information to improve grasping is also an area of active research.

The integration of the sensors (modified to measure only force) in prosthetic hands is an ongoing work, with lots of possibilities.
**Bibliography**


[34] Lu Ma, Mahsa Ghafarianzadeh, Dave Coleman, Nikolaus Correll, and Gabe Sibley. Simultaneous localization, mapping, and manipulation for unsupervised object discovery. In IEEE Int. Conf. on Robotics and Automation (ICRA), pages 1344–1351, 2015.


