Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Manipulation with a Robotic Gripper Using Tactile Information

Lucie Vajnerová

Supervisor: Prof. Ing. Václav Hlaváč, CSc.
Field of study: Cybernetics and Robotics
May 2021

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Vajnerová Lucie**	Personal ID number:	**483515**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Manipulation with a Robotic Gripper Using Tactile Information

Bachelor’s thesis title in Czech:

Manipulace robotickým chapadlem využívající taktilní informaci

Guidelines:

1. Make yourself familiar with 3FG, Takktile sensors and their control.
2. Attach the gripper to a force-compliant industrial manipulator and use ROS for working with it.
3. Solve a simplified peg and hole manipulation task and develop related software. Python environment is recommended.
4. Document your work and software well.

Bibliography / sources:

[1] R.M. Murray, Z. Li., S.S. Sastry: A Mathematical Introduction to Robotic Manipulation, CRC Press, 1994. 456 p.
[2] M.T. Mason: Towards Robotic Manipulation. Annual Review of Control, Robotics, and Autonomous Systems, Vol. 1, 2018. 28 p.

Name and workplace of bachelor’s thesis supervisor:

prof. Ing. Václav Hlaváč, CSc., Robotic Perception, CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **05.01.2021** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

prof. Ing. Václav Hlaváč, CSc.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

Over the course of my work on this bachelor thesis project I have received a great deal of support and assistance.

I would first like to thank my supervisor, Professor Václav Hlaváč, for his invaluable expertise and support over the course of the project.

I would also like to acknowledge Ing. Libor Wagner for his help and knowledge with ROS-related matters and Ing. Antonín Mík for his assistance with creating our technical tools for the peg-in-hole task.

Lastly, I would like to thank my family for their unending patience with my constant state of panic and stress.

Declaration

Abstract

We seek to solve a robotic assembly peg-in-hole task using only feedback information from tactile sensors and a force/torque compliant robot. Tactile sensing is less used in industry than visual-servoing, due to its lower endurance and reliability. We attempt to understand if deeper knowledge of a problem and new tactile sensors technologies can boost their application in industry. We aim to use Robot Operating System of the next generation (ROS 2). We studied first the approaches of others over the last 50 years as well as the current methods. We introduce our methods and solutions and explain their applications. We evaluate our result and analyze its limitations. We conclude optimistically that the tactile sensors development and the available force/torque compliant robot has potential from an industrial point of view. Our implementation was hindered by the COVID-19 constraints which prevented us from experimenting with a real robot Franka Emika Panda. The available ROS 2 version did not support all we needed. The new ROS 2 version, announced for release in June 2021, should solve several issues we had.

Keywords: robotic manipulation, tactile sensing, ROS

Supervisor: Prof. Ing. Václav Hlaváč, CSc.
CIIRC CTU, Praha 6, Jugoslávských partyzánů 1580/3

Abstrakt

Snažíme se vyřešit klasický dílčí úkol robotické montáže, tj. zasouvání kolíku do díry (peg-in-hole), a to pouze pomocí zpětné vazby z taktilních senzorů a při použití silově/momentově poddajného robotu. Hmatové vnímání je v průmyslu méně používáno než vizuální servo, především kvůli nižší trvanlivosti a spolehlivosti. Pokoušíme se pochopit, zda hlubší znalost problému a novějších technologií hmatových senzorů může pomoci jejich většímu průmyslovému zapojení. Naším implementační cílem je použít robotický operační systém nové generace (ROS 2). Nejprve jsme studovali přístupy ostatních, a to jak během 50 let své historie, tak i současných metod. Představujeme své metody a řešení a vysvětlujeme jejich aplikace. Vyhodnocujeme naše experimenty a analyzujeme jejich omezení. Dospěli jsme k optimistickému závěru, že moderní taktilní senzory ve spojení se silově/momentově poddajnými roboty mají silný potenciál z hlediska budoucích průmyslových nasazení. Naši praktickou implementaci ovlivnila omezení způsobená pandemií COVID-19. Nemohli jsme experimentovat s robotem Franka Emika Panda. Také dostupná verze ROS 2 nepodporovala vše, co jsme potřebovali. Nová verze ROS 2 ohlášená na červen 2021 by měla několik nynějších potíží vyřešit.

Klíčová slova: robotická manipulace, taktilní vnímání, ROS

Překlad názvu: Manipulace robotickým chapadlem využívající taktilní informaci

Contents

Figures

Chapter 1

Introduction

This bachelor project and thesis contributes to CIIRC CTU Robotic Perception group effort to progress in robotic manipulation tasks using tactile information sensed in a robot gripper.

Our assignment has been to perform autonomously a traditional robotic manipulation task called peg-in-hole using a 3-Finger Adaptive Gripper (3FG) from the company Robotiq, haptic sensors for pressure feedback from company Takktile and the Robot Operating System (ROS, a widely used open-source robotic software middleware). Our motivation has been to master the state of the art in sensory supported robotic manipulation exploring tactile sensing and a force/torque compliant robot. Tactile sensing has been much less used in industry in general compared with computer vision based feedback, e.g. visual-servoing. Tactile servoing has been less reliable due to tactile sensors limitations. The research conducted in the lab aims at better understanding why is it so and if sensor deficiencies can be replaced by knowing more about the semantics of the particular assembly task.

We had the opportunity to choose between KUKA iiwa LBR R700 model and Franka Emika Panda available in CIIRC CTU labs. We selected the latter because it is supported better in ROS 2 (advanced version of ROS).

The peg-in-hole manipulation task has been addressed since industrial robotics were born in 1960s. We aim to solve the peg-in-hole task using tactile feedback only. We are able to utilise the equipment technology available at CIIRC CTU. We follow the traditional strategy which decomposes the peg-in-hole assembly into two feedback supported steps: contact-state recognition and force/torque control [24].

We have the privilege to use an expensive ($\approx$ € 10k) adaptive gripper 3FG with three mechanical fingers where each can be controlled separately [1]. Its application includes grasping objects with a set speed and force manually defined by the user. This is possible thanks to the 3FG own current feedback. There are four modes of movement and grasping. Thanks to the wide variety of options, the 3FG is capable of picking up objects of various size and shape. The 3FG has been used in academic researches, but its high cost makes it only rarely appear in industry.

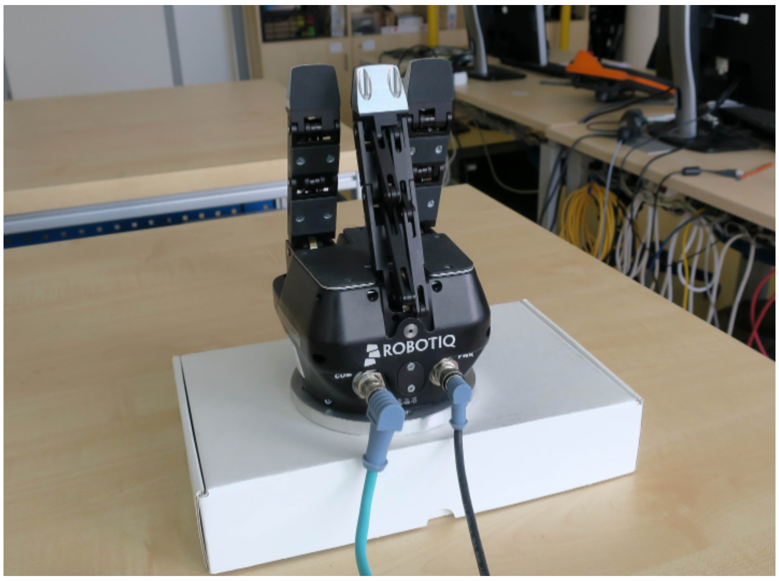
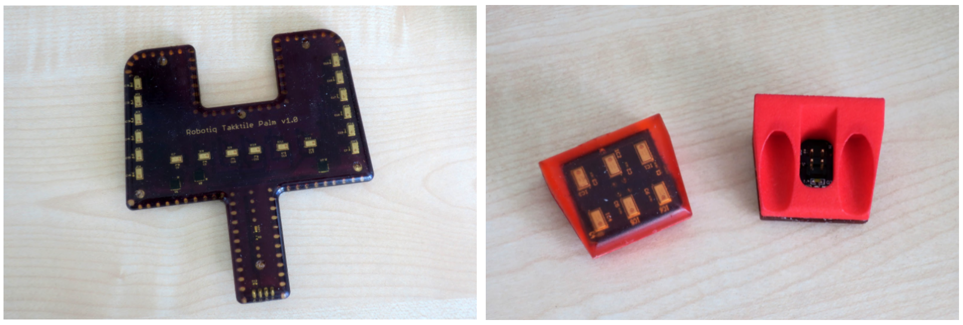


Figure 1.1: Our 3FG photograph

Takktile sensors are a form of tactile technology based on pressure sensing MEMS barometers that are embedded in a rubber casing. In our case, 36 Takktile sensors are mounted on our 3FG, capable of measuring pressure between approximately 0-65 kPa. The Takktile company also provides a communication board using I^2C protocol with a USB bridge. However, our board was destroyed and so we had to work with an Arduino board replacement.

As per its documentation, “The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications” [18]. It is an open-source system used for robot control, giving us a framework we can use to connect our robot to our computer. ROS 2 is an in-development system that aims to replace outdated parts of ROS [15]. ROS 2 adds real-time control of robots, improved communication over different networks, better C++, Python support and more.



(a) : Takktile sensors for 3FG palm **(b) :** Takktile sensors for 3FG finger

Figure 1.2: Our Takktile sensors

Upon instruction from this thesis' advisor, we focused special attention to working with ROS 2, rather than the older variant ROS 1. ROS 2 has been under rapid development by the wide robotic community. However, it has created a caveat for us as some ROS 2 modules have not been fully functional, e.g. ros2_control. Our solution has been to combine ROS 1 and ROS 2 where needed, even if this combination is not straightforward.

The contribution of this thesis is in:

1. Mastering the available gripper (3FG) equipped by tactile sensors (Takktile). We have modified the program of the gripper/sensors microcontroller and documented it for other users.

2. Analysing the use of the gripper with tactile sensors in a peg-in-hole assembly task. We had to manufacture a wooden rectangular peg, i.e. a parallelepiped with matching dimensions to our relatively bulky gripper. We chose the rectangular cross-section of the peg because it is more likely to get stuck in the hole than a circular cross-section peg. Our desire was to make the assembly task more difficult. We used plastics to 3D print the hole. We designed a rather deep hole to stimulate getting stuck[1]. The gripper grasped the peg and we moved the peg by hand, while recording pressures perceived by tactile sensors. We analyzed the data recorded from tactile sensors.

3. We have used our peg and hole in solving the peg-in-hole task with the robot. The COVID-19 restrictions have not allowed us to perform experiments with a real robot. We used a simulator and the model of the Franka Emika Panda robot for the purpose. We have developed the related C++ programs and documented them.

[1]The peg was made by my supervisor. The hole was designed by the technician of our lab, Antonín Mík.

This thesis is organized as follows: Chapter 1 provides the introduction and motivation for the work. Chapter 2 summarizes the assignment given to us. It clarifies the wording of the official brief assignment in more detail and corrects it to match the conditions under which the bachelor project has been developed. Chapter 3 briefly summarizes the related state of the art. Chapter 4 describes our approach, design of methods and their implementation. Chapter 5 evaluates our methods and implementations practically. Chapter 6 concludes the thesis, provides the executive summary and outlines a possible future work on the project.

Chapter 2

Assignment

This chapter extends the formal assignment shown already at page iii. It should help the reader to understand what the diploma thesis' advisor expected from this work. We also include the modifications, which were implemented, because COVID-19 hindered the application of the suggested method to a real robot.

1. Familiarity with the SW environment and 3FG gripper

   Our task was to familiarize ourselves with 3FG (gripper), Takktile sensors and their control. We had to embed the gripper with tactile sensors into ROS environment.

   - We had done the preparatory work on the topic already in the winter semester 2020/2021 while solving a semester project. In the project, we were able to connect the 3FG to a computer, read the data from tactile sensors, perform simple movement of the gripper's fingers.
   - The new work should connect the gripper to the superior control implemented in ROS.

2. Attaching the gripper to a real robot

   We are to attach the gripper to a force-compliant industrial manipulator and use ROS as a middleware enabling its services to a higher-level control. We intended to use ROS 2, despite knowing that some necessary ROS 2 packages are yet in development stages only. ROS 2 enables us to use a secured real-time performance. Due to COVID-19 and technology limitations, we had to constrain ourselves to simulation experiments only.

3. Simplified peg-in-hole manipulation task.

   In this task, a gripper is meant to pick up a peg and carry it over to a hole and place it inside. The feedback from tactile sensors could have been used to help with this manipulation in real life.

   - The formal assignment asks us to connect 3FG (gripper) to the Franka Emika Panda robot.
   - We have to perform a simple peg and hole task using a tactile feedback only.

4. Document the work

   We are tasked with thoroughly documenting our work, the suggested method and developed software. A guide for using 3FG and Takktile sensors has had to be created as well as a GitLab repository for our code to document our work for the simulation.

   - The goal is to make it possible for others to quickly set up their 3FG and Takktile sensors and recreate our work, so that they can continue developing this project further.
   - A development branch has been created on CIIRC GitLab for our simulation and 3FG control.

Chapter 3

State of the Art

In this chapter, we will go over the various strategies of the peg-in-hole task. We will discuss its history and related works of others to ours. We explain basic principles of ROS 1 and explain the development processes of ROS 2 and its differences to ROS 1. In spring 2020, a group of students began working on this project by setting up the 3FG and Takktile sensors.

3.1 Peg-in-hole; Assembly Strategies

A research into various forms of peg-in-hole tasks has been done by a team led by Jing Xu in 2019 [24]. They have divided robotic peg-in-hole assembly strategies into two distinctive types: contact model-based and contact model-free. The contact model-free approach uses learning from environment or through demonstration, thanks to techniques like reinforcement learning methods or interpreting human demonstrations. The contact model-based type of peg-in-hole assembly uses sensing systems to acquire feedback from its environment. The various categories include vision sensors, force sensors and sensor-less systems. A graph of various assembly strategies can be seen in Figure 3.1.

Force feedback is used in active and passive compliance modes. Passive compliance completes the peg-in-hole task using a set of springs and dampers attached to the end-effector, also known as RCC shown in Figure 3.2. Active compliance measures contact forces to build a force feedback. It is also

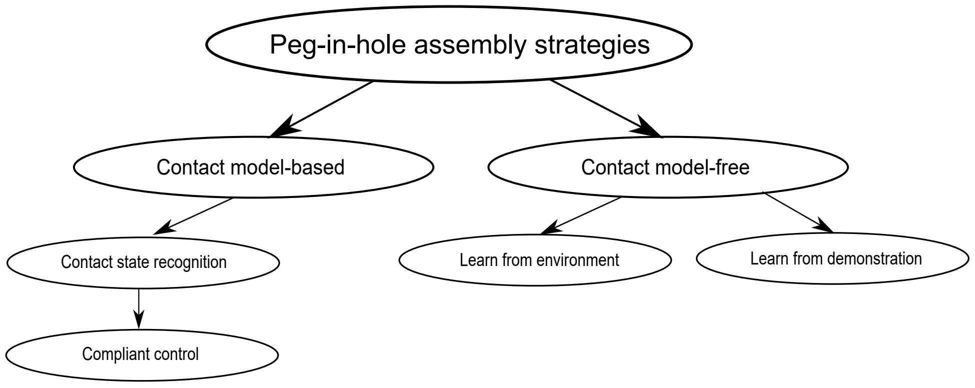


Figure 3.1: Peg-in-hole assembly strategies graph

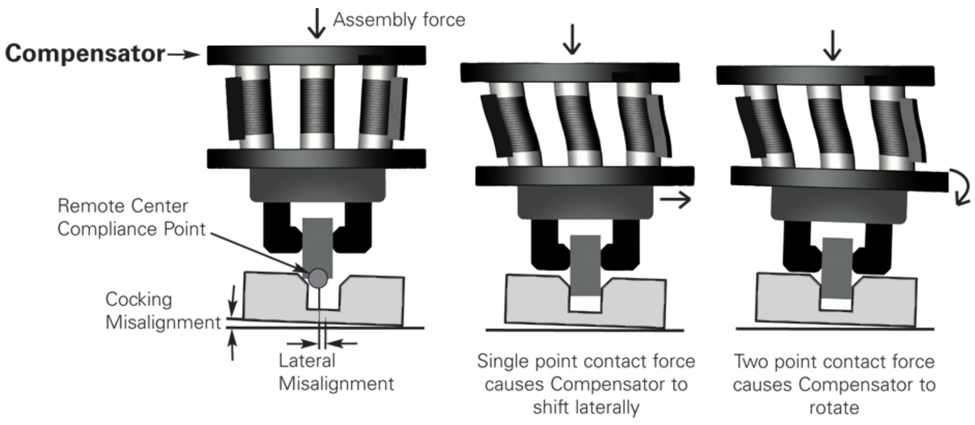

Assembly force
Compensator
Remote Center Compliance Point
Cocking Misalignment
Lateral Misalignment
Single point contact force causes Compensator to shift laterally
Two point contact force causes Compensator to rotate

Figure 3.2: Passive compliance (RCC) [12]

possible to base active compliance on impedance feedback of the robotic arm.

3.2 Peg-in-hole; History

Inserting objects into a hole belongs to basic skills needed in assembly tasks in industry. As soon as the industrial robots appeared in 1960s, the peg-in-hole tasks followed.

In 1975 a team from University of Canterbury conducted research into the peg-in-hole assembly task using tactile methods [23]. They focused on a cylindrical-shaped peg and hole. They discovered that when the peg and the hole come into contact, a reaction force occurs at the point of contact. The reaction depends on several factors: the forcing effort, the peg-hole axes alignment angle, the contact friction and the contact geometry. They deduced that tactile sensing on the peg allows better state recognition than tactile sensing on the hole. The possibility of simple tactile feedback was investigated and found to be better for fine manipulation than vision feedback.

In 1978, Paul C. Watson patented the use of Remote Center Compliance system (RCC), a device used to insert a peg (or other similar objects) into a tight hole without jamming it in the process [20]. The RCC is an example of passive compliance, a method non-dependent on measuring contact forces.

Whitney and Asada assumed chamfered holes in 1982. Whitney [21] divided the assembly process into four parts: chamfer crossing, single-point contact, two-point contact and insertion.

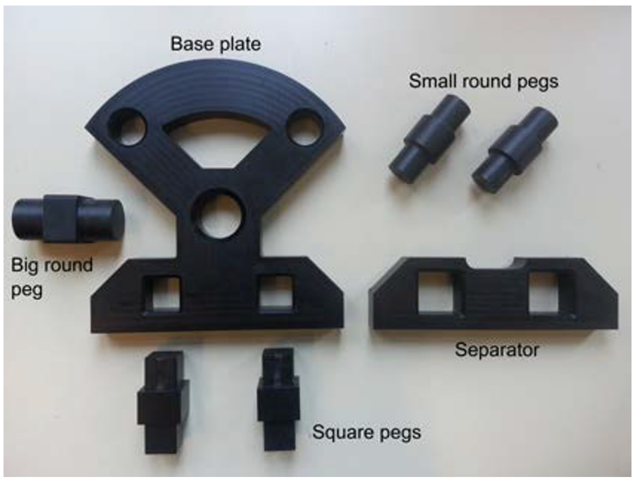
3.3 Peg-in-hole; Modern Approach

Though peg-in-hole assembly is a common industrial task that has been extensively researched, contemporary industry requires much more complex assembly tasks and seeks high precision. Some manual assembly tasks achieve higher flexibility by using collaborative robots and instructive assistance systems, such as remote simplified control over tablets or smartphones [22]. VR (virtual reality) devices can also be used for robotic manipulation, but they are mostly still in their early stages of development.

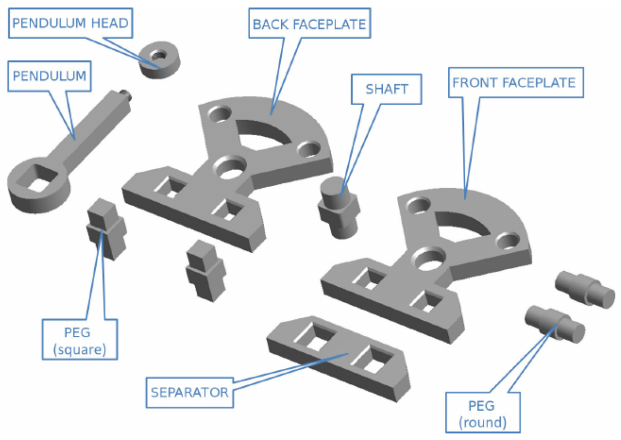
A more complex version of our peg-in-hole task is the Cranfield Assembly Benchmark. This assembly task requires the insertion of different pegs (round and square in cross-section) into the corresponding holes of the base plate and the placement of a separator onto the previously inserted pegs (Figure 3.3). This Cranfield Benchmark was used in many experiments, e.g. in [2], [8]. Both experiments chose the contact model-free approach of learning by human demonstration and achieved a high level of learning speed and precision.

Collaborative robots and the contact-model free approach to solving the peg-in-hole tasks both require human involvement and are not robust to uncertainties or sudden variations to the environment. A possible solution to these issues is to implement a deep reinforcement learning approach. In [3], the choice of SAC (Soft Actor Critic) algorithm led to a high success rate under different conditions like uncertainty of goal position, environmental stiffness and novel insertion tasks. A recurrent neural network such as Long Short Term Memory (LSTM) can be trained using reinforcement learning on a finite number of actions, as presented in [4].

In [5], the focus is on identifying the position of the hole in respect to our robot using only force-guided robotic arm. This is to avoid the low precision of visual-servoing often used in peg-in-hole tasks. Position uncertainty can cause a significant delay in task completion and extra energy expenditure. [5] solves this issue using Expectation Maximization based Gaussian Mixtures Model (EM-GMM) Contact-State recognition scheme that uses captured wrench (the Cartesian force and torque) signals of the manipulated object.



(a) Version from [2]



(b) Version from [8]

Figure 3.3: Cranfield Benchmark Examples

Many types of grippers can be used for the peg-in-hole task. In [19], a research group led by Amirul Syafiq Sadun provided an open-loop grasping analysis of the 3FG (the same gripper we chose to use) and its force feedback. They were using Force Sensing Resistor sensors mounted on the gripper's three fingers and an encoder. They also observed the current of 3FG motors. Their report successfully deduced appropriate sensor response and motion control of the gripper for picking up objects of various stiffness with reasonable sensor response.

3.4 ROS 1 Principles

ROS (Robot Operating System) is a free software system for controlling robotic components from a Linux distribution system. Its main function comes in the form of nodes. A node is a process that does computation. Many nodes can be called simultaneously and act independently of each other. Nodes can be used to control robot motors, observe sensor reactions or plan the robot's path. They use a publish/subscribe messaging model over so called ROS topics (communication channels between the nodes). Publishers send messages while the subscriber nodes read data from topics. There are other ways for communication between nodes, for example the ROS services (a pair of messages with request/reply interactions). In ROS 1, the whole traffic system is controlled by a single master node, which acts as a DNS-server and helps nodes find one another. A simplified visualization of ROS 1 processes can be seen in Figure 3.4.

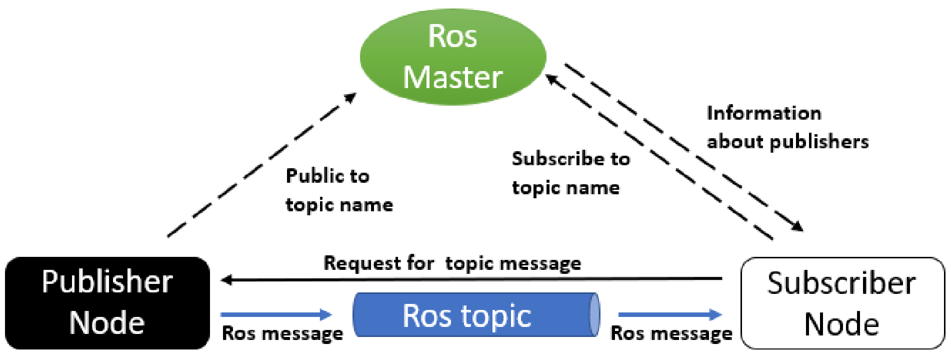


Figure 3.4: Visualization of ROS communication [16]

ROS packages are used to organize software and may include ROS nodes, independent libraries, configuration files or any other form of a usable module. The codes can be written in different languages, the most common being C++ and Python. Due to a large number of publicly available packages and user-friendly systems, ROS is one of the most often-used public middleware for robotics.

3.5 ROS 2 Development

ROS 2 is an improved release of ROS 1 with many modern features. There are few available ROS 2 distributions (a predefined main set of ROS packages).

In 2015, the first beta version of ROS 2 was released to public [14]. The goal is to provide a new ROS 2 distribution at least once a year. The first non-beta distribution was published on December 8, 2017 under the name 'Ardent Apalone' with an EOL (End of Life) date expectancy set at one year. Our work was done under the ROS 2 'Foxy Fitzroy' distribution released on June 5, 2020 with an EOL date in 2023. The newest distribution titled 'Galactic Geochelone' is set to be released on May 23, 2021.

ROS 2 is utilizing the Data Distribution Service (DDS). Unlike in ROS 1, where a master node was necessary and no real-time support was possible, the DDS is suitable for real-time distributed embedded systems thanks to its various transport configurations (e.g., deadline and fault tolerance) [9]. More OS layers are also to be added, making it possible to use ROS 2 on Windows or Mac devices in the future.

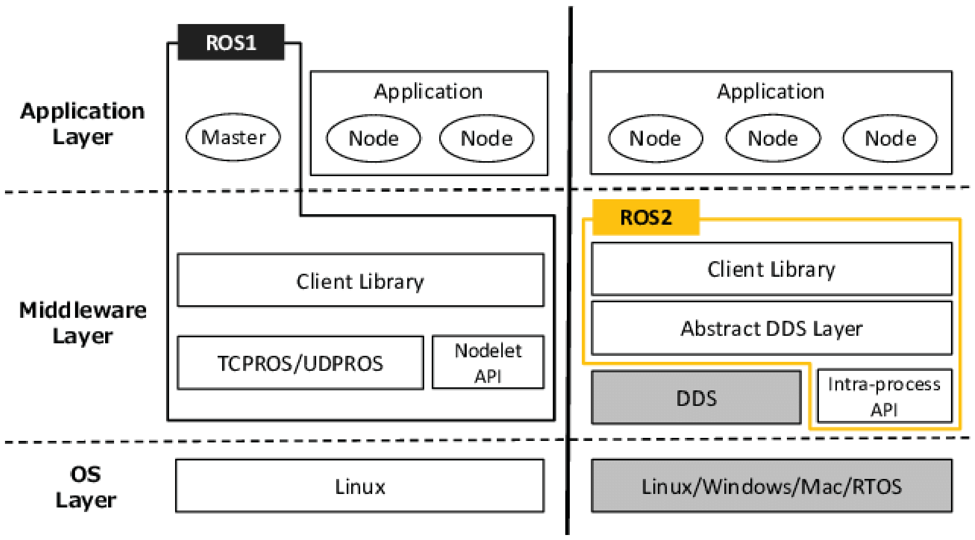


Figure 3.5: Comparison between ROS and ROS 2 processes [9]

The MoveIt software runs on top of ROS and lets us manipulate our robots using ROS messaging and build systems, utilizing such ROS tools like the ROS Visualizer (rviz) and the ROS robot format (URDF) [11]. Motion planning, inverse kinematics, 3D perception, collision checking and more – all is provided in MoveIt itself. MoveIt2 is then simply a straight port of MoveIt over to ROS 2, its first non-beta released on September 4, 2020 [17]. A new MoveIt 2.2 is planned to be released after the 'Galactic Geochelone' distribution in the summer of 2021. MoveIt 2.2 should also provide the much needed ros2_control package integration, necessary for parts of real-life robotic manipulation with Franka Emika Panda robot. Due to these limitations with MoveIt 2.1 and restrictions during the COVID-19 pandemic, we could not finalize our experiment in real life settings and could only simulate our peg-in-hole task.

Despite the still-in-development status of ROS 2, a fair amount of documentation and work has already been done for it. Helpful tutorials and multiple example demos are provided freely to the public. Sadly, most of the documentation is simply a copy of its version for ROS 1, so a fair amount of programming is necessary to implement these options and translate them into an ROS 2 working environment.

3.6 Work of Previous Students on This Project

In the summer of 2020, students Kristýna Míková and Petr Kuželа commenced their work on our 3FG. They mounted the tactile sensors on the 3FG and used an I^2C communication board and a bridge to USB. They tested the capabilities of our tactile sensors with an external pressure dynamometer and documented their results. However, the communication board provided by Soft Robotics Toolkit stopped working after some time and a replacement Arduino board was used from then on instead. The Arduino code they used was based on a program written by Ashrarul Haq Sifat and downloaded from GitHub. Issues with this implementation were found, but not fully resolved. A simplified approach to pressure calculation was chosen to fix these issues.

Their work is carefully documented and their experiments described in the research report [10].

Chapter 4

Our Method and Its Implementation

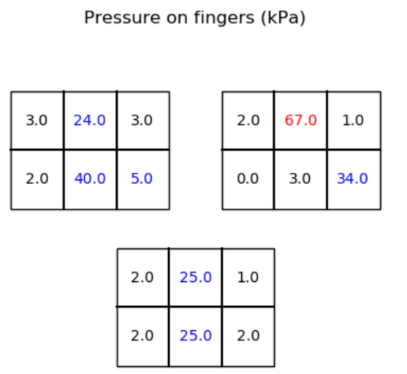
This chapter shall explain our work method, implementations and experiments as we have done them in the chronological order. We first begun our work on the tactile sensors, then moved to the control of 3FG (gripper), then tested holding our peg and sensor feedback and finally worked on the simulation of the peg-in-hole task itself.

4.1 Takktile Sensors

The tactile sensors are mounted on the fingers and the palm of the 3FG. A microcontroller provides communication over a I^2C protocol and uses an integrated 10-bit ADC. With such a connection board, the information from haptic sensors is carried over to the computer, where we can visualize the end results.

Our work began by recreating and setting up our 3FG and the tactile sensors as done previously by Kristýna Míková and Petr Kuželа. The students previously created a Python code that could be used to visualize pressure applied to the Takktile sensors as shown in Figure 4.1a.

We have decided to improve their code. First, we have optimized the code, making it run more effortlessly by removing redundant creation of sensor visual boxes in each iteration and instead only updating the numbers in each loop. A significant delay between the moment of applied pressure and its consequent visualization that previously had a tendency to increase in length, but our modification helped eliminate the issue. Changes in the program for connecting to the Arduino board were made, allowing for repeatable connection should it fail in the first place. Next, we have added 18 palm sensors into our visualization and made the rectangles color changeable based on applied pressure. Our final result can be seen in Figure 4.1b.

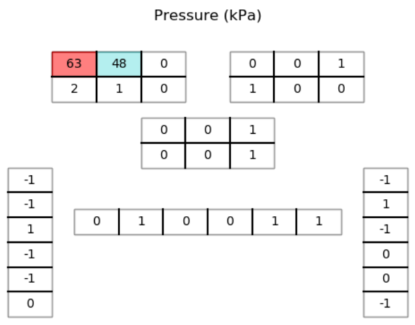


3.0	24.0	3.0
2.0	40.0	5.0

2.0	67.0	1.0
0.0	3.0	34.0

2.0	25.0	1.0
2.0	25.0	2.0

(a) : As created by previous students



63	48	0
2	1	0

0	0	1
1	0	0

0	0	1
0	0	1

-1
-1
1
-1
-1
0

0	1	0	0	1	1

-1
1
-1
0
0
-1

(b) : Our version

Figure 4.1: Visualization of tactile sensors

The indexing of our tactile sensors and their placement on the Python visualization corresponds to what the uses sees when facing the front side of 3FG (that features the logo and connection inputs/outputs). In Figures 4.2a, 4.2b and 4.2c we can observe the specific placement of sensors according to their official factory addresses. Figure 4.2d shows their position in our visualization.

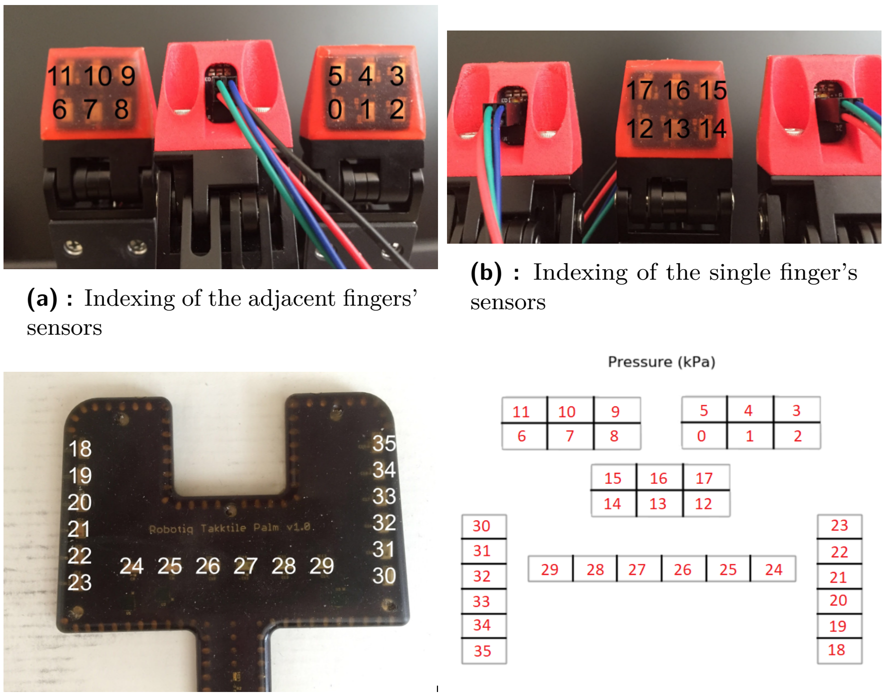
In the work done on the project by previous students, a simplified approach to calculating pressure measured by the Takktile sensors was chosen. The raw pressure conversion is done using Equation 4.1.

$$P_{comp} = a_0 + (b_1 + c_{12} \cdot T_{adc}) \cdot P_{adc} + b_2 \cdot T_{adc} \tag{4.1}$$

P_{adc} is the raw pressure ADC output, T_{adc} is the raw temperature ADC output, a_0 is the pressure offset coefficient, b_1 is the pressure sensitivity coefficient, b_2 is the temperature coefficient of offset (TCO), c_{12} is the temperature coefficient of sensitivity (TCS). To convert P_{comp} into pressure p in kPa, we use Equation 4.2.

$$p = P_{comp} \cdot (\frac{65}{1023}) + 50\ kPa \tag{4.2}$$

Previously, a mistake in the Arduino code made it impossible to read b_1 and b_2 coefficients correctly. Kristýna Míková and Petr Kuželа chose to use average values of the coefficients instead. However, we have found this error was caused by reading binary data as unsigned floats instead of signed floats. We fixed this, making it possible to read factory applied coefficients



(a) : Indexing of the adjacent fingers' sensors

(b) : Indexing of the single finger's sensors

(c) : Indexing of the palm's sensors

(d) : Indexing of our Python visualization

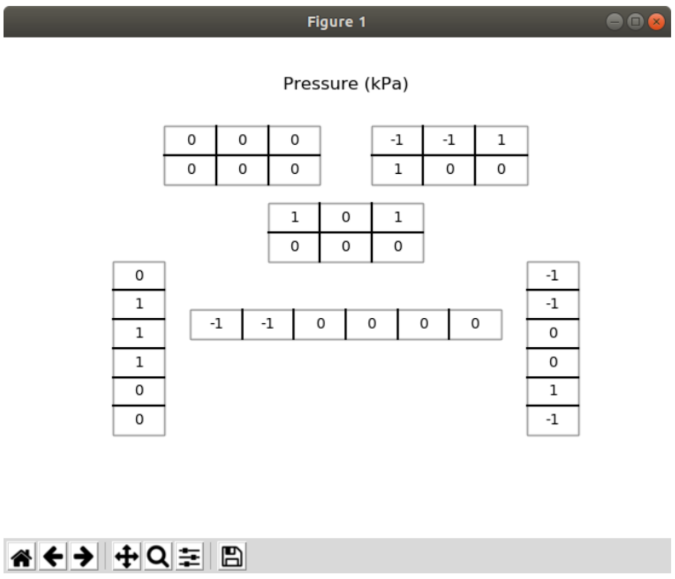
Figure 4.2: Tactile sensors indexing

for each sensor. This also improved the reading of pressure on the palm sensors. Previously, the palm sensor values oscillated between 0 - 7. Now, they oscillate between -1 and 2. Figure 4.3 shows the sensor status while the tactile sensors are not in contact with any object.

We also overcame a problem that came with 10bit numbers coming from the sensors and then overflowing with added pressure. The raw pressure ADC value starts at around 600 when at rest. By adding pressure, the value drops until it reaches 0 and then overflows to 1024. It can reach values around 950 with maximum pressure applied to the sensors. A newly added 'if' clause in our Arduino code ensures that when the raw pressure value is above 900, we subtract 1024 from it. This solution appears to work quite well. In the original visualization code we removed the computation of absolute value, letting the sensors work with negative pressure as well. The exact calculation process can be seen in Listing 4.1.

Listing 4.1: Arduino example code

```
void read_and_send() {
uint16_t otemp, opressure;
float pressureComp;
float *pointer = &pressureComp;
for (int i = 0; i < addressLength; i++) {
```



Pressure (kPa)

0	0	0
0	0	0

-1	-1	1
1	0	0

1	0	1
0	0	0

0
1
1
1
0
0

-1	-1	0	0	0	0

-1
-1
0
0
1
-1

Figure 4.3: Tactile sensors while at rest

```
    readData(addressArray[i], &otemp, &opressure);
    if((float)opressure > 900.0) {
        *pointer = a0[i]  + (b1[i] + c12[i] * (float)
           otemp) * ((float)opressure - 1024)+ b2[i] *
           (float)otemp;
    }
    else {
        *pointer = a0[i] + (b1[i] + c12[i] * (float)
           otemp) * (float)opressure + b2[i] * (float)
           otemp;
    }
    // Calculate temp & pressure
    float oPressure = ((65.0 / 1023.0) * pressureComp)
       + 50.05; // kPa
    float oTemp = ((float) otemp - 498.0F) / -5.35F +
       25.0F; // C
    Serial.print((float)oPressure,0);
    Serial.print(',');
}
Serial.println(addressLength);
delay(10);
}
```

We have tested the functionality of our new visualization, finding it to be reacting faster to outside stimulants and encountering fewer lags and crashes. Our new implementations thus seemed to work as intended.

4.2 3FG Control

The 3-Finger Adaptive Robot Gripper from Robotiq is fitted to be installed on a Franka Emika Panda arm [1]. Robotiq provides software for testing and control purposes. A gripper software called URCap is available for easy and basic control of the gripper and a robotic arm. Options for simple programming and setting up of tasks is also possible. The primary communication protocol is Modbus RTU. Modbus TCP, EtherNet, CANopen protocols and others are also available.

First, we have decided to test the communication between our 3FG and computer. We have downloaded the Robotiq User Interface (RUI) and installed it under Windows. The application can be downloaded from the official website of Robotiq. We connected to the gripper via USB for Modbus RTU. The application scanned for all available Robotiq devices and their IP addresses. After it had successfully detected our 3FG connection, we were able to connect to the gripper. The RUI provides simple control options, including switching gripper modes; setting a required position, speed and force; current measurement and individual finger control. The RUI also lets us change the 3FG IP address, if necessary.

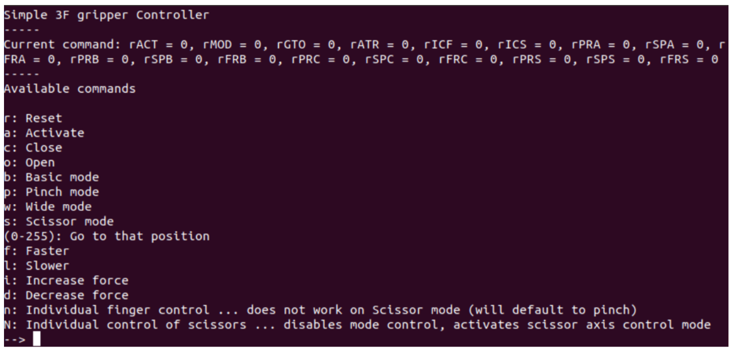
Next, we decided to control our 3FG over ROS 1. This decision was made before we decided to switch to ROS 2. We first downloaded a ROS-Industrial Robotiq meta-package from public repository. For our needs, we have only focused on the 3FG related packages. The most important ones are: `robotiq_3f_gripper_control`, `robotiq_modbus_tcp` and `robotiq_3f_gripper_articulated_msgs`.

First, we attempted to set our connection over the Modbus RTU protocol. We had, however, encountered an issue where our gripper did not respond to our commands beyond simply activating. Unable to find a solution, we switched to Modbus TCP communication protocol. This time, the connection was established successfully and we had full control over ROS for our 3FG. The `robotiq_modbus_tcp` package gives us the option to establish the connection. The `robotiq_3f_gripper_control` package defines three important ROS nodes: the Modbus TCP Communication Node, Simple Controller and Status Listener. The `robotiq_3f_gripper_articulated_msgs` package includes the description of messages used by ROS topics.

In order to control our 3FG over ROS, we first have to start a Modbus TCP communication node. It's necessary to know the IP address of the gripper. We let this node run in the background while we work with the 3FG.

An easy way to switch between modes is provided by a Simple Controller ROS node. The 3FG has four different modes of movement that can each be used to pick up objects of various size and shape. The 'Basic' and 'Wide' modes are generally used for manipulating bigger objects, while 'Pinch' and 'Scissor' modes allow higher precision for holding small objects. The node can change speed and force of the 3FG and specify the exact position of the fingers. All options can be seen in Figure 4.4a. The ROS node responds to simple keyboard commands.

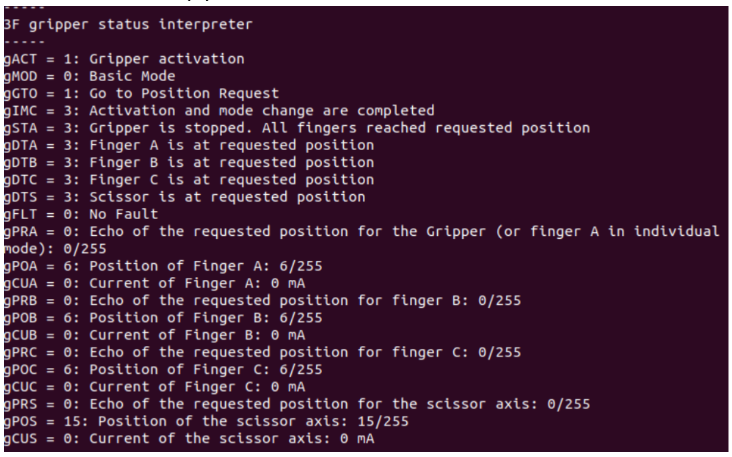
Status Listener ROS node for our 3FG lists the names and values of each register. More detailed explanation for each register is provided for simpler understanding. Status Listener node can be used to check the current status of the 3FG, its finger positions and more. Its visualization can be seen in Figure 4.4b.



```
Simple 3F gripper Controller
-----
Current command: rACT = 0, rMOD = 0, rGTO = 0, rATR = 0, rICF = 0, rICS = 0, rPRA = 0, rSPA = 0, r
FRA = 0, rPRB = 0, rSPB = 0, rFRB = 0, rPRC = 0, rSPC = 0, rFRC = 0, rPRS = 0, rSPS = 0, rFRS = 0
-----
Available commands

r: Reset
a: Activate
c: Close
o: Open
b: Basic mode
p: Pinch mode
w: Wide mode
s: Scissor mode
(0-255): Go to that position
f: Faster
l: Slower
i: Increase force
d: Decrease force
n: Individual finger control ... does not work on Scissor mode (will default to pinch)
N: Individual control of scissors ... disables mode control, activates scissor axis control mode
-->
```

(a) : Simple Controller ROS node



```
-----
3F gripper status interpreter
-----
gACT = 1: Gripper activation
gMOD = 0: Basic Mode
gGTO = 1: Go to Position Request
gIMC = 3: Activation and mode change are completed
gSTA = 3: Gripper is stopped. All fingers reached requested position
gDTA = 3: Finger A is at requested position
gDTB = 3: Finger B is at requested position
gDTC = 3: Finger C is at requested position
gDTS = 3: Scissor is at requested position
gFLT = 0: No Fault
gPRA = 0: Echo of the requested position for the Gripper (or finger A in individual
mode): 0/255
gPOA = 6: Position of Finger A: 6/255
gCUA = 0: Current of Finger A: 0 mA
gPRB = 0: Echo of the requested position for finger B: 0/255
gPOB = 6: Position of Finger B: 6/255
gCUB = 0: Current of Finger B: 0 mA
gPRC = 0: Echo of the requested position for finger C: 0/255
gPOC = 6: Position of Finger C: 6/255
gCUC = 0: Current of Finger C: 0 mA
gPRS = 0: Echo of the requested position for the scissor axis: 0/255
gPOS = 15: Position of the scissor axis: 15/255
gCUS = 0: Current of the scissor axis: 0 mA
```

(b) : Status Listener ROS node

Figure 4.4: Visualization of ROS nodes

We have become familiar with the provided Simple Controller and Status Listener ROS nodes. Using our knowledge of 3FG registers, we have expanded the Simple Controller node with additional functionalities, like individual finger control and individual control of scissors. While controlling each finger individually, we can set unique speed, force and position for every finger. The fingers follow the same path as in 'Basic' mode, meaning all three fingers close down towards the palm like a hand closing into a fist. In the 'Scissors' mode, the two adjacent fingers move towards each other while the opposite finger remains stationary. Thanks to the individual scissors' control, we can set the speed, force and position of the two fingers as they follow the 'Scissors' mode trajectory. We have created a guide detailing each option to ensure better understanding for future users [6].

Thanks to the RUI, we have established the 3FG was functioning correctly. We have encountered an issue, when trying to connect over the Modbus RTU protocol did not work until we used a network switch. Additionally, we had to use Modbus TCP protocol for ROS control of our gripper, as Modbus RTU did not function correctly. Our new control options for Simple Controller worked just as we intended.

At this point, we have decided to connect our Takktile sensor visualization and our 3FG ROS controller. We have taken our tactile sensor visualization Python code and implemented it as a new ROS node.

The new Sensor Listener ROS node functioned as expected. After a long period of operational time, however, the visualization seems to accumulate lag from real time response. In extreme cases, the visualization itself crashes. The node can simply be restarted to fix this issue.

4.3 Experiments with Our Peg

For our peg-in-hole task, we must implement the use of feedback from our tactile sensors. This will allow us to finish our assigned task using a contact model-based active compliance. A working real life environment for the peg-in-hole task must be created for this purpose. We must test the tactile sensor response with the peg in 3FG's grasp. The 3FG gripper used in our experiments is depicted in Figure 4.5.

We have created a wooden peg in a cuboid shape. Dimensions of the peg are as follows: 100 x 50 x 290 mm. Thanks to the combined efforts of Antonín Mík (a lab technician, who designed and 3D printed the hole) and Václav Hlaváč (thesis advisor, who manufactured the peg in his woodworking workshop at home), the 3D printed hole was fixed to a wooden board to increase stability. Our final setup can be seen in Figure 4.6. Antonín Mík

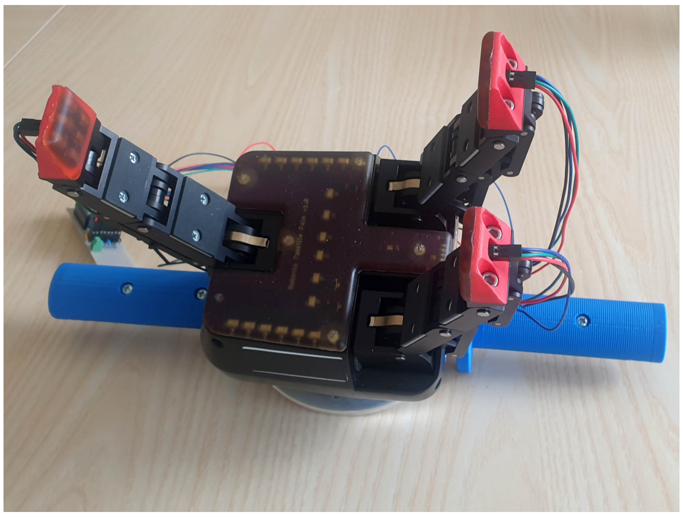


Figure 4.5: 3FG (the gripper) and tactile sensors used in our experiments.

also designed and 3D printed a two-sided handle that could be connected to the 3FG and used to manually simulate a robotic arm (Figure 4.5). It enables an experiment where a human acts instead of a robot in the peg-in-hole task. The outcome of the experiment is a time series of values, which the individual tactile sensors measure.

We experimented with our peg-in-hole, 3FG and tactile sensors. We were most interested in the way tactile sensors reacted to holding the peg as it was being inserted into the hole. We intentionally tested the sensors' reaction if the peg got stuck or was being inserted incorrectly. The 'Close' command of 3FG simple controller was used to catch the peg with our gripper. Later we have started using the option of 'Set position' to compare the results. We have moved the peg around and slowly inserted the peg in the hole. We were sure to test as many different positions as possible and document each case carefully for future use.

4.4 ROS 2 Simulation

While still in development, ROS 2 provides many desired upgrades from ROS. As such, we have decided at this point to move the rest of our work over to ROS 2. We have familiarized ourselves with new processes of ROS 2 and created a simulation of our peg-in-hole task in preparation for future real-life control. We will create a script of commands using the MoveGroup class that subscribes to the `move_group` node and controls predefined movement of the robot in rviz.

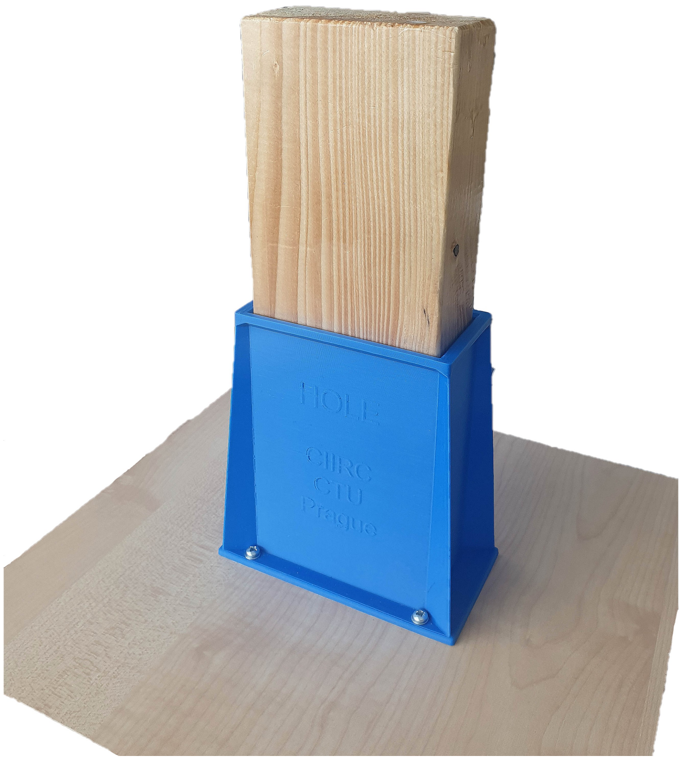


Figure 4.6: Our peg-in-hole setup

We have begun our work with assistance from Libor Wagner, a lab technician. Thanks to his previous experience with ROS 2, we were able to use ROS 2 inside an image in a Singularity container. The image includes ROS 2 distribution Foxy and MoveIt2. We have learned to use rviz visualisation tool for MoveIt2 before moving on to create our simulation.

Learning to use the MoveGroup class, we have decided to base our simulation on one of the provided tutorials [13]. The provided script was written in C++ and so we have chosen to write our simulation in this language as well. We begun with simply moving the robotic arm around a box with an attached cylindrical peg in its gripper. Both the box and the peg were created in the environment using Collision Object messages. However, we ran into an issue which we called 'ghost peg', for our peg kept moving (in other words phasing) through the box.

To fix the 'ghost peg' error, we decided to make our box part of the robot's description instead. The model of our robot is represented in the Unified Robotic Description Format (URDF) file. Learning to edit and write in the XML file format, we managed to modify the robot's URDF to include our box. Using the combination of URDF box and Collision Object message to create our peg, we finally managed to solve our 'ghost peg'. We changed the peg to a cuboid shape and set its dimensions and the box's to be a perfect copy of those in real life.

We wrote our simulation into a set of simple commands. Single steps can be seen in Figure 4.7. Their order is as follows:

1. Move the robot from its starting position (4.7a) to the right side of the gripper (4.7b) by defining a pose goal and executing the movement.
2. Add the peg using Collision Object message and attach it to the end-effector (the gripper).
3. Move the robot from the right side of the box to its left side (4.7c), to demonstrate how the peg avoids collision with the outline of the box.
4. Move the robot so that the peg stops above the hole (4.7d).
5. Switching to Cartesian path planning, we move the peg (and the robot) down along the z-axis and insert it into the hole (4.7e).

During the final stages of programming, we have found a possible reason for our 'ghost peg' error. After setting a pose goal, there are two options of reaching the goal. The first option is to call the Plan function first creating the Plan object (which describes our found trajectory); the second step is to subsequently select the Execute function separately, which results in moving the robot according to the Plan object. The second option combines the two functions in a single function called Move. In the first case, we encounter the 'ghost peg'. However, using the Move function, the 'ghost peg' error completely disappeared. It would thus seem the 'ghost peg' error originates in the Plan function and the Plan object (which stores the plan function results). This seems to be an innate error with ROS 2 code and is possibly to be solved in future distributions.

We attempted to replace the original end-effector gripper in Franka Emika Panda robot URDF with our 3FG. We used meshes provided by the Robotiq GitHub repository to set our new URDF gripper. Visually we were successful, but it does not function correctly during motion planning yet. The Robotiq meta-package provides detailed description of gripper's kinematics, sadly it is only set for ROS 1. We have tried to port this into ROS 2, but were ultimately unsuccessful.

We also tried to make a simulation using Python code. We have learned to create basic packages sending simple messages. Sadly, when we tried to

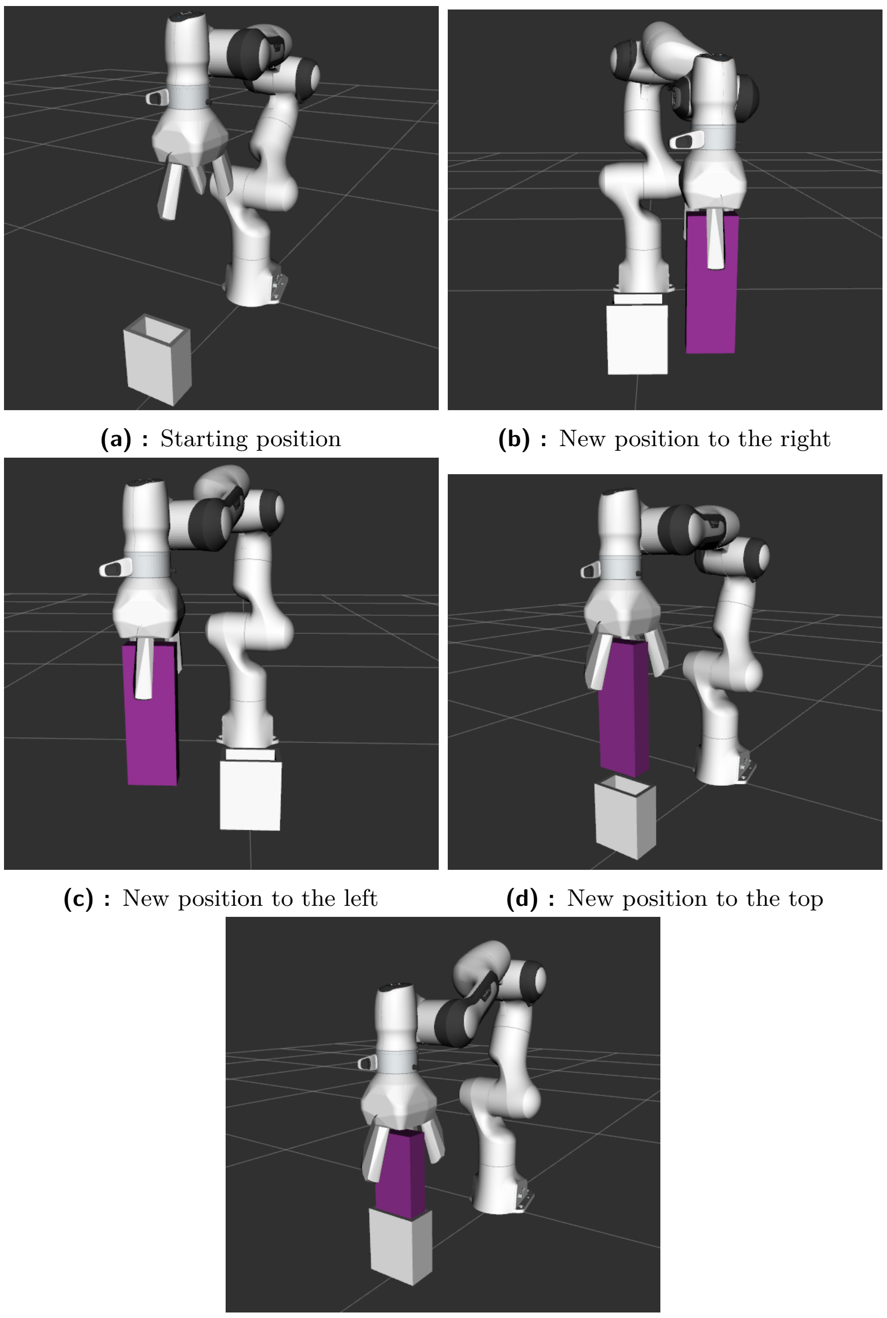

(a) : Starting position **(b) :** New position to the right **(c) :** New position to the left **(d) :** New position to the top **(e) :** Final position

Figure 4.7: Our simulation

recreate our simulation in Python, we ran into an issue. In ROS 1, the Move Group Python Interface uses a moveit_commander namespace, where we can find all necessary class description for moving our robot. Unfortunately, the moveit_commander namespace has not been ported into ROS 2 yet. This made us give up our attempts at recreating our simulation in Python under ROS 2.

Chapter 5

Evaluation

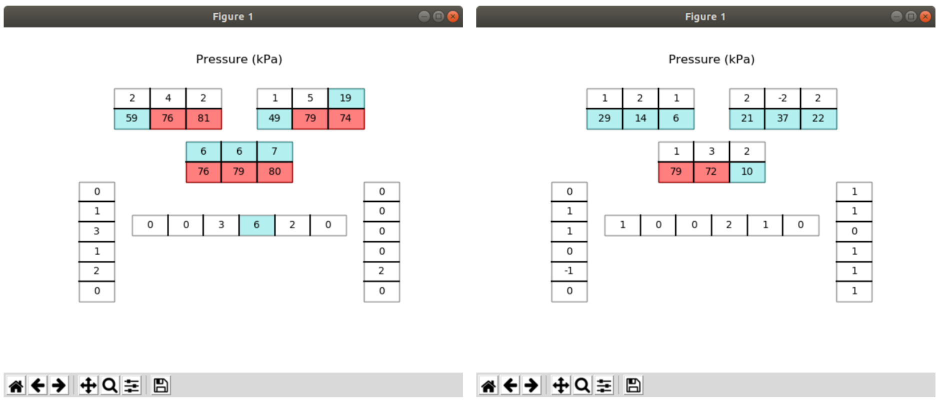
In this chapter, we will go over each of our method implementations and evaluate their results.

We have successfully managed to implement a visualization for our Takktile sensors and fix its previous issues. Our new version measures pressure more precisely and steadily. Sadly, we were not able to fix the issue, which causes the visualization to crash after running for an extended time. While we have done our best to fix connectivity issues with our Arduino board, sometimes the sensors do not all load correctly. Mostly, this results in our visualization showing the last palm sensor (address 35) only as 0 kPa, no matter the pressure we apply to it. The other sensors then tend to be moved 'in a row' of their set addressing (as seen in Figure 4.2). This is most likely due to problems with the I^2C protocol and the connection to our Arduino board that causes a sensor data to be 'lost' during communication.

For our 3FG our ROS control nodes work just as we intended. We had to switch from Modbus RTU communication protocol to Modbus TCP during the design stage of the project. For now, Simple Controller node only responds to a set of very basic keyboard commands. Our new node for tactile sensors visualization faces the same issues as mentioned for its Python code version, but otherwise functions as intended.

During our experiments with holding the peg in our gripper and inserting it in a hole, we have discovered the best mode for holding our peg was the 'Basic' mode. Sensors responded to the fingers closing around our peg as expected. However, while inserting the peg into the hole, the reaction of sensors to the various different positions were similar even in very different use-cases and sometimes the tactile sensors did not even register a change in pressure detection. We tried to decrease the force with which the gripper closed around the peg, but noticed no visible difference in sensors' reactions under such change. We have found that if we open the gripper slightly after calling the 'close' command, the sensors react much better. Sadly, there does not seem to be one 'optimal' position the 3FG can be set to and hold the peg securely with best sensor results. Position 77 or position 78 seemed to be closest to our desired goal, but these positions change depending on each activation and calibration of the gripper and are not necessarily always the same.

We can compare the difference on a set of given figures. In Figure 5.1 we have a situation in which our gripper holds our peg securely and no other forces are being applied to the construct. Figure 5.2 reflects the changes in the sensors that occurred once we manually applied pressure on the peg from the back side of the 3FG. In the case of Figure 5.2b, we can read the situation more easily than in Figure 5.2a.

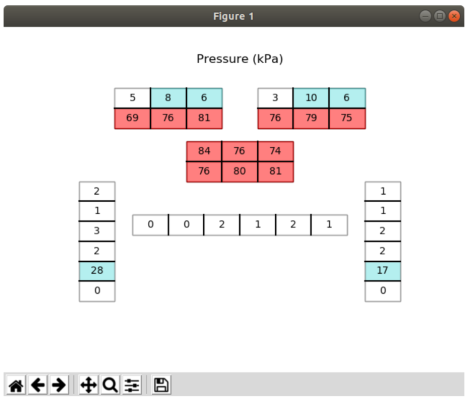


(a) : Using CLOSE command **(b) :** Using SET POSITION command

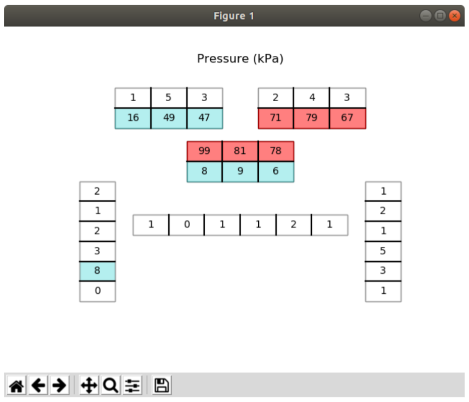
Figure 5.1: Visualisation of tactile sensors while holding a peg with no external influence

We experimented to find out the best way to hold the peg. The optimal way we found was to call the 'Close' command, use the gripper Status Listener node to check the actual position of fingers and then use the Simple Controller to move to a new position with a subtraction of 1 unit. In this case, the tactile sensors provided the best reading and sensed the most positions and issues during insertion of the peg into the hole. However, this method is not very favorable for our future peg-in-hole task.

Our simulation of the peg-in-hole task works as expected. Sometimes, the inverse kinematics task does not correctly calculate. In that case, the robot simply stays in its old location without moving to the new set goal state. Our



(a) : Using CLOSE command



(b) : Using SET POSITION command

Figure 5.2: Visualisation of tactile sensors while holding a peg and pressing on the back of its face

attempts at translating our simulation from C++ into Python ended in a failure due to the missing ROS 2 packages. Our simulation also includes the 3FG (gripper) mesh as the end-effector. We fully implemented the 3FG in our robot description file, but did not manage to write an inverse kinematics task for it. This means we can see our 3FG during rviz planning visualization and check that our joints can correctly move from an 'open' state into a 'close' position. However, when we attempt to call for gripper position change using a 'Plan and Execute' option of rviz, the task fails due to the right kinematics descriptor missing.

Chapter 6

Conclusion

6.1 Brief Summary of the Thesis' Achievements

As part of our work, we had to familiarize ourselves with the 3FG and its control. We used ROS nodes to send commands to the gripper to active it, change its modes, set position and speed of its fingers and catch objects. Using Takktile sensors mounted on 3FG fingers and its palm, we helped improve a simple visualization of pressure recognition. We did this by upgrading the Python code and recreating it as a new ROS node. We also re-configured the Arduino code for the tactile sensors for more precise pressure calculation.

We had created a wooden peg (100×50×290 mm in dimensions) and printed out our 'hole' using 3D printing technology. First, we had tested the reaction of tactile sensors when in contact with the peg and experimented with the 3FG holding the peg under various force settings. Then we inserted the peg into the hole and documented the pressure sensors' response. We have deduced that relaxing the hold of our gripper on the peg leads to better sensor response and is more suitable for future pressure feedback manipulation.

Deciding to implement our experiment in the new ROS 2 environment, we had first begun with creating a simulation for our peg-in-hole manipulation task. Our simulation was using the Franka Emika Panda robot. The simulation was created thanks to a provided 'Run Move Group' tutorial and programmed in C++. We also learned to modify 'URDF' files to suit our requirements and added our hole and gripper into the scene. We attempted to create a second simulation in Python code, but missing packages for ROS 2 made it impossible to finish.

We did not manage to create a real-life experiment of our peg-in-hole task. This was mostly due to the restrictions placed on our work by the COVID-19 pandemic and the fact that we were missing vital parts of necessary software from ROS 2 to transform our simulation onto a real robot. By June 2021, a new distribution of ROS 2 will contain much needed packages to allow us to continue working on this project and finally move to real-life implementation.

6.2 Contributions

Our first contribution is the analysis of the peg-in-hole assembly task assuming only tactile and force/torque feedback is available. The thesis summarizes the state of the art and formulates openings for further investigation.

Our second contribution covers methods and software tools enabling use of 3FG (the gripper) and the tactile sensors attached to it. The related software, mostly developed in this bachelor thesis project, is available in two separate CIIRC GitLab repositories.

Everything necessary for real-life 3FG control is available in the repository [6]. This includes: modified ROS 1 nodes for 3FG control; Takktile sensors' visualisation Python code; and the new code for our Arduino board. We have also added a folder for our documentation and our guide for 3FG usage and control of nodes. We have also uploaded a more detailed research for our peg-in-hole experiment with our tactile sensors.

The second repository includes a simulation of our peg-in-hole task under ROS 2 [7]. The repository was created with an easy set-up and use in mind and with a simple guide provided by the README.md file. The repository holds our C++ code for our simulation and necessary launch files for rviz and its nodes. All URDF and mesh files were added and configured for installation in the CMakeLists.txt for our unique description of robot and its environment.

6.3 Suggestion for Future Improvements

The work completed and assembled while working on this project has many possibilities for further development. The project can be further expanded by transforming our simulation onto a real-life Franka Emika robot. The missing ROS 2 packages should be added this summer, making it possible for us to continue working on this project. Ideally, we would also port our ROS 1 control nodes for our 3FG into ROS 2, either by writing new ROS 2 dedicated nodes with same functionalities or by using a bridge between ROS 1 and ROS 2.

Bibliography

[1] 3-finger adaptive robot gripper official website. `https://robotiq.com/products/3-finger-adaptive-robot-gripper`. Accessed: 2020-04-27.

[2] Fares Abu-Dakka, Bojan Nemec, Aljaz Kramberger, Anders Buch, Norbert Krüger, and Ales Ude. Solving peg-in-hole tasks by human demonstration and exception strategies. *Industrial Robot: An International Journal*, 41:575–584, Oct 2014.

[3] Cristian Camilo Beltran-Hernandez, Damien Petit, Ixchel Georgina Ramirez-Alpizar, and Kensuke Harada. Variable compliance control for robotic peg-in-hole assembly: A deep reinforcement learning approach. *CoRR*, abs/2008.10224, 2020.

[4] Tadanobu Inoue, Giovanni De Magistris, Asim Munawar, Tsuyoshi Yokoya, and Ryuki Tachibana. Deep reinforcement learning for high precision assembly tasks, 2017.

[5] Ibrahim F. Jasim, Peter W. Plapper, and Holger Voos. Position identification in force-guided robotic peg-in-hole assembly tasks. *Procedia CIRP*, 23:217–222, 2014. 5th CATS 2014 - CIRP Conference on Assembly Technologies and Systems.

[6] Lucie Vajnerová (Maintainer). Robotiq 3f gripper control. `https://gitlab.ciirc.cvut.cz/vajneluc/robotiq_3f_gripper_control/-/tree/master`, May 2021.

[7] Lucie Vajnerová (Maintainer). Run move group. `https://gitlab.ciirc.cvut.cz/vajneluc/run_move_group`, May 2021.

[8] David Martínez, Guillem Alenya, and Carme Torras. Relational reinforcement learning with guided demonstrations. *Artificial Intelligence*, 247:295–312, 2017. Special Issue on AI and Robotics.

[9] Yuya Maruyama, Shinpei Kato, and Takuya Azumi. Exploring the performance of ros2. In *Proceedings of the 13th International Conference on Embedded Software*, EMSOFT '16, New York, NY, USA, 2016. Association for Computing Machinery.

[10] Using takktile sensors with robotiq three finger adaptive gripper. `http://people.ciirc.cvut.cz/~hlavac/pub/MiscText/2020-08-04KuzelaMikovaUsingTakkTileSensorsRobotiqGripper.pdf`. Accessed: 2021-05-13.

[11] Official moveit website. `https://moveit.ros.org/`. Accessed: 2021-05-03.

[12] Ati industrial automation: Rcc remote center compensators. `https://www.ati-ia.com/products/compliance/compensator_product_desc.aspx`. Accessed: 2021-04-27.

[13] Move group c++ interface – moveit2_tutorials foxy documentation. `http://moveit2_tutorials.picknik.ai/doc/move_group_interface/move_group_interface_tutorial.html`. Accessed: 2021-04-27.

[14] Ros and ros 2 releases. `https://docs.ros.org/en/foxy/Releases.html#releases`. Accessed: 2021-04-27.

[15] Changes between ros 1 and ros 2. `http://design.ros2.org/articles/changes.html`. Accessed: 2021-04-27.

[16] Hands-on introduction to robot operating system (ros). `https://trojrobert.github.io/hands-on-introdution-to-robot-operating-system(ros)/`. Accessed: 2021-05-15.

[17] Moveit roadmap. `https://moveit.ros.org/documentation/contributing/roadmap/`. Accessed: 2021-05-03.

[18] Official ros website. `https://www.ros.org/`. Accessed: 2021-05-03.

[19] A. S. Sadun, J. Jalani, and F. Jamil. Grasping analysis for a 3-finger adaptive robot gripper. *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA)*, pages 1–6, 2016.

[20] Paul C. Watson. Remote center compliance system, U.S. Patent 4098001A, filed October 13, 1976, issued July 4, 1978.

[21] D. E. Whitney. Quasi-Static Assembly of Compliantly Supported Rigid Parts. *Journal of Dynamic Systems, Measurement, and Control*, 104(1):65–77, 03 1982.

[22] Josef Wolfartsberger, Jean Hallewell Haslwanter, and René Lindorfer. Perspectives on assistive systems for manual assembly tasks in industry. *Technologies*, 7:12, 01 2019.

[23] P. C. Wong. *Peg-hole assembly; an investigation into tactile methods*. PhD thesis, University of Canterbury, 1975.

[24] J. Xu, Zhimin Hou, Zhi Liu, and H. Qiao. Compare contact model-based control and contact model-free learning: A survey of robotic peg-in-hole assembly strategies. *ArXiv e-prints*, abs/1904.05240:arXiv:1904.05240, April 2019.