

Bachelor Thesis



**Czech
Technical
University
in Prague**

F3

**Faculty of Electrical Engineering
Department of Cybernetics**

Exploratory Action Selection to Learn Object Properties from Haptic Exploration Using a Robot Hand

Michal Mareš

Supervisor: Mgr. Matěj Hoffmann, Ph.D.

Supervisor–specialist: Mgr. Karla Štěpánová, Ph.D.

Field of study: Cybernetics and Robotics

May 2020

Acknowledgements

I wish to express my deepest gratitude to Matěj Hoffmann, who supervised my thesis, and Karla Štěpánová for their valuable advice, encouragement, patient guidance and answering my questions. Both of them were always willing to put aside some time to help me despite their free time being scarce, especially Karla's.

Furthermore, I would like to recognize the invaluable assistance of Teymur Azayev, Petr Švarný, Zdeněk Straka and Hynek Chlup. Each of them was willing to give me an insight into specific topics included in my work.

Last but not least, I wish to thank my family, girlfriend and friends for tolerating my bad moods, lack of time and their unshakable support throughout my studies and particularly in the last few months.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 22, 2020

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 22. května 2020

Abstract

Although two objects can appear the same to the naked eye, they can have entirely different properties. One way to extract these properties is by using haptic exploration. In this work, I focused on soft materials and objects differing primarily in their elasticity/stiffness. I studied whether a multi-fingered hand with tactile and force feedback can correctly classify such objects and infer their elasticity/stiffness and density. I also investigated which finger configuration and speed of object squeezing is best suited for the task.

The work setup consisted of a three-finger robotic gripper Barrett Hand, twenty polyurethane foams and nine objects. The gripper was equipped with tactile matrices — both on the palm and each fingertip, and a joint torque sensor for each fingertip. An action primitive was defined as a tuple of finger configuration number and joint speed. A ROS package for the automation of the measuring process was implemented. Over 1000 measurements were collected, visualized and active taxel detection was employed. An LSTM neural network was created to classify the data and measurements were divided into datasets for the training of extracting specific object properties.

In most cases, the network could correctly classify the specific objects or place a foam into the correct elasticity/stiffness/density interval based on its parameters. In an ablation study, when training on just the tactile or fingertip torque data, I show that tactile sensors are more important for correct classification. Furthermore, the LSTM network trained on measurements taken at lower speeds could generalize and be utilized with similar accuracy at higher speeds. In some cases, the preference of action for a specific purpose was established, such as lateral finger

configuration for density interval classification or opposite finger configuration for any object with the smallest dimension larger than 70 mm.

Keywords: robotic grippers, tactile sensors, model-free object classification, object parameter extraction, LSTM network

Supervisor: Mgr. Matěj Hoffmann, Ph.D.

Abstrakt

Přestože dva objekty mohou na pohled vypadat stejně, mohou mít zcela odlišné vlastnosti a nezbyvá, než je zjistit použitím hmatu. V této práci jsem se zaměřil na měkké materiály a objekty s různou mírou pružnosti/tuhosti a zkoumal, zda může tříprstá robotická ruka se zpětnou vazbou z taktilních senzorů a senzorů točivého momentu tyto objekty správně klasifikovat a zjistit jejich vlastnosti. Dále jsem se pokoušel zjistit, které konfigurace prstů a rychlosti svírání jsou pro tyto úlohy nejvhodnější.

Kromě tříprstého robotického chapadla Barrett Hand jsem měl k dispozici dvacet polyuretanových pěn a devět předmětů. Chapadlo bylo osazeno čtyřmi taktilními ploškami – jedno na dlani a pak vždy na posledním článku prstu, a senzorem točivého momentu v posledním kloubu každého prstu. Akční primitivum jsem definoval jako uspořádanou dvojici číselného označení konfigurace prstů a kloubové rychlosti. Implementoval jsem balíček zjednodušující měření objektů do prostředí ROS. Celkem jsem provedl přes 1000 měření, která jsem následně vizualizoval. Vytvořil jsem způsob pro detekci aktivních taxelů. Pomocí LSTM neuronové sítě jsem klasifikoval předměty nebo zjišťoval konkrétní vlastnosti z měření rozdělených do datasetů.

Vě většině případů dokázala neuronová síť správně klasifikovat jednotlivé předměty nebo je na základě jejich vlastností správně zařadit do intervalů hodnot těchto vlastností. Zkoumal jsem i případ, kdy byly při trénování zcela vynechány buď taktilní data nebo data ze senzorů točivého momentu a zjistil jsem, že taktilní data jsou pro správnou klasifikaci důležitější. Dalším zjištěním bylo, že neuronová síť, která byla učena na měřeních pořízených při nižší rychlosti, dosahovala

podobných výsledků i pro měření pořízených při vyšších rychlostech. Pro některé úlohy jsem našel ideální akce, např. pro zjištění intervalu hustoty předmětu jsou lepší sdružené prsty, naopak k uchopení předmětů s délkou nejkratší hrany více než 70 mm je nutné použít protichůdné uspořádání prstů.

Klíčová slova: robotická chapadla, taktilní senzory, klasifikace objektů bez modelu, zjišťování vlastností předmětu, LSTM síť

Překlad názvu: Výběr průzkumných akcí pro robotickou ruku za účelem zjištění vlastností předmětů

Contents

1 Introduction	1	4 Results	27
1.1 Motivation	1	4.1 Data comparison	27
1.2 Goals	2	4.2 Object classification	30
1.3 Related work	2	4.2.1 Classification on foam set	30
1.3.1 Terrain classification	2	4.2.2 Individual actions on object set	31
1.3.2 Object manipulation divided to primitives	3	4.2.3 Knowledge transfer	31
1.3.3 Haptic sensing	3	4.3 Classification into discrete intervals	35
1.3.4 Extracting properties	3	4.3.1 Elasticity	35
1.3.5 Thesis contribution	4	4.3.2 Density	36
1.4 Outline	4	4.4 Elasticity regression	37
2 Materials and methods	5	4.5 Pressure reference calculation	39
2.1 Barrett Hand	5	4.6 Action selection	40
2.1.1 Joint states and torque sensors	6	5 Conclusion, discussion and future work	41
2.1.2 Control modes	7	5.1 Conclusion	41
2.1.3 State	7	5.2 Discussion and future work	42
2.1.4 Tactile sensors	7	Bibliography	45
2.1.5 Force/torque sensor at the base	8	Project Specification	49
2.1.6 Forward kinematics	8		
2.2 Action primitives	11		
2.3 Objects and materials	12		
2.4 Soft object classification and stiffness/density regression	15		
2.4.1 Long short-term memory neural network	15		
2.4.2 Problem formulation	16		
3 Data collection	19		
3.1 ROS package	19		
3.2 Visualization	20		
3.2.1 Data preprocessing	20		
3.2.2 Object compression	21		
3.2.3 Overview of used plots	23		
3.3 Datasets	25		

Chapter 1

Introduction

1.1 Motivation

In the past decade, machine vision experienced major improvements thanks to deep learning and growing interest in autonomous vehicles. However, when confronted with a problem of grasping an object with a robotic gripper, which is common in areas such as manufacturing or assistive technologies, it sometimes might not be capable of delivering satisfactory results. Two objects, despite looking the same, can have completely different properties. Furthermore, if the object is deformable and changes its shape after being grasped based on visual information, it may not be fixated and fall out or the application of too strong forces may damage it. To deal with issues similar to the ones described, another method must be used to extract the missing properties.

Haptic exploration is one of the possible methods. Inspiration can be seen in the human hand, which is “equipped” with mechanoreceptors for pressure, temperature or pain sensing. In robotics, tactile and torque sensors are commonly employed to receive similar signals. By combining the a priori information from vision and other sources about the object’s shape, size. . . and processing the haptic signals from following exploratory actions, we can obtain a comprehensive picture of the object. This allows us to, for example, select whether grip applying kinematic constrain or force grip is more suitable for the object and manipulate it without causing irreversible damage.

This work is part of a collaborative project Interactive Perception-Action-Learning for Modelling Objects (IPALM) between five academic institutions in Europe. The project’s purpose is to “develop methods for the automatic digitization of objects and their physical properties by exploratory manipulations” [1].

1.2 Goals

The goal of this project is to use a robotic gripper Barrett Hand to collect data from tactile and joint torque sensors and use them to determine parameters of grasped objects. A repertoire of actions will be introduced and collected data will be visualized. Measurements will be visualized and organized into datasets to enable the training of an LSTM neural network, which will then be used to determine the parameters. Finally, actions will be organized in a way that enables the selection of an individual action to extract specific object property.

The path taken in this work presents a model-free approach. I am going to focus on triparametric solid objects (i.e., none of the dimensions of the object can be neglected) and their elastic deformation.

1.3 Related work

The most relevant research for this thesis comes from the classification of materials and their properties. I am interested primarily in utilizing haptic sensing in robotic grippers, but similarities can be found among articles about terrain classification by legged robots as well. For a recent overview of the state of the art methods used in robotic manipulation and sensing, see [2].

1.3.1 Terrain classification

Recently, [3] researched the classification of materials with the use of a neural network. A 6-legged robot with accelerometers, pressure sensors, 3D force and 6D force/torque sensors collected data during regular operation on various terrain types. In contrast, a 3-axis optical force sensor was used for material classification alone. In both cases, raw signals without any additional processing were passed into a recurrent neural network architecture able to reduce the length of the data by passing values of overlapping moving windows into a convolutional network into a long short-term memory (LSTM) network with the addition of the prior factor estimated before each iteration. An accuracy of 97.96% was reached for the terrain classification and 100% accuracy for material recognition.

Something similar was presented in [4]. This time, a quadruped robot equipped with joint encoders, inertial and foot pressure sensors, gathered measurements on different surfaces, which were then classified. In addition to coordinated gaits, a random one was also studied. However, its accuracy was significantly lower than that of the coordinated movements.

■ 1.3.2 Object manipulation divided to primitives

Manipulation primitives, their abstraction and execution on different robotic setups were the main focus of [5]. First, an action is represented by a final state machine (FSM), where the states are manipulation primitives. The transitions between the states are triggered by abstract events, such as reaching a stable grasp. The events are detected by the available sensor specific to each setup. A mechanism to translate it to an embodiment specific FSM is proposed and the whole process is successfully tested on two different setups.

■ 1.3.3 Haptic sensing

In [6], model-free classification is carried on with a single grasp on a compliant gripper with force sensors. Features are collected at two moments during the grasping process: first when the gripper first touches the object; second, when the actuators stop moving. Furthermore, they are able to achieve similar accuracy using only 2 out of 8 force sensors on each of the fingers and thus simplifying the hardware in the future.

A human-like robot hand with artificial muscles and soft skin with tactile receptors was used in [7]. The grasping process had three stages. First, the object was handed to the gripper and after a fixed time grasp was pronounced stabilized. Then the grip could be adjusted two times, each time with different two fingers. With shapes cylinder, prism and ball considered classes, a Jordan-type recurrent neural network with context nodes classified each object. After 350 000 learning steps, cylinders reached the lowest accuracy of 80% on never before seen evaluation objects. Furthermore, different classes could be reliably recognized at different times, even during the grasping process.

A robotic manipulator Cody with a forearm covered with force-sensitive skin was used to conduct experiments in [8]. Data were sampled at a fixed speed, then truncated to begin in the moment of contact between the forearm and the object. Next, low dimensional representation was computed using principal component analysis (PCA). Finally, the k-nearest neighbor classifier (k-NN) was used to distinguish between fixed and movable objects, then rigid-fixed, rigid-movable, soft-fixed, soft-movable and finally the specific objects. The article also investigates the effects of specific features, their scaling or the time window length on the performance.

■ 1.3.4 Extracting properties

A reinforcement learning algorithm was used in [9] to learn an arbitrary object's center of mass in simulation. Their learned locations were used to stack the objects on top of each other and in different orders. The stacking was successful both in simulation and with a real robot and 3D printed objects. What yet remains is to test the algorithm directly with a real setup.

1.3.5 Thesis contribution

The most relevant work is [6], where model-free classification using force sensors is described. The key differences are that the robotic gripper I use has more possible configurations and a richer sensor repertoire. This gives my setup the ability to use different action for each task, which can lead to better accuracy.

Bednarek et al. [3] propose a way to preprocess the input time-series and then use an LSTM neural network for the classification of the terrains and materials. This is one of the few articles where elastic and deformable objects are also considered. In my case, the objects will be exclusively elastic ones and the LSTM neural network will be used to classify them.

A bionic hand from [7] equipped with tactile receptors has multiple motors and multiple DOF, similar to the Barrett Hand. However, I will not be using dynamic interaction or readjusting the grip in any way.

With the exception of [9], where finding the object's center of mass is the primary goal, none of the articles I encountered tried to determine the exact properties of the materials, but rather classified materials or objects. This is one of the main contributions I am going to bring to the field, as I will try to extract specific values of elasticity/stiffness and density.

1.4 Outline

First, I am going to present my experimental setup, define actions, give an overview of grasped objects and introduce the neural network used for data processing (Chapter 2).

In Chapter 3, the process of grasping the items and measuring of the data will be described. Their visualization and division into datasets for training the neural network will also be presented.

Next, the obtained results will be shown and analyzed in Chapter 4. The summary of preferred actions for extracting specific parameters of an object will be given.

Finally, in Chapter 5, I am going to discuss the results, identify the limitations and suggest possible improvements that could be made in the future.

Chapter 2

Materials and methods

All the code used in this thesis is available at this online repository [10]. Specifically, my work is in the `BarrettHand/` folder.

2.1 Barrett Hand

Robotic gripper used for all of the experiments is Barrett Hand (model BH8-282) from Barrett Technologies, supplied to us by Robotnik (<https://www.robotnik.eu/>). It has three fingers, of which two can rotate around the base. This design allows numerous configurations of the three fingers, as described in Section 2.2, with the only restriction that the spread joints of Finger 1 and 2 are mimicking each other.



Figure 2.1: Barrett Hand.

Numerous sensors are available on the gripper. There are matrices of tactile sensors on the palm and each finger, a torque sensor in each fingertip and a 6-axis torque/force sensor in the base (not used in this work).

It is completely self-contained, meaning all the hardware necessary for

its operation is enclosed in the Barrett Hand itself. The low-level messages communicate via the CAN bus protocol. To control it, I will be using Robot Operating System (ROS), specifically packages supplied to us by Robotnik `barrett_hand` [11], `bhand_controller` [12] and `rqt_bhand` [13].

2.1.1 Joint states and torque sensors

There are eight joints in total on the Barrett Hand. They can be seen in Figure 2.2; those marked red are controllable while the white ones are not. I am going to refer to the `bh_j11_joint` and `bh_j21_joint` as F1-Spread and F2-Spread joints; then, first joint in each finger chain as F_i -Base joint and the last one as F_i -Tip joint, where i stands for finger number. The joint coordinates are published to the `/joint_states` ROS topic.

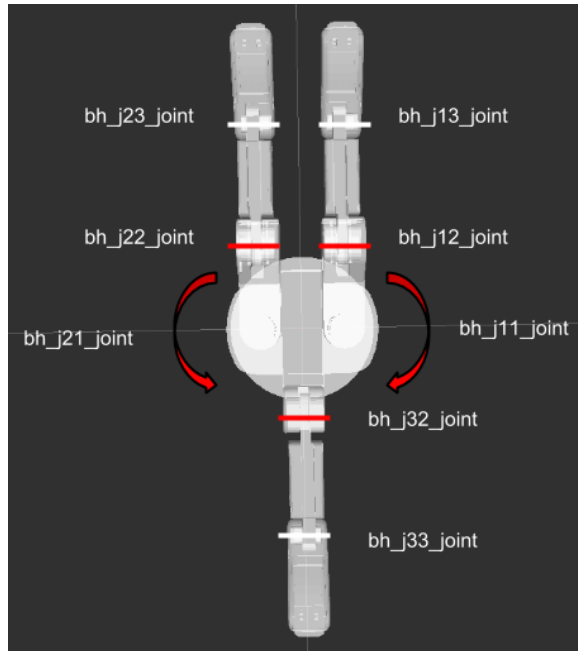


Figure 2.2: Joint types, taken from the official documentation [12].

Each of the fingers has only one motor between the palm and the base link, which drives both joints in the finger – the mechanism is called TorqueSwitch. In addition to having only one motor, which results in much lighter fingertips, a complex mechanism also determines the appropriate where to direct the torque. For a full description, see [14].

In reality, we can observe the mechanism in the following situation. When the base finger link can not move any further (for example, it came into contact with a rigid object), the torque of the motor transforms to the fingertip link. It encloses the object securing Barrett Hand’s grip.

In addition to the tactile sensors, the hand possesses fingertip joint torque sensors. These readings are published in the effort field of the `/joint_states`

topic in N/m and, unlike tactile sensors, refresh the value at the set 200 Hz frequency, see Section 2.1.4. Although the array published to the topic gives an idea of torque sensors being present in each joint, in reality they are not. Values for spread joints are always zero, and each finger's base joints only mimic values from the fingertip.

■ 2.1.2 Control modes

The package `bhand_controller` [12] has two control modes, position and velocity. The user can switch from one to the other by calling a ROS service `/bhand_node/set_control_mode/` and specify the desired mode.

In position control, the user specifies the joint coordinates of each of the actuated joints in radians, that is F3-Base, F1-Spread, F1-Base, F2-Spread and F2-Base (F1-Spread and F2-Spread set the same motor as mentioned before) in a predefined message and publishes it to `/bhand_node/command` topic. The gripper then uses its PID regulator to get to the desired position.

The second mode is velocity control. The process is similar to the position control only the values passed in the message represent speed in rad/s for each actuated joint this time around.

During the experiments, I will be using velocity control because it enables fingers to move at a constant speed, which is beneficial for my goal.

■ 2.1.3 State

Topic `/bhand_node/state` publishes information regarding the Barrett Hand operations. It includes state value and its description, from which the user can read, for example, failure states. Furthermore, it includes current control mode, boolean whether the hand is initialized or not, and the temperature of motors.

■ 2.1.4 Tactile sensors

What makes Barrett Hand unique are tactile sensors located underneath the blue plastic covers. Each of the tactile pads includes 24 capacitive cells of various surface areas, as seen in figure 2.3. Values from sensors are computed directly on the hardware and are then published in an array on ROS topic `/bhand_node/tactile_array` in N/cm^2 .

Although the topic publishes at 200 Hz, values are always constant for some time. From my calculations, the value changes approximately every eight messages, which means the effective frequency is around 25 Hz. This information might be useful to take into account later on.

Fingertip Torque Sensors P/N: B0106		BarrettHand with Tactile Sensors P/N: B4335	
Function	Senses torques about last joint in each finger	Function	Localizes pressure across palm and fingers
Quantity	3 (1 per finger)	Quantity	96 active cells
Element Type	Metal foil strain gage	Element Type	24 capacitive cells per sensor pad
Range	+/- 1 N-m	Range	10 N/cm ²
Resolution	0.04 N-m	Resolution	Palm: 0.02 N/cell; cell area 1.0 cm ² Finger: 0.01 N/cell; cell area 0.3 cm ² Fingertip: 0.01 N/cell; cell area 0.15 cm ²

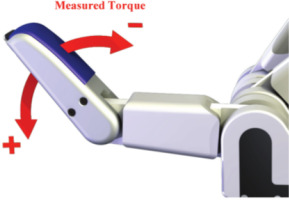




Figure 2.3: Sensor specification, taken from the official documentation [12].

ROS topic	Frequency [Hz]
/bhand_node/force_torque	200
/bhand_node/tact_array	200
/bhand_node/state	1
/joint_states	200

Table 2.1: ROS topics and their publishing frequency.

2.1.5 Force/torque sensor at the base

Attached to the gripper's wrist is a 6-axis force/torque sensor. Because this thesis focuses on the gripper-object interaction only, and this sensor is mostly useful when attached to a robot arm, I am not going to use it during the experiments.

2.1.6 Forward kinematics

The forward kinematics is taken from [14]. Transformation matrix ${}^{i-1}T_i$ from the coordinate system $i - 1$ to i is equal to

$$\begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i) \cos(\alpha_{i-1}) & \cos(\theta_i) \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -d_i \sin(\alpha_{i-1}) \\ \sin(\theta_i) \sin(\alpha_{i-1}) & \cos(\theta_i) \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & d_i \cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.1)$$

where:

- a_{i-1} is the distance from z_{i-1} to z_i measured along x_{i-1} ,
- α_{i-1} is the angle between z_{i-1} to z_i measured about x_{i-1} ,
- d_i is the distance from x_{i-1} to x_i measured along z_i ,
- θ_i is the angle between x_{i-1} to x_i measured along z_i .

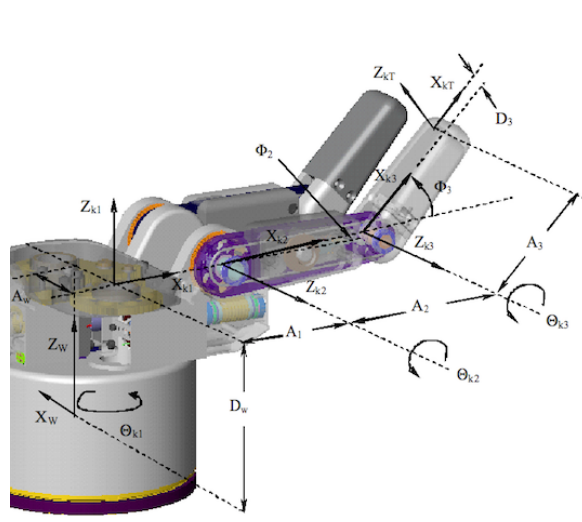


Figure 2.4: DH frame assignment for generalized finger from [14].

Joint	a_{i-1}	α_{i-1}	d_i	θ_i
1	$r \cdot A_W$	0	D_W	$r \cdot \Theta_{k1} - j \frac{\pi}{2}$
2	A_1	$\frac{\pi}{2}$	0	$\Theta_{k2} + \Phi_2$
3	A_2	0	0	$\Theta_{k3} + \Phi_3$
4	A_3	$-\frac{\pi}{2}$	D_3	0

Table 2.2: Variables of the transformation matrix for each joint.

Θ_{k1} , Θ_{k2} , Θ_{k3} are the values read from the gripper's encoders from the `/joint_states` topic. Values of the rest of the variables are in Table 2.2. Parameters used are in Table 2.3 and 2.4. I did not find an option to figure out Φ_2 initialization offset, so I am going to assume it is 0° , which may cause slight inaccuracy. In Figure 2.4 the kinematic chain is annotated.

If we choose the base of the Barrett Hand as the origin and compute the transformation matrices and then multiply them:

$${}^W\mathbf{T}_1 {}^1\mathbf{T}_2 {}^2\mathbf{T}_3 {}^3\mathbf{T}_4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad (2.2)$$

Then, the resulting vector (without the fourth coordinate) is the fingertip's position in Cartesian coordinates.

The python script to compute forward kinematics is located in [10] at path `BarrettHand/preprocessing/fwd_kinematics.py`. I will use it while plotting measurement to compute the displacement of fingers from the palm.

Parameter	A_W	A_1	A_2	A_3
Value	25 mm	50 mm	70 mm	50 mm
Parameter	D_W	D_3	Φ_2	Φ_3
Value	84 mm	9.5 mm	0° to 0.4°	42°

Table 2.3: BH-282 parameters.

	F1	F2	F3
k	1	2	3
r	-1	1	0
j	1	1	-1

Table 2.4: Constants values for each finger for forward kinematic.

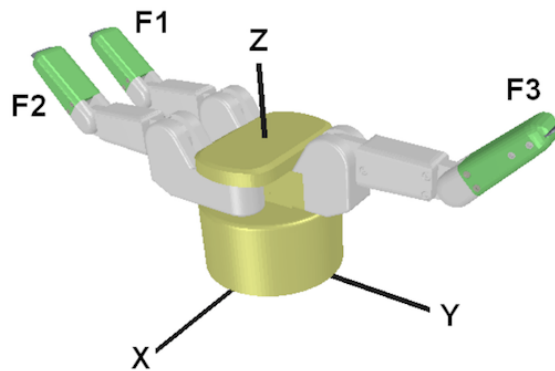
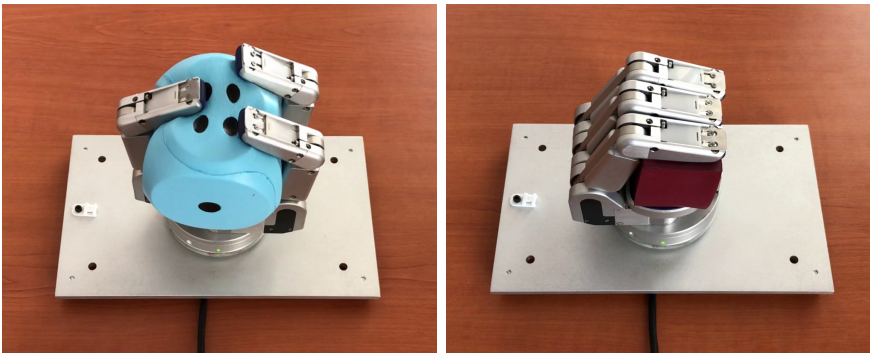


Figure 2.5: Barrett Hand axis, [14].

2.2 Action primitives

The gripper allows numerous configurations of the three fingers. However, I will define three I am going to use during the experiments for my purposes.

1. configuration 1 — opposite finger configuration, Fig. 2.6a
2. configuration 2 — opposite finger configuration without the utilization of the palm tactile array¹, not shown
3. configuration 3 — lateral finger configuration, Fig. 2.6b



(a) : Finger configuration 1 (opposing fingers).

(b) : Finger configuration 3 (lateral fingers).

Figure 2.6: Used finger configurations.

The two configurations I am going to use can be seen in Figure 2.6. Configuration 2 was also specified during the pilot data collection, representing a top grasp with opposing finger with the Barrett Hand attached to the KUKA arm. However, due to time limitations, this configuration will not appear in the rest of the work.

Not all the configurations are suitable for all the objects. For example, the “blue die” (introduced later in Table 2.5) can not be squeezed in configurations with lateral fingers as it will not fit under the fingers. Video of such attempt can be seen in [15] `objects_180320/video/bbdie_a3_s0.3_t10_n1.mov`. Other items, where the lateral finger configuration can not be used, are marked in Table 2.5.

The configuration alone is not enough to define an action, as the speed can prove just as important. Therefore I propose the action is defined as a tuple, where the first value denotes the configuration ID and the second one speed, for example:

$$(1, 0.6)$$

¹Intended for the use when attached to the robot arm and performing a top grasp on an object lying on the table.

would mean configuration with opposing fingers at speed 0.6 rad/s. For a complex grasp, a list of tuples could describe the speed of each of the controllable joints from a specified initial state, but I am going to focus on the mentioned simplified actions.

2.3 Objects and materials

As mentioned previously, I am going to focus on soft objects. Furthermore, the objects should be homogeneous and of a simple shape, such as cubes or rectangular prisms. This requirement is essential, so the surface area in contact with the gripper will not change dramatically. For example, if the object would be a sphere, the closer the gripper gets to the center, the bigger the contact area is, which will affect the values measured. Overall, at my disposal were numerous ordinary objects, the whole YCB objects set and 20 polyurethane foams.

The first set used are objects mainly bought in stores and one objects from the YCB dataset [16] — the rectangular prism with three holes in it, which is also the only significantly non-homogeneous object in my experiments. The objects’ dimensions are in Table 2.5 and their appearance in Figure 2.7. During the pilot grasping experiments, my colleagues and I noticed that the Kinova cube is an appropriate size for all of the grippers in our laboratory. Because of that, I cut two more cubes of the same dimensions from different materials. Therefore the “blue cube” is the same material as the “blue die” and “yellow cube” is the same material as the “yellow sponge”.

Description	Label	Dimensions [mm]
Kinova cube	kinova	56x56x56
Blue cube	bdielcube	56x56x56
Yellow cube	yspongecube	56x56x56
Blue die ²	bbdie	90x90x90
White die	germandie	59x59x59
Pink die ²	mpdie	75x75x75
Darkblue die	sbdie	43x43x43
YCB object	ycb	75x50x50
Yellow sponge	yellow sponge	195x135x65

Table 2.5: Dimensions and labels of objects set.

The second set I focused on consists of polyurethane foams. Their names, dimensions and parameters from datasheets are listed in Table 2.6. Figure 2.8 depicts the foams. Some of them (GV and V types) have a memory-foam-like behavior — after compression, they return to their original dimensions very slowly. Thanks to the datasheets provided to us along with the foams, I can use their parameters as a ground truth for the experiments. I am going to call Compression stress value at 40% (CV_{40}) (defined in ISO standard [17])



Figure 2.7: Objects.

elasticity for simplicity. Elasticity is closely related to stiffness. However, stiffness takes into account both the material's elasticity and geometry.

Type	Dimensions [mm]	Density [$\text{kg} \cdot \text{m}^{-3}$]	Elasticity CV_{40} [kPa]
V4515	118x120x40	45	1.5
V5015	119x120x42	50	1.5
GV5030	118x119x40	50	3.0
GV5040	118x118x39	50	4.0
N4072	118x117x37	40	7.2
NF2140	105x100x50	21	4.0
T1820	125x125x50	18	2.0
T2030	125x120x40	20	3.0
T3240	123x123x50	32	4.0
T2545	125x125x50	25	4.5
RL3529	119x118x40	35	2.9
RL4040	117x120x40	40	4.0
RL5045	118x118x39	50	4.5
RP1725	118x120x41	17	2.5
RP2440	118x120x38	24	4.0
RP27045	117x119x39	270	4.5
RP30048	123x121x39	300	4.8
RP3555	117x119x39	35	5.5
RP2865	118x118x38	28	6.5
RP50080	121x118x39	500	8.0

Table 2.6: Properties of used polyurethane foams, CV_{40} stands for “compression stress value at 40%”, [17].



Figure 2.8: Foams.

2.4 Soft object classification and stiffness/density regression

Using time-series of tactile and torque feedback from the Barrett Hand, I will attempt to classify objects from different sets as well as determine their stiffness/elasticity and density. To this end, the Long short-term memory recurrent neural network will be employed.

To deal with variable-length time series measured on the Barrett Hand, there are two options:

1. divide time series into a fixed number of sections and compute features for each of them
2. use the time series itself as an input

In this section, I am going to focus on the second option. I created a long short-term memory (LSTM) neural network using the PyTorch Python library [18]. All of the codes mentioned in this section are available in [10].

2.4.1 Long short-term memory neural network

LSTM is a type of recurrent neural network (RNN) first introduced in [19] and has been a popular option for sequence learning, such as text or even speech recognition and translation. It tackles the vanishing gradient problem present when learning very long dependencies with regular RNNs. The general architecture of an LSTM cell is shown in Figure 2.9. There are many variations, but I will be using this as a “black box” from the PyTorch library.

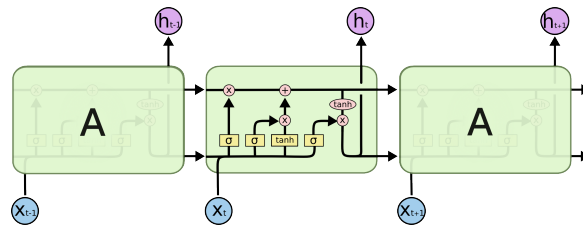


Figure 2.9: LSTM cell in detail, [20]. Yellow rectangles are learnable layers, pink circles are pointwise operations, lines merging represent concatenation, forking copies the data into two vectors.

In addition to the official PyTorch forums, my main sources were articles [21] and [22]. To address the variable-length problem, I studied [23]. To load the dataset, I wrote my own function `create_dataset` available in `Barrett_neural/lstm_utilities.py`, which loads the `.npz` data files (NumPy file format) and optionally can omit tactile or torque sensory channels — this will be used in the ablation studies later on. Next, a loader is created with a specified batch size in `create_loader`. In each batch, the data are

padded with zeros to the length of the longest measurement, but information about original length is kept. From the loader, the time series is passed into the LSTM layer. According to the original length, the output of the last LSTM cell (representing the LSTM-computed features) with non-zero input is selected. The features are then passed into two linear layers to compute the output. Then, one output is selected using the softmax layer in the case of classification, or left untouched in the case of regression — more on that later. The architecture of the network is in `Barrett_neural/lstm_model.py`.

Each sensory channel represents a single time series being analyzed. The values are not modified in any way; I am relying on the neural network to extract the features. Sensory channels are:

1. taxels³ from each tactile matrix ($4 \cdot 24 = 96$)
2. joint coordinates (8)
3. fingertip joint torque (3)

During training, the PyTorch automatic differentiation engine is called and used to backpropagate from the loss of the output and weights are updated according to the set optimizer. Data are divided into the training and validation set. Specific datasets will be introduced in Section 3.3.

2.4.2 Problem formulation

Given $X = \mathbb{R}^{s \times n}$ represents the space of all possible measurements and $Y = \mathbb{R}$ the space of labels, I am searching for a function f ,

$$f : X \rightarrow Y, \quad (2.3)$$

which will map an input $\mathbf{X} \in X$ to a label $y \in Y$. The input of the function \mathbf{X} is a specific measurements represented by a matrix of dimensions $s \times n$, where s is the number of sensory channels and n number of measurements.

I am going to list the possible use-cases. The network architecture is the same for each of them, but the labels differ. The number of sensory channels is automatically determined by the `dataset_creation` function mentioned earlier.

Classification

- Training script: `BarrettHand/neural/lstm_train_class.py`
- Benchmark: $\text{accuracy} = \frac{\text{number of correctly classified measurement}}{\text{total number of measurements}}$

³Taxel is usually used as a tactile sensor synonym in robotics. The abbreviation comes from TActile piXEL.

- Input labels: specific objects
- Output: specific objects' labels

■ Classification into discrete intervals

- Training script: `BarrettHand/neural/lstm_train_bin.py`
- Benchmark: accuracy = number of correctly classified measurement / total number of measurements
- Input labels: specific foams with their elasticity/density interval, which is assigned based on the foam's reference value, see Table 2.6
- Output: elasticity/density intervals

Three intervals were created for elasticity, later for density. There are always fewer of the stiffer/denser foams, so I incorporated weights calculation to the `create_dataset` function to avoid skewing the weights in favor of the class with the most training examples⁴. Furthermore, I tried to create the borderline between the classes to maximize the region with no foams of such elasticity/density, while roughly maintaining the balance between the number of training examples.

To enable results for individual foams in the validation set, evaluation process and some of the data loading components had to be changed. They are defined along with the training script in the `BarrettHand/neural/lstm_train_bin.py` script. That enables us to see individual foams in the confusion matrix instead of just the class. Therefore, results and confusions can be better understood — for example, when the foam is the same material as another but has a higher density/elasticity.

■ Regression

- Training script: `BarrettHand/neural/lstm_train_regression.py`
- Benchmark: mean absolute error (MAE)
- Input labels: specific foams with their elasticity/density reference value, see Table 2.6
- Output: continuous number representing elasticity/density

As accuracy could not be used because of the continuous output (even a small difference such as 10^{-12} would indicate incorrect output), mean absolute error was used instead. However, this can still be skewed by a few consistently incorrect foams, function to plot the results were added. An example from training is shown in Figure 2.10.

⁴This was not necessary for the ordinary classification, as the training sets were balanced.

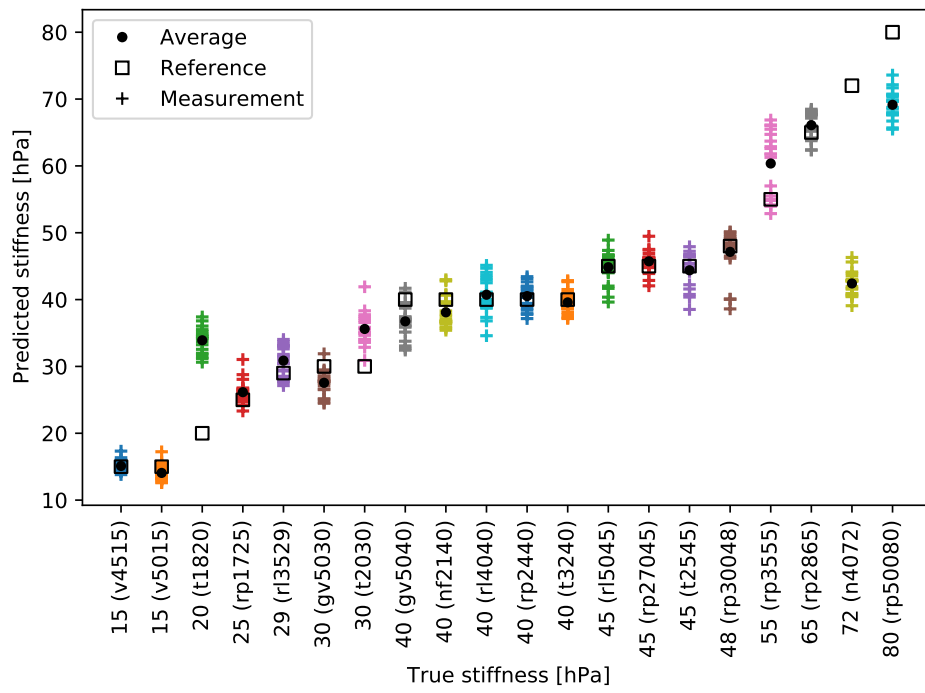


Figure 2.10: Elasticity regression output plot.

Chapter 3

Data collection

This section is going to describe data measuring, their pre-processing and visualization. Furthermore, they will be organized into datasets (available at [24]), which can then be directly used to train the neural network mentioned in the previous chapter.

3.1 ROS package

To begin with, I created a ROS package, which would enable me to measure the process of squeezing an object. For all of the measurements, velocity control is used, as it closes the hand at a constant speed. However, this also means the internal hardware controller is used, and the joints' force can not be set explicitly.

By reverse-engineering Barrett Hand's GUI controller [12] I was able to write a script `BarrettHand/bhand_gather/scripts/grasp_object.py` available at [10]. Its purpose is to make measuring simple: the user specifies the tactile threshold, speed, finger configuration, object name and measurement number. ROS bag is then started to save all the messages into a `.bag` file, which will be processed later. Then, the hand closes at the set joint speed published to the `/joint_states` topic for each joint (see Section 2.1.1), until one of the three events occur:

1. joints reach their maximum position; therefore the hand can not close any further
2. threshold is exceeded on any of the taxels
3. movement stops because of the object resistance

After one of those events, the measurement stops and the `.bag` file is saved in a specified location. The location and default used-defined values, such as speed, can be set in `BarrettHand/bhand_gather/scripts/config.py`.

The next step is to convert the `.bag` files to `.npz` files; a NumPy zipped archive of `.npy` files, each containing and named after one of the saved variables. As the rest of the work is written in Python, they are more convenient to work with. Script `BarrettHand/pre-processing/process_data.py` is designed to do precisely that.

3.2 Visualization

Although all of the data are measured at discrete time steps, I am going to use the function `matplotlib.pyplot.step` instead of `matplotlib.pyplot.scatter`. This keeps the last value until the next one is received. This is especially useful for the tactile sensors, because, as mentioned in Section 2.1.4, values appear to refresh faster than they do. Another reason is that there are simply too many values and plotting it as a scatter appeared more confusing.

In Figure 3.1, activations of all taxels for each of the tactile surfaces while squeezing an object are shown — each line represents one taxel. This is not very clear; therefore, I will always show only one plot per tactile surface, such as average in Fig. 3.2a, where F1, F2, F3 and PALM refer to the tactile surfaces. Both in the original plots of all taxels and in the averaged plot, we can notice they start at different values, although I would expect all the values to be around zero and then going up at roughly 5.5 s when the gripper started squeezing the object. This is because of uncalibrated tactile sensors. Therefore, some data pre-processing was necessary.

All of the plots can be created by `BarrettHand/pre-processing/plot_data.py`.

3.2.1 Data preprocessing

First, I created a new script to truncate data, available at [10] in `BarrettHand/preprocessing/truncate_data.py`. It opens a `.npz` file created previously and shows a plot such as Fig. 3.2b showing both the averaged tactile sensors in relation to the right pressure y -axis and joint torque in relation to the left torque y -axis. F1, F2, F3 and PALM pressure refer to the tactile surfaces, F1_TIP, F2_TIP and F3_TIP torque refer to the fingertip joint torque sensors. The user clicks on any of the torque plots and selects where the gripper first touched the object.

An average of each individual taxel is computed from the last 100 samples before the first contact and saved to a new `.npz` file, which includes only the

¹We can notice the green taxel is moving in the opposite direction than all of the others. I believe this is caused by the incorrect installation of the sensor onto the tactile matrix, specifically in the opposite direction that all of the others. Because of this, I am going to ignore this taxel in the detection of active taxels, but the neural network will still get its data.

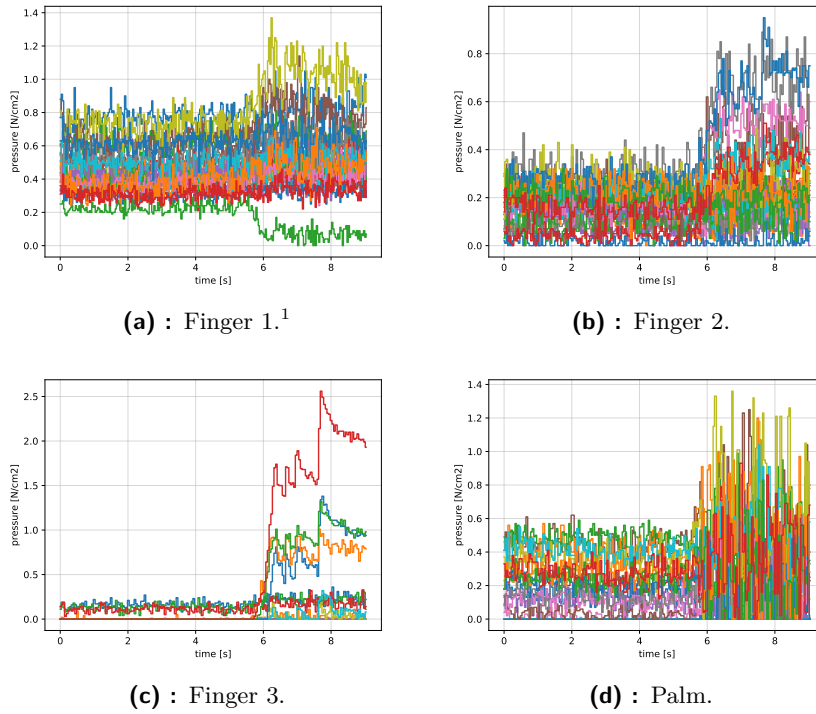


Figure 3.1: Plots of all taxels on each of the tactile surfaces. Foam GV5030, opposite finger configuration, speed 0.3 rad/s, non-truncated.

data after initial contact. It is not computed from the whole portion before the contact, because I noticed changes in the values as the Barrett Hand's temperature increases while in operation, therefore, taking the most recent values should, in a way, compensate the thermal drift. I considered some form of automatic detection as well, but due to the uncalibrated tactile sensors and the idle torque values being different for each speed, this proved rather difficult, if not impossible.

The new .npz file with an average for each taxel enables us the active taxel detection. A taxel is considered active when it crosses 130% of the average value before the first contact — the additional 30% is to ignore the taxels' noise. The average value is subtracted, therefore calibrating the taxel to return values around zero when inactive. A new plot can be created, which only computes the average of the active taxels. The updated plot is shown in Fig. 3.3.

3.2.2 Object compression

To obtain information about the object's compression, I use the forward kinematics described in Section 2.1.6. Specifically, in configuration 3, when the fingers are lateral and press against the palm. If $c(t)$ is the compression of the object at time t , $d(t)$ is the orthogonal projection of distance from

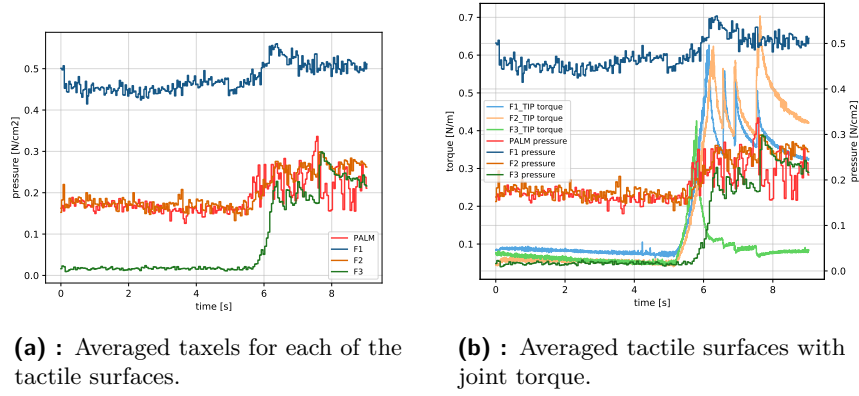


Figure 3.2: Foam GV5030, opposite finger configuration, speed 0.3 rad/s, non-truncated.

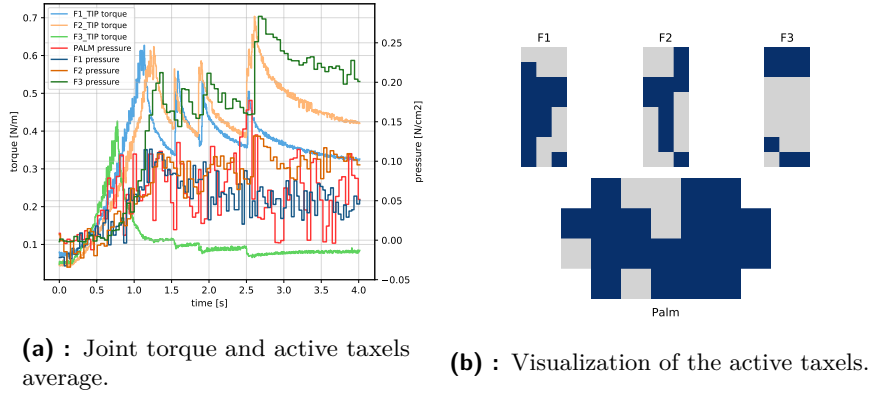


Figure 3.3: Foam GV5030, opposite finger configuration, speed 0.3 rad/s, truncated.

fingers to the palm along the z -axis (Fig. 2.5) at time t and t_0 is the moment of the first contact, then the compression function is:

$$c(t) = |d(t) - d(t_0)|. \quad (3.1)$$

However, the same is not applicable to configuration 1 with opposing fingers. The distance component along the y -axis could be computed, but the fingers can bend the object and even cross each other in some cases; thus, the projection would not reflect the object's deformation.

There are two types of compression plots. First, shown in Figure 3.4, displays the compression of the object $c(t)$ throughout the time. The second type, Figure 3.5, assigns the compression to each time-step, sorts the torque and pressure data in ascending order according to the compression and plots it. As the compression can remain the same for some time while the torque and pressure values change, there can be more than one value for one x -axis compression value.

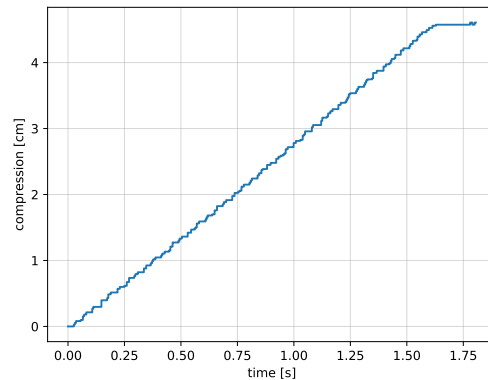


Figure 3.4: Compression of V4515, lateral finger configuration, speed 0.3 rad/s, truncated.

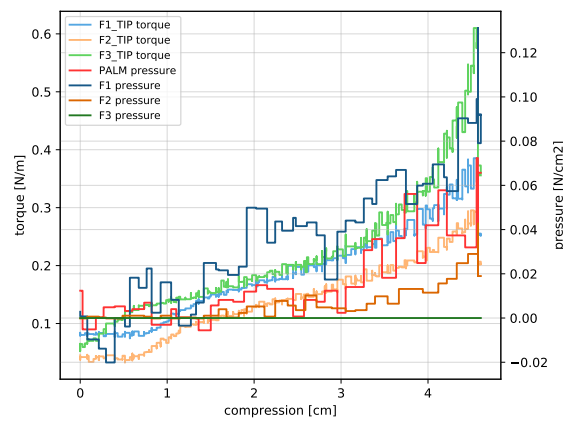


Figure 3.5: Torque and pressure vs compression of V4515, lateral finger configuration, speed 0.3 rad/s, truncated.

3.2.3 Overview of used plots

To complete the overview of plots, I am adding a joint coordinates plot in Figure 3.6. This leaves us with the following plots:

1. pressure of all taxels for each tactile surface vs. time
2. pressure of active taxels² for each tactile surface vs. time
3. average pressure of all taxels for each tactile surface vs. time
4. average pressure of active taxels² for each tactile surface vs time
5. joint coordinates vs. time
6. joint torque vs. time

²Available for manually truncated measurements.

3. Data collection

7. joint torque and average pressure³ vs. time
8. joint torque and average pressure³ vs compression of the object⁴
9. compression of the object vs time⁴
10. active taxels map

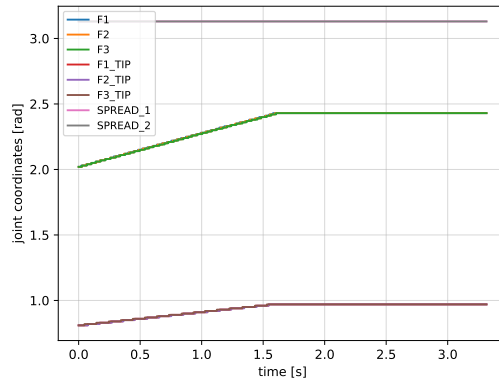


Figure 3.6: Joint coordinates vs time of GV5030, lateral fingers, speed 0.3 rad/s.

³Average pressure of active taxels if the measurement is truncated, the average pressure of all taxels otherwise.

⁴Available for lateral finger configuration only.

3.3 Datasets

To be able to train the neural network, various datasets for studying different aspects of object parameters were created from the individual measurements (specific measurement can be in more than one dataset, datasets available at [24]). In Table 3.1, an overview of all datasets is presented, including the parameters of training and validation sets.

Dataset name		Action	Measurements
action1-03	trn	(1, 0.3)	4-6
	val	(1, 0.3)	4
	labels		all objects
action3-03	trn	(3, 0.3)	4-6
	val	(3, 0.3)	4
	labels		all objects except blue die, purple die
objects1	trn	(1, 0.3)	8-10
	val ⁵	(1, 0.6)	8-10
	labels		all objects
objects2	trn	(1, 0.3)	8-10
	val ⁵	(3, 0.3)	8-10
	labels		all objects except blue die, purple die
objects3	trn	(1, 0.3), (3, 0.3)	8-10
	val ⁵	(1, 0.6), (3, 0.6)	8-10
	labels		all objects
objects4	trn	(3, 0.3)	8-10
	val ⁵	(3, 0.6)	8-10
	labels		all objects
objects5	trn	(1, 0.3)	8-10
	val	(1, 1.2)	8-10
	labels		all objects
objects6	trn	(1, 0.6)	8-10
	val	(1, 1.2)	8-10
	labels		all objects
foams	trn	(1, 0.3), (3, 0.3)	14-16
	val	(1, 0.3), (3, 0.3)	4
	labels		all foams
stiffbin	trn	(1, 0.3), (3, 0.3)	[127, 138, 48]
	val	(1, 0.3), (3, 0.3)	[28, 40, 12]
	labels		elasticity intervals [15-30, 40-55, 65-80]

Table 3.1: Dataset overview. Trn is short for training set, val for validation and labels. Measurements are examples per class, total number of examples for regression datasets. Datasets which have 's' appended at the end of the name have the training and validation set switched. Actions are in accordance with the notation established in Section 2.2.

Dataset name		Action	Measurements
stiffbin2	trn	(1, 0.3), (3, 0.3)	[120, 155, 40]
	val ⁵	(1, 0.3), (3, 0.3)	[19, 39, 20]
	labels	elasticity intervals [15-30, 40-55, 65-80]	
stiffbin3	trn	(1, 0.3)	[60, 79, 20]
	val ⁵	(1, 0.3)	[10, 20, 10]
	labels	elasticity intervals [15-30, 40-55, 65-80]	
stiffbin4	trn	(3, 0.3)	[60, 76, 20]
	val ⁵	(3, 0.3)	[9, 19, 10]
	labels	elasticity intervals [15-30, 40-55, 65-80]	
densebin	trn	(1, 0.3), (3, 0.3)	[154, 111, 48]
	val	(1, 0.3), (3, 0.3)	[40, 28, 12]
	labels	elasticity intervals [15-35, 40-50, 270-500]	
densebin2	trn	(1, 0.3), (3, 0.3)	[154, 119, 40]
	val ⁵	(1, 0.3), (3, 0.3)	[40, 20, 20]
	labels	density intervals [15-35, 40-50, 270-500]	
densebin3	trn	(1, 0.3)	[79, 60, 20]
	val ⁵	(1, 0.3)	[20, 10, 10]
	labels	density intervals [15-35, 40-50, 270-500]	
densebin4	trn	(3, 0.3)	[75, 59, 20]
	val ⁵	(3, 0.3)	[20, 10, 10]
	labels	density intervals [15-35, 40-50, 270-500]	
stiffregress_old	trn	(1, 0.3), (3, 0.3)	334
	val ⁵	(1, 0.3), (3, 0.3)	61
	labels	continuous elasticity	
stiffregres_new	trn	(1, 0.3), (3, 0.3)	275
	val ⁵	(1, 0.3), (3, 0.3)	61
	labels	continuous elasticity	
experiment	trn	(1, 0.3), (3, 0.3)	395
	val	empty	
	labels	continuous elasticity	
experiment_a1	trn	(1, 0.3)	200
	val	empty	
	labels	continuous elasticity	
experiment_a3	trn	(3, 0.3)	195
	val	empty	
	labels	continuous elasticity	

Table 3.1: Dataset overview. Trn is short for training set, val for validation and labels. Measurements are examples per class, total number of examples for regression datasets. Datasets which have 's' appended at the end of the name have the training and validation set switched. Actions are in accordance with the notation established in Section 2.2.

⁵The validation dataset consists of never-before-seen objects.

Chapter 4

Results

In this chapter, I am going to present the results of my work. I will refer to the datasets from previous chapter, see Table 3.1 and Figures 2.7 and 2.8. For action primitives and finger description, please refer to Section 2.2 and Figure 2.5.

4.1 Data comparison

To get a qualitative idea of what the data look like, I am going to present some plots of object compression.

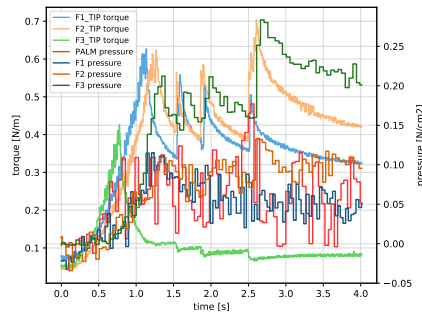
In Figure 4.1, tactile and torque plots of two objects made of the same material with the same density but different elasticity can be seen with their active taxels maps. The repeated peaks are caused by the internal controller of the Barrett Hand speed. From my understanding, it repeatedly readjusts the maximum motor torque safety threshold, so the motor does not get damaged, and during that time, the joint torque drops. This phenomenon is visible even for some of the objects and for other foams as well.

Next presented in Fig. 4.2 are two foams with the same elasticity, but different density and material. The scale of the pressure sensors is the same for each figure, but the RL4040 reaches higher torque values. A human can see the difference, but there is no specific trait in the curves which would enable us to tell which is which with certainty. Furthermore, we do not know which traits are related to the foam's elasticity and which to density.

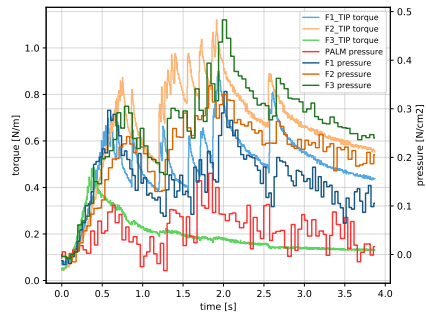
Foams with maximum and minimum elasticity are presented in Figures 4.3 and 4.4, the first one is for opposite finger configuration, the latter one for lateral. We can see the difference between configurations. In Fig. 4.3 we can once again see the repeating peaks caused by the controller.

Fig. 4.4, on the other hand, is much cleaner and the difference between the stiffer and more flexible foams is observable by the human eye — the gradient is larger for the stiffer object.

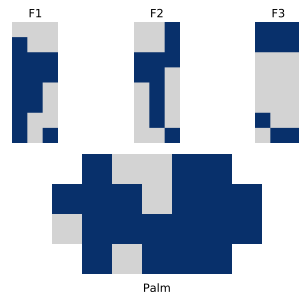
4. Results



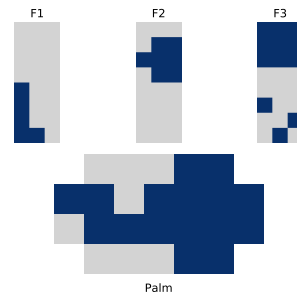
(a) : Joint torque and active taxels average of foam GV5030.



(b) : Joint torque and active taxels average of foam GV5040.

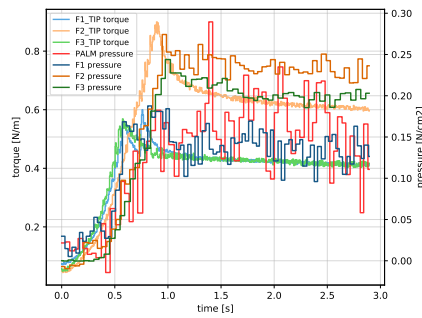


(c) : Active taxels map of foam GV5030.

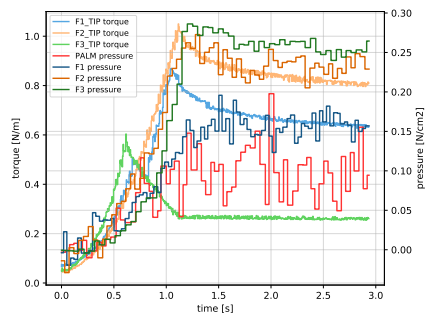


(d) : Active taxels map of foam GV5040.

Figure 4.1: Foams with different elasticity. Opposite finger configuration, speed 0.3 rad/s, truncated.

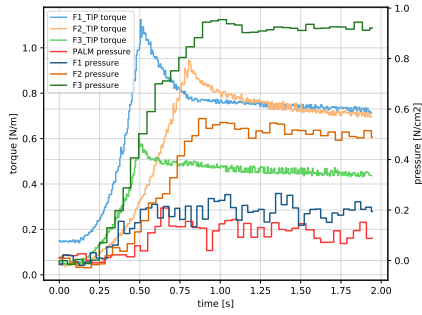


(a) : Foam RP2440.

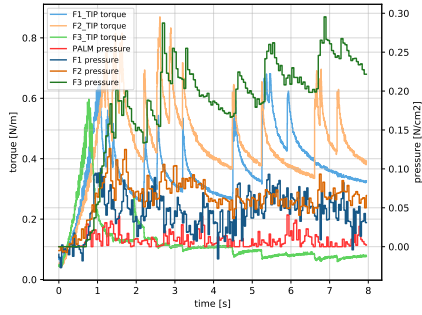


(b) : Foam RL4040.

Figure 4.2: Two foams with same elasticity but different density and material. Opposite finger configuration, speed 0.3 rad/s, truncated.

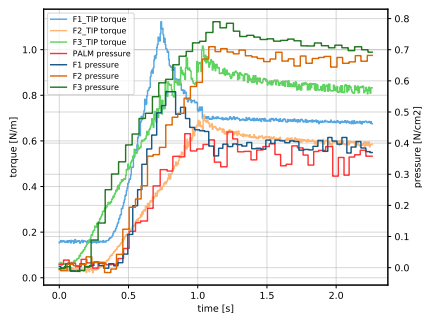


(a) : Foam RP50080, elasticity 80.

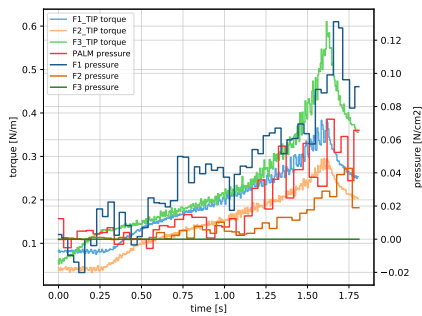


(b) : Foam V4515, elasticity 15.

Figure 4.3: Maximum and minimum elasticity. Opposite finger configuration, speed 0.3 rad/s, truncated.



(a) : Foam RP50080, elasticity 80.



(b) : Foam V4515, elasticity 15.

Figure 4.4: Maximum and minimum elasticity. Lateral finger configuration, speed 0.3 rad/s, truncated.

4.2 Object classification

In this section, a neural network I designed will try to classify the individual objects, as described in Section 2.4.2. All experiments were run with Adam optimizer with the `amsgrad` option enabled. Cross entropy loss was used as the criterion function.

I will refer to the datasets from previous chapter, see Table 3.1 and Figures 2.7 and 2.8.

4.2.1 Classification on foam set

All measurements of the polyurethane foams were taken at the speed of 0.3 rad/s. To begin with, I examined their classification on foams with both finger configurations. The experiment reached accuracy 91.25%; the classification matrix is shown in Fig. 4.5.

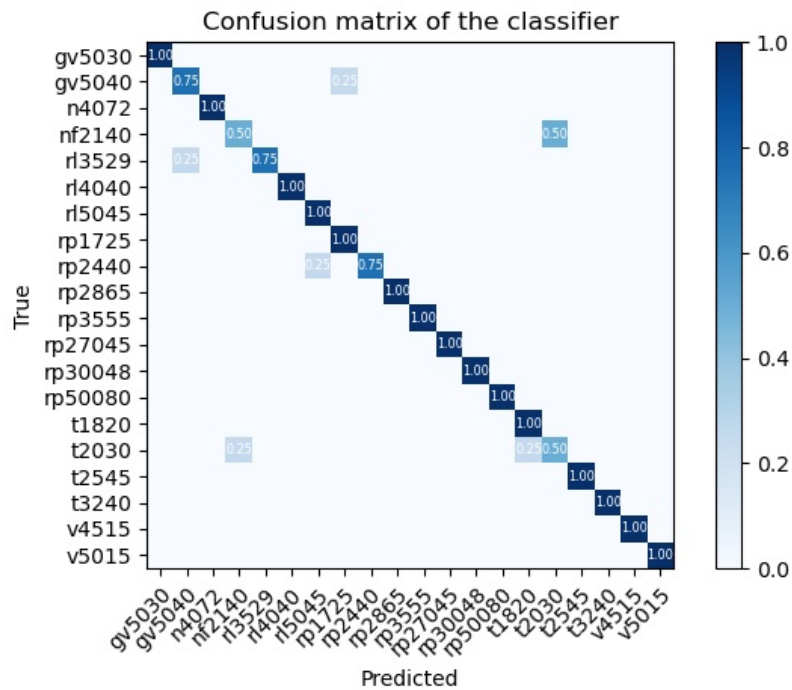
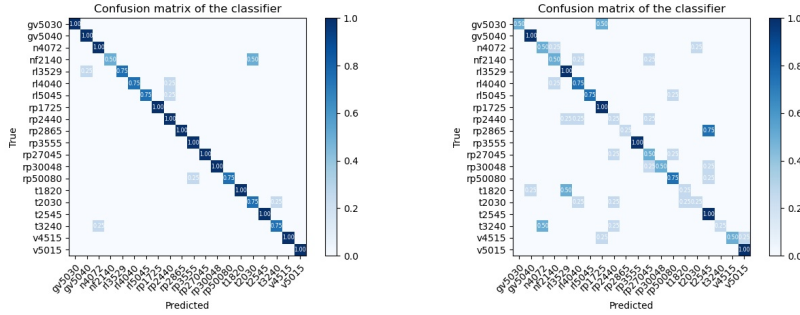


Figure 4.5: Foam20 dataset, accuracy 91.25%. This represents the ability of classifying each foam.

Next, I trained the same network but omitted data from either the torque or the tactile sensors. Confusion matrices are shown in Fig. 4.6. The ablation study showed the tactile sensors as superior over torque sensors. This is interesting because from plots, we can notice the tactile sensors introduce much noise compared to the torque ones. In my opinion, this is caused by the position information conveyed by the taxels. Furthermore, during the

training, the ablation of torque data reached an accuracy of 80% around epoch 85 in contrast to ablation of tactile data, where barely 40% were roughly reached at epoch 100 and presented models were saved at epoch 140 to effort ablation and epoch 375 for tactile ablation.

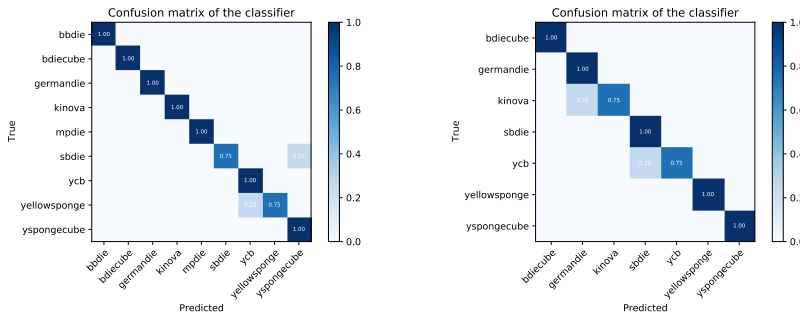


(a) : Ablation of effort data – tactile sensors only; accuracy 90.00%. (b) : Ablation of tactile data – effort sensors only; accuracy 62.50%.

Figure 4.6: Ablation study of foam20 dataset.

4.2.2 Individual actions on object set

To verify the ability to classify the objects, I created datasets for individual actions, see Section 2.2 for more info. The results are shown in Figure 4.7, both actions appear to lead to similar performance on this task.



(a) : Dataset action1-03, accuracy 94.44%. (b) : Dataset action3-03, accuracy 92.85%.

Figure 4.7: Object classification by individual actions.

4.2.3 Knowledge transfer

In this section, I am going to examine whether or not a classifier trained on a set of measurements gathered by specific action can be used to classify the same parameters when exploring the object with a different action.

■ Different finger configuration

To start with, I looked into how the transfer from one configuration to the other works at the same speed. Results are correct in most cases for both directions, but there is no particular object that prevents the classifier from reaching better accuracy—it differs from measurement to measurement. The results are shown in Fig. 4.8.

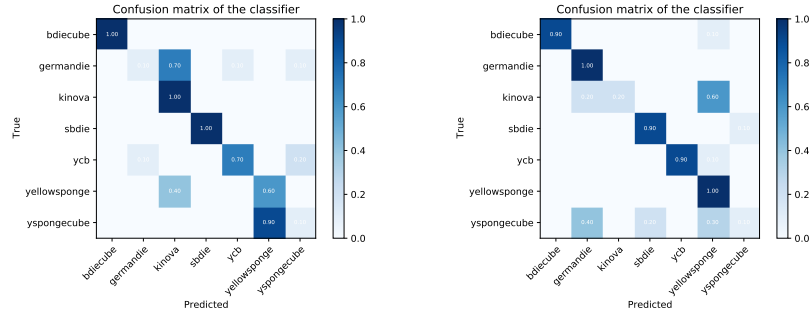


Figure 4.8: Knowledge transfer from configuration 1 to configuration 3 and vice versa at speed 0.3 rad/s.

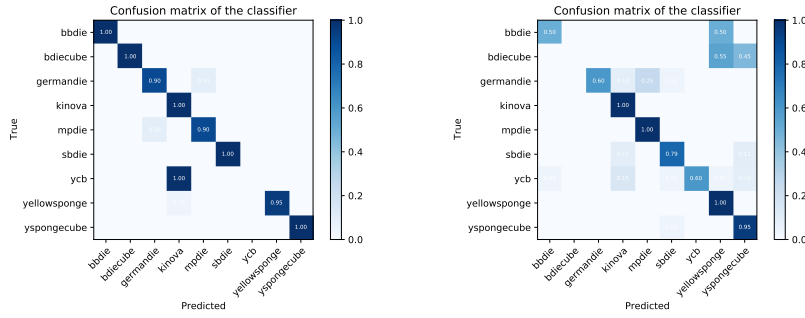
■ Different speeds of squeezing

To address the question of whether the “knowledge” of object classification is transferable to different speeds of squeezing, I experimented with datasets, where training sets consist entirely of one speed and validation sets of another.

The first one consisted of both finger configurations and is shown in Fig. 4.9. When transferring from 0.3 to 0.6 rad/s, the only object incorrectly classified in all cases is the YCB prism. A possible reason might be that the prism is extremely heterogeneous because of the holes from one side, which could have significantly distorted the measurement. Furthermore, the material of the YCB prism feels very similar to the Kinova cube. When transferring from 0.6 to 0.3 rad/s, accuracy is 14% lower.

However, better results can be obtained by separating the datasets of the two configurations, as shown in Figure 4.10 for opposite configuration and Figure 4.11 for lateral. The difference between the slower to faster and faster to slower is much more significant this time and consistent between the two finger configurations. From these results, I propose the idea that transfer is more successful from lower to faster speed rather than the opposite way. We can once again notice the incorrect classification of the YCB prism in Fig. 4.10a.

We might come to a conclusion, lateral finger configuration (3) is superior to the opposite finger configuration (1). However, configuration 3 has an

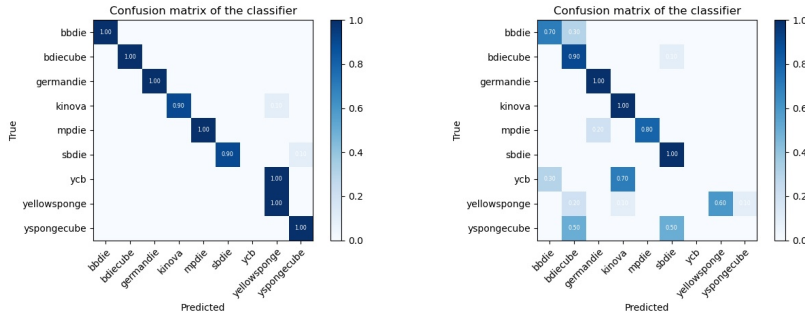


(a) : Objects3 dataset, accuracy 84.90%, 0.3 \rightarrow 0.6 rad/s.

(b) : Objects3s dataset, accuracy 71.06%, 0.6 \rightarrow 0.3 rad/s.

Figure 4.9: Knowledge transfer from 0.3 to 0.6 rad/s and vice versa for both finger configurations.

easier classification problem because the larger objects could not be grasped in lateral finger configuration.



(a) : Objects1 dataset, accuracy of 86.66%, 0.3 \rightarrow 0.6 rad/s.

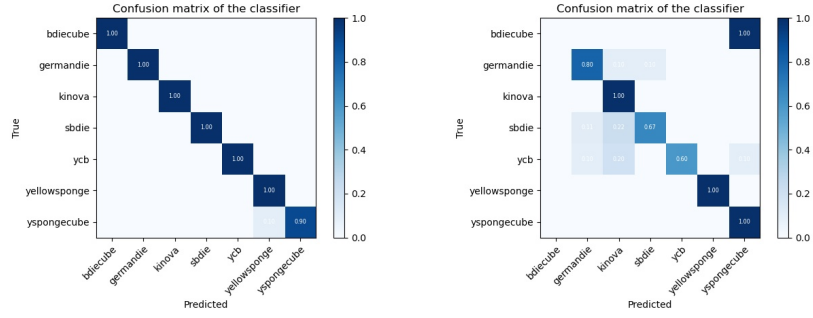
(b) : Objects1s dataset, accuracy 66.66%, 0.6 \rightarrow 0.3 rad/s.

Figure 4.10: Object classifier knowledge transfer for speeds 0.3 and 0.6, opposite finger configuration.

To verify this idea, more speeds need to be tested. In Fig. 4.12 speed transfers 0.3 \rightarrow 1.2 and 0.6 \rightarrow 1.2 rad/s are portrayed. We are confronted with lower accuracy at 0.6 \rightarrow 1.2 transfer. Surprisingly, training at a much lower speed appears to be better than training at a higher speed for the same validation speed of 1.2 rad/s.

For completeness, Fig. 4.13 displays knowledge transfer from 1.2 to 0.3 and 0.6 rad/s. We can see the results are similar to the previous ones, that is lower accuracy than from slower to faster. Difference between Fig. 4.13b and Fig. 4.12b is more marginal but still noticeable.

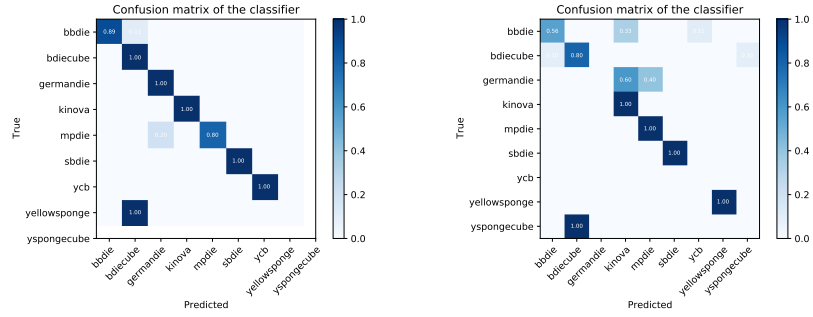
4. Results



(a) : Objects4 dataset, accuracy 98.55%, 0.3 \rightarrow 0.6 rad/s.

(b) : Objects4s dataset, accuracy 72.46%, 0.6 \rightarrow 0.3 rad/s.

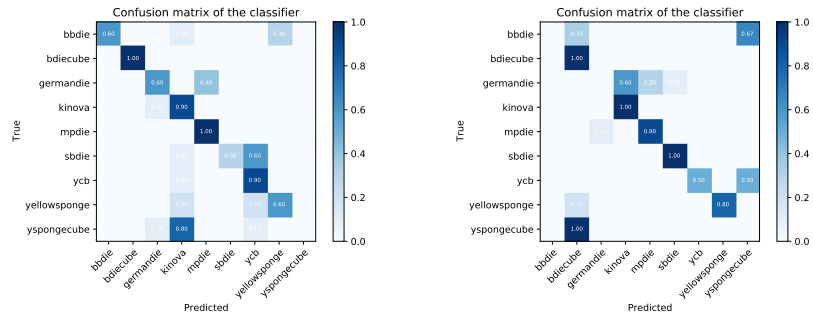
Figure 4.11: Object classifier knowledge transfer for speeds 0.3 and 0.6, lateral finger configuration.



(a) : Objects5, acc 83.54%, 0.3 \rightarrow 1.2.

(b) : Objects6, acc 67.08%, 0.6 \rightarrow 1.2.

Figure 4.12: Object classifier knowledge transfer for speeds 0.3 \rightarrow 1.2, 0.6 \rightarrow 1.2, opposite finger configuration.



(a) : Objects5s, acc 65.55%.

(b) : Objects6s, acc 58.42%.

Figure 4.13: Object classifier knowledge transfer for speeds 0.3 \rightarrow 1.2, 0.6 \rightarrow 1.2, opposite finger configuration.

4.3 Classification into discrete intervals

In this section, foams will be sorted into intervals based on their elasticity or density, as described in Section 2.4.2. For the datasets used, please refer to Table 3.1. Adam optimizer and cross-entropy loss were used.

4.3.1 Elasticity

In Figure 4.14, confusion matrices of datasets created for elasticity interval classification are shown. Subfigure on the stiffbin dataset contains training data for all of the objects and four never before seen measurements are used for validation. The accuracy of this experiment is 92.50 %, which was expected, as the objects were the same in both training and validation sets, validation set just consists of different measurements.

When some objects are completely separated¹, as shown in the other three confusion matrices, accuracy drops significantly. Stiffbin2 contains the configurations mixed together, while stiffbin3 contains only opposite finger configuration and stiffbin4 only lateral configuration. However, no configuration appears to be superior in this task.

Later, my colleague Pavel Stoudek discovered during his master's thesis that some of the foams do not correspond to their rated elasticity or density values, one of them being the N4072. This could explain its incorrect classification. I am going to discuss this further in Chapter 5.

¹Meaning the network didn't see any of their measurements during the training phase.

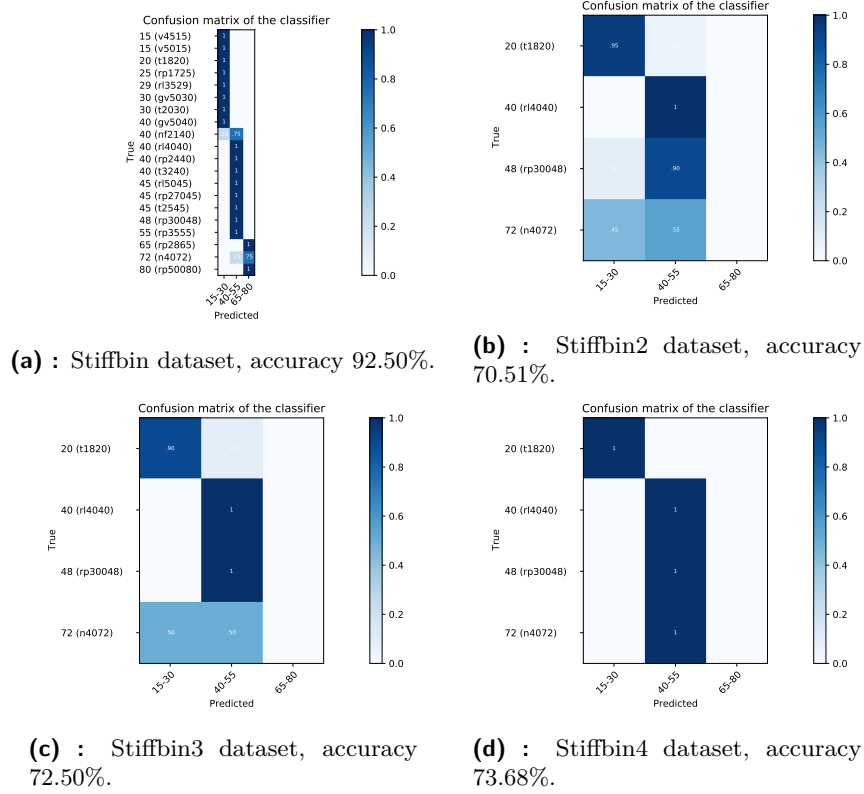


Figure 4.14: Elasticity interval classification.

4.3.2 Density

To see whether the interval classification method will yield better results when classifying by density, another four datasets were created with the same measurement distribution as in the case of elasticity. Again, the dataset with all of the foams partly in both training and validation sets reached almost perfect accuracy, see Fig. 4.15a.

The accuracy was significantly lower when some of the foams were not in the training datasets, as seen in 4.15b for densebin2. Although the accuracy of densebin4 (lateral configuration only) was similar to densebin2, densebin3 exceeded that and reached 100% accuracy consistently around epoch 20. From these results, we can deduce that the opposite finger configuration is better at density classification.

Some of the accuracy differences compared to the stiffbin datasets could result from different validation foams selection.

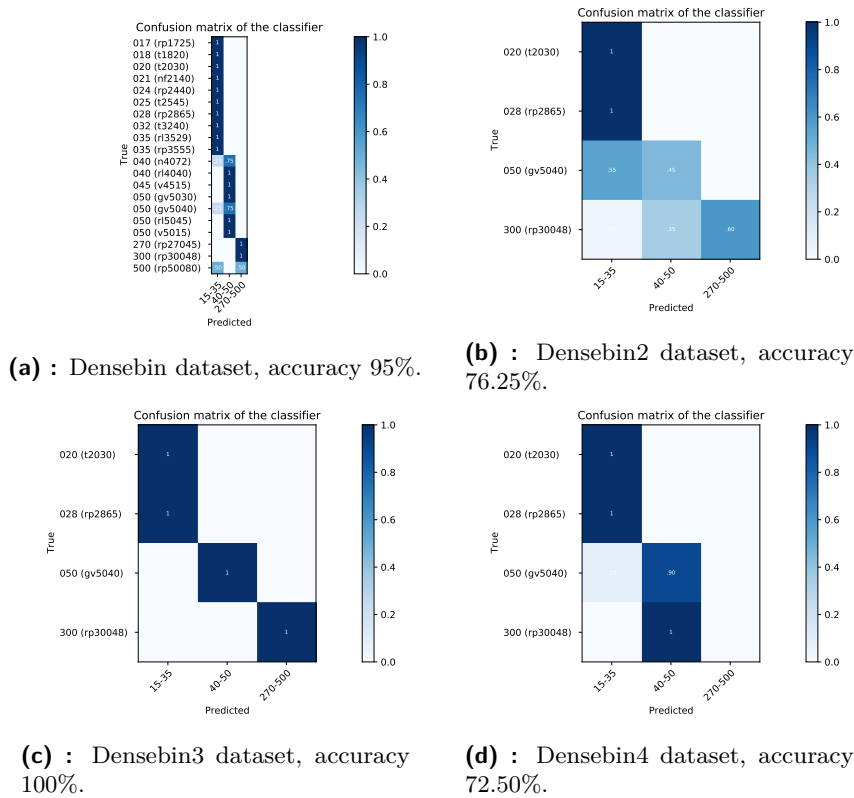


Figure 4.15: Densebin datasets.

4.4 Elasticity regression

Unlike the previous neural networks, this one was trained with SGD with momentum. SmoothL1Loss was used as the criterion function.

The goal of the elasticity regression network was to output a number corresponding to the elasticity coefficient for any object grasped. Training dataset had discrete labels, which were the elasticity of the foams in hPa. The best model was evaluated based on mean average error (MAE). A few incorrectly classified measurements can significantly distort this metric, therefore I also created a plot to show the goal, average and all of the measurements. Best training is shown in Figure 4.16. Foam with elasticity 30 is rated similarly as the foam with elasticity 40. When looking at the same plot for the training dataset, we can see some of the foams' elasticity values are not correct even there.

I am again going to refer to the information from my colleague Pavel Stoudek, who found out some of the foams are incorrectly labeled [25]. I decided to try to determine which ones are off and by how much by only training the neural network on all foams. From the previous experiment, I knew that after cca. 200th epoch, the model starts to overlearn and loses the ability to generalize. However, I have to assume the majority of the references

4. Results

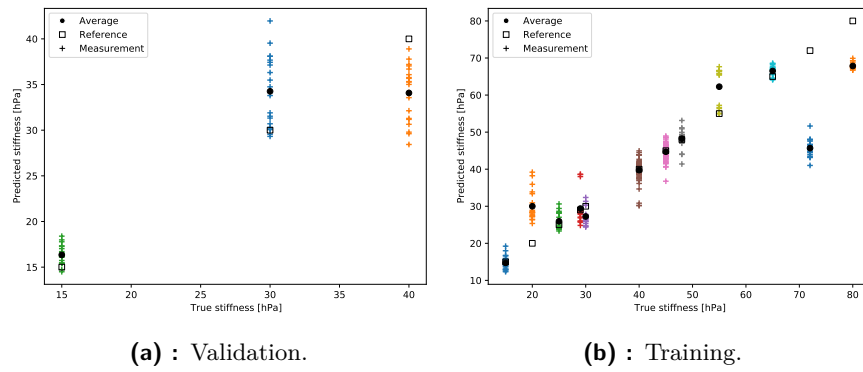


Figure 4.16: Dataset stiffregres_old, MAE 3.90.

are correct, otherwise, this approach will not yield correct results.

Figure 4.17 shows training on both finger configurations. N4072 is classified roughly at elasticity 40 instead of 72 hPa, T1820 is classified as stiffer than the stated value, RP50080 has a similar elasticity to RP2865. All of these findings are consistent with Pavel Stoudek’s measurements. The other two subfigures show the same network, but they are trained only on specific configurations. The results are consistent across the configurations as well, so there is not much added value to these specific experiments. This does not mean all of the other references are correct, however.

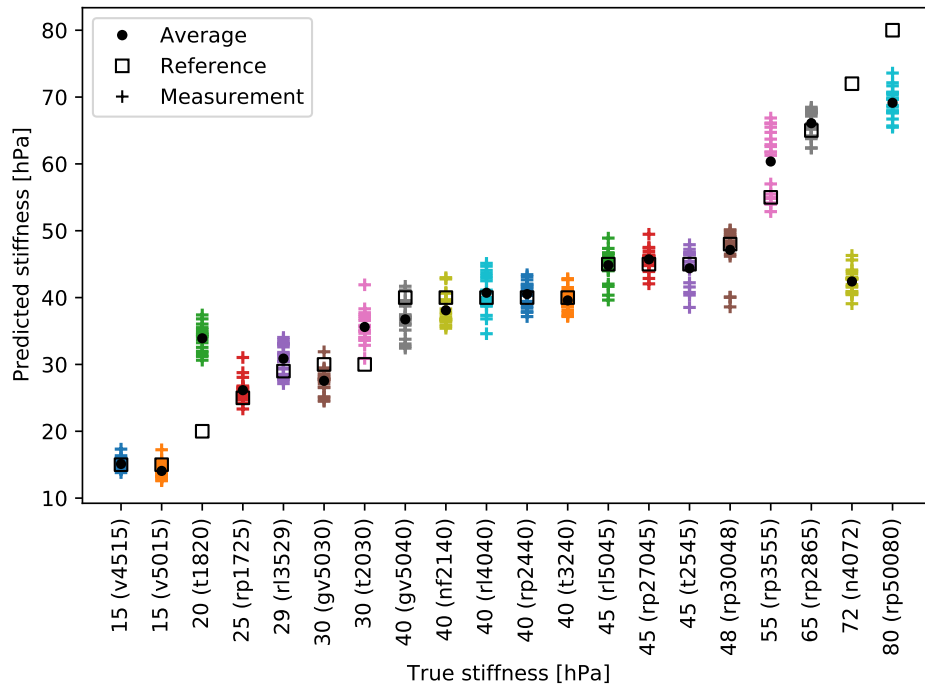
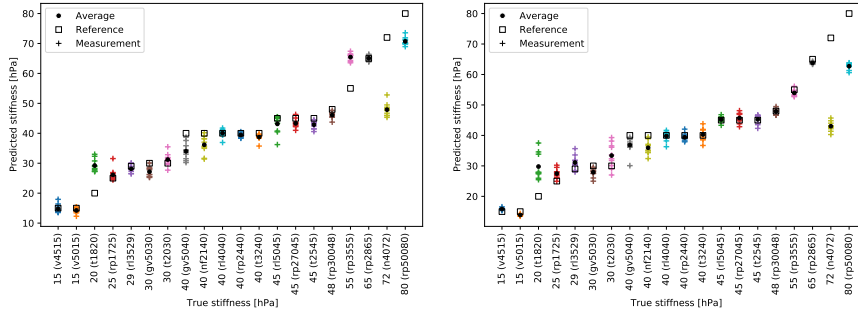


Figure 4.17: All foams in the dataset trained in elasticity regression network, epoch 200, both configurations, MAE 4.73.

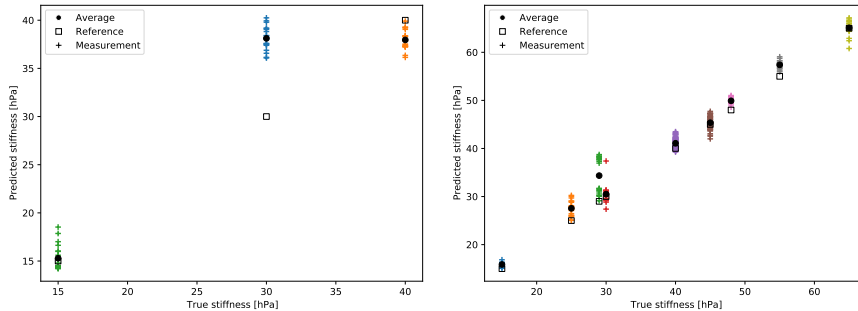


(a) : Only opposite finger configuration, MAE 4.17.

(b) : Only lateral finger configuration, MAE 4.40.

Figure 4.18: All foams in the dataset trained in elasticity regression network, epoch 200.

I executed the training for the elasticity regression network again, but this time without the N4072, T1820 and RP50080, results are in Figure 4.19. Unfortunately, no noticeable improvement can be seen in the validation set, although the training set noticeably improved. A possible reason could be that other foams have incorrect reference values as well or that the overall diversity of the dataset is too low.



(a) : Validation.

(b) : Training.

Figure 4.19: Dataset stiffregres_new, MAE 4.41.

4.5 Pressure reference calculation

An attempt to fix the incorrect references for polyurethane stated in Table 2.6 foams was made. According to the accompanying datasheets, the reference elasticity value is the pressure measured at 40% compression of the foam.

The calculation script with manual selection of the 40% compression is presented in `BarrettHand/preprocessing/pressure_calculation_demo.py`, [10]. The initial idea is as follows: let's call p_{ij} (N/cm²) the pressure of taxel j on tactile matrix i , S_{ij} (cm²) its surface area and F_{ij} (N) the force applied to it. Then we can calculate the total force F and total surface area

S as:

$$F = \sum_{i=0}^3 \sum_{j=0}^{23} p_{ij} \cdot S_{ij} \quad (4.1)$$

$$S = \sum_{i=0}^3 \sum_{j=0}^{23} S_{ij} \quad (4.2)$$

and the total pressure p is

$$p = \frac{F}{S}. \quad (4.3)$$

However, this only accounts for part of the contact area and contact forces between the object and the gripper. In other places, we can know neither the contact area nor the force applied. For example, foam GV5030 gives us 705 Pa instead of 3000 Pa. This is unlike any of the parallel grippers used in [25], where the contact area is the same throughout the grasp. To the best of my knowledge, there's no way of calculating the elasticity of an object.

4.6 Action selection

The action selection part of my work proved to be more difficult than expected. In most tasks, it is not clear which action performs better. I am going to list the cases where a significant difference in the performance between actions could be seen.

For density classification, configuration 3 at speed 0.3 rad/s performed with 100% accuracy, unlike configuration 1.

For classification, training data should be collected at a lower speed. The neural network trained at a lower speed can be applied to the same objects at higher speeds while maintaining roughly the same accuracy.

For Young modulus estimation, I suggest using the torque data. There is much noise in the tactile data, not to mention the sensors are not calibrated. I attempted to correct their behavior by detecting the active taxels as described above, but no such thing had to be done with the joint torque sensors.

For objects with the smallest dimension longer than 70 mm, lateral finger configuration can not be used as the object will fall out. Use the opposite finger configuration instead.

Chapter 5

Conclusion, discussion and future work

5.1 Conclusion

My work focused on learning object properties from haptic exploration. First, I defined action primitives for the used gripper as a tuple of finger configuration and speed. A ROS package was implemented to simplify the measuring process, conversion of data from ROS bag to NumPy array for later visualization. Over 1000 measurements were collected for different objects and organized into datasets for extracting different object properties. An LSTM neural network capable of handling variable-length time series was built for classification and material property (elasticity/stiffness, density) extraction purposes. Both the measurements and datasets are available in [24], the ROS package and LSTM network with its utilities are in [10].

Measured data were visualized using the written Python scripts. A method to detect active taxels and calibrate them was proposed and the new data were visualized again with a map of the active taxels. The approximate object compression plot was created with the use of the gripper's forward kinematics for lateral configuration.

Deviations in the reference values of the polyurethane foams were discovered and taken into account. Ablation studies showed that despite the noisiness of the tactile data, they are more important than the joint torque data for classification performance. I also proposed an idea, that when the classification network is trained on lower speed measurements, the model performs with similar accuracy on higher speed measurements as well without any modifications. This was repeatedly reproduced and, therefore, can be considered verified. According to the prior assumptions, it was discovered that non-homogeneous objects are difficult to classify. Where applicable, the preference of action for a specific task was introduced.

In addition to all of the above, I documented the Barrett Hand gripper; its drawbacks are further discussed in the next section. In collaboration with Bc. Pavel Stoudek, we collected numerous measurements of soft objects and,

through that, contributed to the creation of a dataset, which can be further studied.

5.2 Discussion and future work

In this section, I am going to discuss the results, limitations of the taken approach and possible future work.

One of the major drawbacks I had to deal with was the lack of reliable reference data, which is further discussed in [25]. Although the polyurethane foams we received included a datasheet with specific values, the allowed deviation of $\pm 15\%$ skews the data significantly in the case of object properties extraction (classification tasks were not affected). Neural networks can not be expected to perform well when training data are incorrect and even more so when their quantity is small. Arrangements were made to measure both the objects and PU foams accurately by Ing. Hynek Chlup, Ph.D., from the Faculty of Mechanical Engineering, FEE, CTU, but due to the Covid-19 outbreak and limitations during the quarantine, measurements were not finished before the deadline of the thesis. However, once the correct reference values are obtained, labels can be changed and training can be run again without making any modifications to the datasets.

I am aware that better local minima could possibly be found by fine-tuning SGD with momentum. However, due to time limitations and so many datasets to be evaluated, I did not spend time with that approach. Better accuracy could also be reached by decreasing the dimensionality of the data. In the results section, ablation was mentioned with the outcome that tactile data are more important than those from joint torque sensors. This only decreases the dimensionality by 3 (one joint torque sensor for each finger). Future research could try to combine raw data from taxels to reduce the dimensionality more significantly.

Polyurethane foams should be measured at different speeds. This was planned but was not completed due to time limitations and the fact that other parts of the thesis were more complicated than expected.

The action selection part of the work proved to be much more intricate than expected, as in most cases, the performance on the task was similar for the actions studied. A reinforcement learning algorithm could be used in the future to select the action. However, such a complex task needs a way to evaluate the results and, therefore, correct validation data. That was not possible because of the reasons mentioned.

The model-free approach taken in this work enables us to use raw data directly from the sensors. On the other hand, due to the Barrett Hand's uncalibrated taxel readings and even possible temperature effect on them, the network may have to be trained again for specific hardware, possibly even for another Barrett Hand. Retraining includes collecting the training data

and creating the datasets. Pre-processing the data, calibrating the sensors, taking temperature drifting into account or all of those as mentioned above could lead to better transferability between the hardware of the same kind.

During the experiments, I focused on homogeneous objects with simple geometry, such as cubes and rectangular prisms. Future work should take into account more complicated geometric shapes, including numerous prisms, balls and even non-homogeneous objects.

Finally, when estimating stiffness/elasticity from measurements made with Barrett Hand, torque sensors are less noisy than taxels and more similar to the plots from other grippers mentioned in [25]. However, Barrett Hand gripper is not at all suitable for scientific measuring of the objects because of the noise of the taxels and it's complicated kinematics. For example, we can guarantee neither the contact surface with the object nor reliably calculate objects' compression, which is crucial for precise material property measurements.



Bibliography

- [1] “Interactive perception-action-learning for modelling objects,” unknown (accessed May 15, 2020). [Online]. Available: <https://sites.google.com/view/ipalm/>
- [2] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, “Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey,” *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018. [Online]. Available: <https://journals.sagepub.com/doi/10.1177/0278364918779698>
- [3] J. Bednarek, M. Bednarek, P. Kicki, and K. Walas, “Robotic touch: Classification of materials for manipulation and walking,” in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2019, pp. 527–533. [Online]. Available: <https://ieeexplore.ieee.org/document/8722819>
- [4] M. Hoffmann, K. Štěpánová, and M. Reinstein, “The effect of motor action and different sensory modalities on terrain classification in a quadruped robot running with multiple gaits,” *Robotics and Autonomous Systems*, vol. 62, no. 12, pp. 1790–1798, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S092188901400133X>
- [5] J. Felip, J. Laaksonen, A. Morales, and V. Kyrki, “Manipulation primitives: A paradigm for abstraction and execution of grasping and manipulation tasks,” *Robotics and Autonomous Systems*, vol. 61, no. 3, pp. 283–296, 2013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0921889012002217>
- [6] M. V. Liarokapis, B. Calli, A. J. Spiers, and A. M. Dollar, “Unplanned, model-free, single grasp object classification with underactuated hands and force sensors,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5073–5080. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/7354091>

- Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- [19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: <https://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735>
- [20] C. Olah, “Understanding lstm networks,” 2015 (accessed May 16, 2020). [Online]. Available: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [21] W. Falcon, “Taming LSTMs: Variable-sized mini-batches and why PyTorch is good for your health,” 2018 (accessed February 24, 2020). [Online]. Available: <https://towardsdatascience.com/taming-lstms-variable-sized-mini-batches-and-why-pytorch-is-good-for-your-health-61d35642972e>
- [22] devforfu, “A simple LSTM-based time-series classifier,” 2019 (accessed February 23, 2020). [Online]. Available: <https://www.kaggle.com/purplejester/a-simple-lstm-based-time-series-classifier>
- [23] M. Maghoumi, “Use PyTorch’s dataloader with variable length sequences for LSTM/GRU,” 2018 (accessed March 7, 2020). [Online]. Available: <https://www.codefull.org/2018/11/use-pytorchs-dataloader-with-variable-length-sequences-for-lstm-gru/>
- [24] M. Mareš, 2020. [Online]. Available: https://drive.google.com/open?id=1d5TSQldJTEhYx9Pf_e7IOspK3o8gNfJ
- [25] P. Stoudek, “Extracting material properties of objects from haptic exploration using multiple robotic grippers,” 2020, master’s thesis published simultaneously with this work.

I. Personal and study details

Student's name: **Mareš Michal**

Personal ID number: **474428**

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:

Exploratory Action Selection to Learn Object Properties from Haptic Exploration Using a Robot Hand

Bachelor's thesis title in Czech:

Výběr průzkumných akcí pro robotickou ruku za účelem zjištění vlastností předmětů

Guidelines:

1. Familiarization with the Barrett Hand (BH8-282, 3 fingers, 96 tactile sensors, 3 fingertip torque sensors).
2. Pilot data collection - grasping/squeezing objects, focusing on soft/deformable materials.
3. Develop a repertoire of movement primitives for the hand aimed at acquiring different object properties (e.g., squeeze, slide, etc.). The actions can form hierarchies (e.g., there could be squeezing in different configurations, at different speeds etc.) or sequences. See Felip et al. 2013.
4. Employ selected visualization (e.g. PCA), clustering, or classification algorithms on time series of tactile and force sensors when manipulating the objects using actions from the repertoire. Assess whether material properties like stiffness can be extracted (Liarokapis et al. 2015).
5. Simulate that there is a goal to learn about specific object properties (e.g., stiffness, elasticity, surface roughness) and develop an action selection algorithm to choose the appropriate manipulation action (e.g., using reinforcement learning (McGovern et al. 2019) or information gain).
6. Create datasets of haptic exploration of the objects.
7. Discuss the results and identify the limitations of individual grasping primitives in discovering object material properties.

Bibliography / sources:

- [1] Bednarek, J., Bednarek, M., Kicki, P., & Walas, K. (2019). Robotic Touch: Classification of Materials for Manipulation and Walking. In 2019 2nd IEEE International Conference on Soft Robotics (RoboSoft) (pp. 527-533). IEEE.
- [2] Hoffmann, M., Stepanova, K. & Reinstein, M. (2014), 'The effect of motor action and different sensory modalities on terrain classification in a quadruped robot running with multiple gaits', Robotics and Autonomous Systems 62, 1790-1798.
- [3] Felip, J., Laaksonen, J., Morales, A., & Kyrki, V. (2013). Manipulation primitives: A paradigm for abstraction and execution of grasping and manipulation tasks. Robotics and Autonomous Systems, 61(3), 283-296.
- [4] Liarokapis, M. V., Calli, B., Spiers, A. J., & Dollar, A. M. (2015, September). Unplanned, model-free, single grasp object classification with underactuated hands and force sensors. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 5073-5080). IEEE.
- [5] McGovern, S.; Mao, H. & Xiao, J. (2019), Learning to estimate centers of mass of arbitrary objects, in 'Intelligent Robots and Systems (IROS), 2019 IEEE/RSJ International Conference on', pp. 1848-1853.

Name and workplace of bachelor's thesis supervisor:

Mgr. Matěj Hoffmann, Ph.D., Vision for Robotics and Autonomous Systems, FEE

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

Mgr. Karla Štěpánová, Ph.D., Robotic Perception, CIIRC

Date of bachelor's thesis assignment: **19.12.2019** Deadline for bachelor thesis submission: _____

Assignment valid until: **30.09.2021**

Mgr. Matěj Hoffmann, Ph.D.
Supervisor's signature

doc. Ing. Tomáš Svoboda, Ph.D.
Head of department's signature

prof. Mgr. Petr Páta, Ph.D.
Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his 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