Molly: a low-cost haptic robot

Laia, Tom, Òscar, Conrado, Pau, Claudia, Raquel, David

Abstract

This paper presents a novel approach for haptic object recognition using a three-fingered robotic end-effector. The proposed method extracts object curvature as a feature and compares the haptic samples acquired by the robot with those taken by a virtual robot in a simulated environment. The seamless integration of physical and virtual environments enables accurate object recognition and classification, which is crucial for a wide range of applications in robotics.

To evaluate the proposed approach, three different types of sensors have been employed to optimize the process of sampling and maximize the object recognition accuracy.

Finally, to demonstrate the adaptability of the approach in real-world scenarios, the designed robot has been utilized to take samples and compare them visually with the ones obtained by the virtual robot. The experimental results reveal that the proposed approach achieves high accuracy and can effectively recognize and classify different objects.

Overall, this research contributes to the advancement of haptic object recognition in robotics, and the proposed approach provides a promising avenue for future research in this area.

1 Introduction

In recent years, significant progress has been made in the field of Computational Mechanics and Engineering Science (CMES). One of the areas where this progress has been particularly notable is in the area of working with virtual figures. However, there is a need to move from the simulated to the real world, which requires the design of a low-cost haptic end-effector.

In this paper, we present an approach for haptic object recognition using a multi-fingered robotic end-effector. Our method involves extracting the curvature of objects as a feature and comparing the curvature samples obtained by the robot with those taken by a virtual robot in a simulated environment. To achieve this, various sensors are evaluated in terms of accuracy, delay and introduced noise, and the most suitable are selected for the analysis. Finally, we

demonstrate the effectiveness of our approach by comparing the results obtained on both real and virtual environments.

1.1 Related Work

Haptic shape recognition is a crucial practice in robotics that plays a vital role in accomplishing numerous tasks, including grasping and in-hand manipulation [16]. Various approaches that use unique tactile data have already been presented in the literature [1, 12, 7]. Depending on the input, these methods can be categorised into one of the following three groups: contact point distribution, pressure patterns in tactile arrays, and the combination of contact points and pressure patterns. A brief state-of-the-art of each method is presented to give an overview of the progress in haptic shape recognition.

1.1.1 Contact point distribution methods

Methods based on contact points restrict the geometry of the object and globally determine its shape by using the spatial coordinates of the robot's end-effectors at the time of contact [16][22].

In this context, there are several different approaches to haptic object recognition. One of them, as described in [6], uses a method based on point clouds, where a point cloud is a set of data points in a three-dimensional space representing the surface of an object. In this method, an anthropomorphic robotic hand touches the object and collects data points, which are then analysed using statistical point cloud properties to identify the object. However, when an object is touched up close, the resulting point cloud can be sparse (i.e. there are fewer data points) and deformed (i.e. the shape is not accurate). These challenges are specific to haptic object recognition and must be overcome to improve the accuracy of the method.

Another interesting approach uses a descriptor for categorizing tactile 3D objects [22]. Since it simply requires the relative geometry of points on the object surface, it is easy to design and invariant to object movement. In a physics-based simulation, it demonstrates how gripper closures utilizing a robotic hand with sparse tactile arrays can produce contact clouds that resemble the geometry of the object. Using information gathered by a haptic hand, it demonstrates the descriptor's capacity to distinguish between several item instances on a real robot.

While these methods are effective in limited situations when objects tend to be permanent and immobile, they can be time-consuming when numerous contacts are required to determine the overall shape of the object. Furthermore, the concatenation of transformations involving the locations of the contact points and the configuration of the hand position, which are known to be error-prone, has a significant impact on the precision of the system [6].

1.1.2 Pressure patterns in tactile arrays methods

The development of high-resolution tactile arrays has led to numerous approaches based on tactile patterns. The measurement of contact forces or the deformation of a soft elastomer skin can serve as the basis for tactile sensors that can recover pressure patterns. The latter enables the processing of computer vision descriptions that can be applied to tactile patterns as images. For example, the use of an optical tactile sensor with high resolution and constrained shape such as GelSight [21] has effectively aided numerous difficult robotic tasks [3].

Gelsight is a transparent elastomer slab with a reflective coating membrane covering it. This component is highly detailed in [20]. An application of this technology in haptics is a new GelSight design for robot gripper [3], which considerably improves geometric precision while maintaining a compact size.

This particular form of tactile sensor still has a number of challenges to overcome, such as the expensive processing procedures needed to manipulate its high dimensional raw output [13]. Additionally, when the data is mapped into a feature space with a lower dimension, it loses some of its physical meaning, which is another challenge that needs to be addressed.

1.1.3 Combination of pressure patterns in tactile arrays and contact point distribution methods

Combining the two tactile sense modalities is beneficial for object recognition in robots, leading to more reliable procedures[17] [9]. Studies have presented unique algorithms, such as Iterative Closest Labeled Point (iCLAP)[9]. This one assigns unique label numbers to local tactile features, creating a 4D point cloud of the object using the label numbers and corresponding 3D coordinates. The partial 4D point cloud from multiple touches is compared to reference models repeatedly to find the best fit for recognition.

Another study [17] uses a basic underactuated robot hand equipped with low-cost pressure sensors to perform tactile object identification and feature extraction. It also uses complementary methods based on parametric estimation and random forests machine learning, which can operate independently or jointly to enhance recognition. The method works for arbitrary object positions and orientations, and it fits for real-world scenarios without adding labored processing or manipulation overhead.

However, these methods can be resource-intensive and challenging to implement, prompting some researchers to turn to robotic simulators to minimize the need for physical trials. Thus, it is beneficial to explore the development of a more streamlined and efficient haptic acquisition system, where the collected data is simpler to process.

Our study introduces a new haptic acquisition system that can explore objects in a simple way, with a tactile end-effector using only three numerical values. These three values can be converted into a single one while retaining some of its geometrical details, and by combining our geometric properties we can produce a distribution that characterizes the geometry of an item.

1.2 Curvature as a feature for capturing data

In our method for haptic object recognition, we focus on a parameter related with curvature as the primary feature for classifying objects. To capture curvature information effectively, we have designed our end-effector in a specific configuration that facilitates the extraction of curvature data.

Our end-effector consists of three fingers evenly spaced and aligned, each equipped with contact sensors for sampling. The fingers explore a stimulus by moving at a constant velocity in the normal direction of its surface until they sense a collision. The collision event starts when a finger makes the first contact with the object, and the system records the time interval between this initial contact and the subsequent touches by the remaining fingers. As a result, we get a feature with three values, (t_1, t_2, t_3), where each t_i represents the time elapsed from the first contact sensor and the contact sensor i. This means that the initial sensor's contact time is always 0, which becomes the reference time for subsequent sensors. For a visual representation, refer to Figure 1.

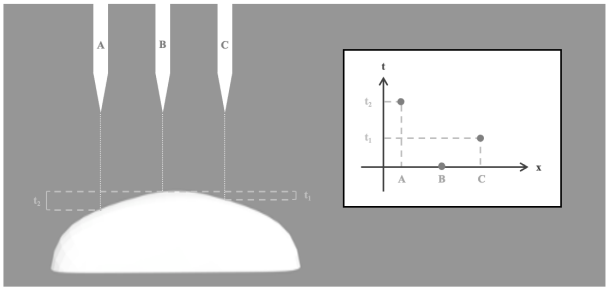


Figure 1: Representation of an end-effector contact event. The red dots symbolize the three touch sensors, while the blue area represents the stimuli. The graph in the box (top right) displays the relationship between the sensors' positions and the time intervals until surface contact. Upon contact with the stimulus surface, the system generates the temporal feature, described as (t_1, t_2, t_3), where $t_2 = 0$ and $t_1 < t_3$.

The system receives this three-position vector containing the contact times of each finger, and computes the local curvature with an approximation of the second derivative. Thus, the curvature from the object being explored is computed from the time differences in the form of a discrete function, calculated as follows:

$$(t_1 - t_2) - (t_2 - t_3)$$

It is important to acknowledge that two data capture systems configured identically may operate at varying speeds, leading to an inconsistency in time comparison. Nevertheless, despite the difference in operational speed, both systems cover the same distance when sampling identical objects. When comparing times obtained by two systems A and B sampling at different speeds, we need

to use a multiplying factor. This factor is the relation between the speeds of the systems, which can be expressed as:

$$k = \frac{v_A}{v_B}$$

Using this factor, we are able to convert time vectors from system A to the other system's reference (likewise, the process applies vice versa):

$$\vec{t_B} = k \cdot \vec{t_A}$$

This process yields a time vector that can be directly compared with the time vector obtained by the system with which we wish to compare.

2 Robotic end-effector

The robot consists of two parts: the robotic arm and the end-effector, which is attached to the former. As we describe in the next section, we use a commercial robotic arm. This arm is responsible for guiding the end-effector to interact with the stimulus under exploration. On the other hand, as suggested in the previous section, we have developed the end-effector from scratch.

In this section, we describe in detail the robotic set in its entirety, both in terms of hardware and software. First, we briefly present the commercial robotic arm that was used for the project. Then we outline the hardware implementation of the end-effector and the various sensors that were tested. Finally, we discuss the implementation of the software that controls both the end-effector and the robotic arm.

2.1 Robotic arm

The robotic arm used is the model UR3 from Universal Robots [18], which allows the end-effector to move and thus sample the objects. The UR3 robot features six rotating joints that users can control and manipulate, but our approach only utilizes two of these joints. The overall movement of the end-effector is vertical and orthogonal to the surface of the table, at a velocity of 6.4 cm/s. The computer is connected to the robotic arm via sockets, with one IP and one port, so we can control the robot and send commands from our computer.

To capture the samples, we have developed a simple code to achieve correct arm movement. The robot is initially at a fixed resting position, where each of its joints has a pre-defined orientation in order to configure the initial coordinates of the whole arm. Then, once the order of sampling is sent, it starts to descend vertically. It will only stop under one of the following circumstances:

(A) The three sensors have been triggered as a consequence of them touching the surface of the object, obtaining a valid sample.

(B) If the descent continues and exceeds the predetermined limit distance of 10 centimeters (approximately the vertical distance between the configured

resting position of the end-effector and the table). On that occasion, the sample will be discarded.

(C) In case of a protective stop activation from the robot controller interface.

In the first two cases (A) and (B), the robot goes up to the resting position again until an order of going down arrives. In case (C), we would have to manually reactivate the robot.

2.2 Hardware

In this section we describe the physical parts of the system. We divide the description of the hardware into three parts: the structure of the end-effector, the sensors of the fingers, and the microcontroller that contains the logic of the sampling.

2.2.1 End-effector Structure

The whole structure has been designed with SketchUp (for Web) and subsequently has been printed with a 3D printer using PLA. There are three main parts in the hand structure: the box, the fingers, and the lid. For extended details on their design specifications, refer to Appendix A.

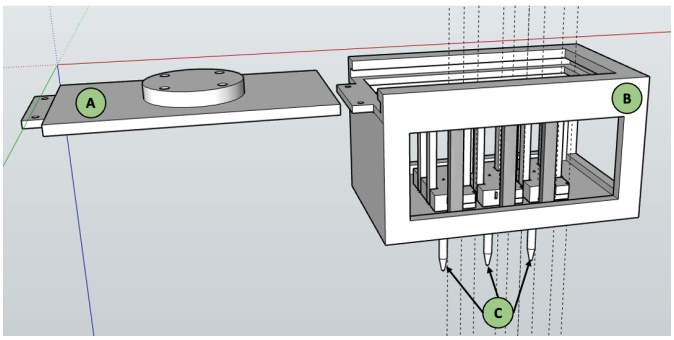


Figure 2: Whole-hand structure. (A) Lid, (B) the box, and (C) the fingers.

2.2.1.1 Box lid

To be able to join the end-effector structure to the robotic arm, we need an adapter that allows the structure to be screwed to the end of the robotic arm. To do this, we design the box with a sliding top. This lid has also a tab with the same holes and the same dimensions as the box tab to prevent the lid from slipping. This ensures that the lid and the box are securely fitted. For more details on the box lid, refer to Appendix A.3.

2.2.1.2 Box's base

We have designed the box so that it can house both the electronic components and the three fingers. It also accommodates various sensors which have undergone evaluation. Consequently, the mechanics and dimensions of these sensors are relevant constraints in the design of the box, as they need to be strategically placed to ensure optimal sampling.

The box has four windows, three on the sides and one at the top, which are necessary to facilitate the manipulation of the circuitry and sensors. Also, it has two threaded holes that allow the lid and the box to fit perfectly with the help of two small screws, as shown in Figure 2.

On one side of the box, as seen in Figure 3, we have the optical sensors. On the other side of the box, as seen in Figure 4, we have both the linear variable resistors and the micro switches. Each one of the sides has sensor separators for their correct fastening (see B's in Figure 3 and in Figure 4). On the linear variable resistors and micro switches side, there is a step of 1 mm so the micro switch can rest (see D's in Figure 4).

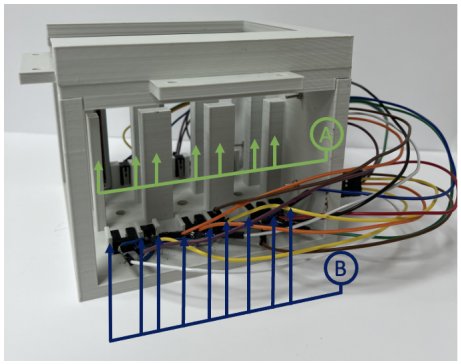


Figure 3: Image of one of the box's sides, the one that contains the optical sensors. A are the guide rails and B are the sensor separators. Between these sensor separators, we find the optical sensors.

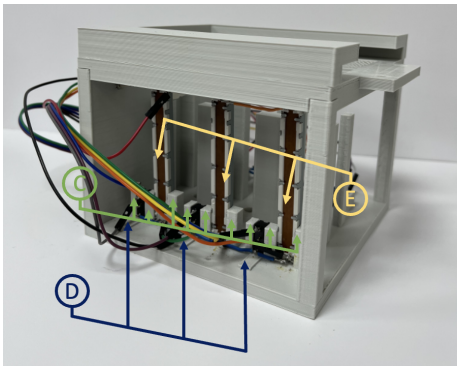


Figure 4: Picture of the other box's side, containing the variable resistors and the microswitches. C are the sensor separators and D are the 1 mm steps for the microswitches. The microswitches are the black components between the sensor separators, and E are the variable resistors (the three columns).

2.2.1.3 Fingers

The design of the fingers also has been conditioned by the shape and operation of the sensors. For detailed design specifications, refer to appendix A.1.

2.2.2 Sensors

We have evaluated three different sensors for our end-effector: optical sensors, micro switches and linear resistors. The last two are directly connected to the microcontroller, while optical sensors need an adaptation circuit for its operation. We have used one input pin of the microcontroller for every sensor, meaning a total of nine pins (three different sensors for each finger). Refer to Figure 5 for an overall diagram of the hardware.

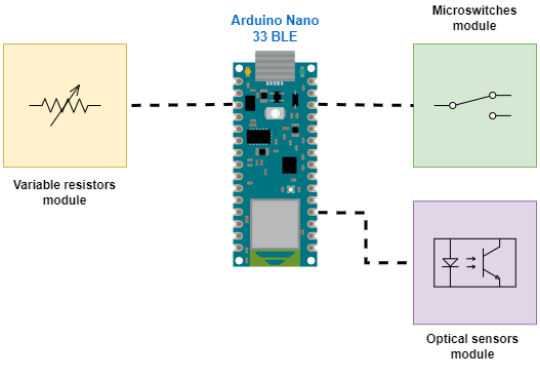


Figure 5: Simple block diagram of the hardware design, where the microcontroller monitors all sensors' modules.

2.2.2.1 Optical Sensors

The optical sensor model used is the TCST1103 from Vishay Semiconductors [19], a through-beam type of sensor that consists of one LED, the transmitter, and a phototransistor that works as a receiver. Both the transmitter and the receiver are placed in opposite directions to each other.

The LED projects a light beam onto the phototransistor, which acts as a receiver. When the light beam is interrupted, the phototransistor detects the change and the collector current decreases. If a resistor is placed in series with the phototransistor, a voltage is generated.

To convert the analog output of the sensor to a digital signal, we use the comparator mode of the operational amplifiers. This approach is necessary because we aim to use the interruptive pins of the board, which require a digital input. Thus, this sensor needs the support of an electronic circuit for its operation, composed of a 10 kΩ potentiometer and an operational amplifier[10]. The operational amplifier operates as a comparator, and the potentiometer allows us to set the voltage threshold at which the comparator switches. The circuit implemented is included in Figure 6.

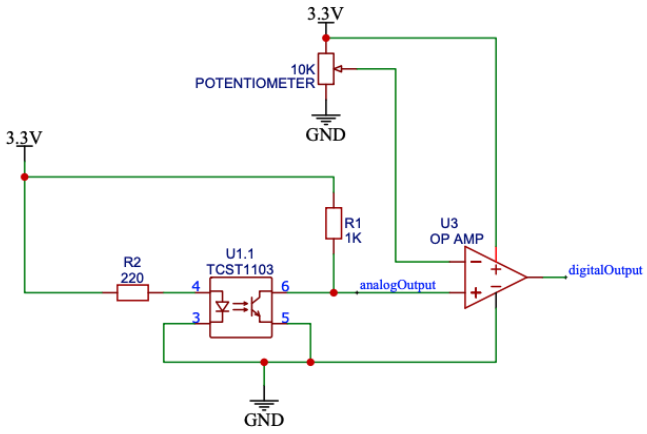


Figure 6: Electrical schematic of each optical sensor, illustrating the connection to an operational amplifier. The sensor's output signal is compared to a threshold set by a potentiometer at the other input terminal of the amplifier. When the sensor signal exceeds the threshold, the amplifier outputs a logic high (3.3V); otherwise, it outputs a logic low (0V).

In resting position, the protruding extension is blocking the passage of light to the receiver, so there's no collector current and the digital output is high. Once the extension stops obstructing the passage of light, the collector current increases and the analog output decreases. Consequently, the output of the operational will be digital low. Therefore, to perceive whether the finger has touched any surface with this sensor, we will have to detect a tension falling

edge.

2.2.2.2 Micro Switches

The micro switch model employed is the D2F-01FL-T from Omron [11]. It features three pins: one common, one connected when pressed and one connected when not pressed. The micro switch operates by changing the position of its contacts when it comes into physical contact with a surface.

This sensor gives a digital output, thus it is connected directly to one of the digital interruptive pins of the board with no need of extra circuitry. The pin of the sensor we use as output is the Normally Connected (NC), so in resting position the micro switch is pressed and gives us a value of digital high (logic '1'). When the finger is lifted up due to the contact with the object's surface, the micro switch will be deactivated and will output a value of digital low (logic '0'). Thus, similarly to the optical sensor, a tension falling edge must be detected when the finger comes into contact with the surface of the object.

2.2.2.3 Variable Resistors

The linear variable resistor used is the model RS6011Y1401A from Alps Alpine [14]. It is a 50kΩ variable resistor, with 20% tolerance and connected to 3.3V, powered by the board and ground.

The linear variable resistor has a total length of 75 mm, but only 60 are available for the movable lever's sliding in our design. When the lever is at one end of the resistor, the value of the resistor is close to 50kΩ and the voltage output is low. Conversely, when the lever is at the other end, the value of the resistor is close to 0 Ω and the the voltage output is close to 3.3V.

This sensor gives also an analog output, but unlike the others it is not converted to digital. Instead, the analog output from this sensor is routed to one of the analog pins on the microcontroller board. In order to convert the analog signal into a digital value that can be processed by the microcontroller, the board's ADC (Analog to Digital Converter) is used. The ADC converts the analog signal to a digital value with a resolution of 10 bits, meaning that the digital output can have a range of 1024 possible values (2^{10}-1).

Considering we have 60 mm of sliding and a resolution of 10 bits from the ADC, the resolution in terms of distance is $60\,\text{mm}/2^{10} = 0.05859375\,\text{mm}$.

2.2.3 Mechanical evaluation of sensors

In order to determine the optimal sensors for samples' acquisition with our robot, an assessment of the mechanical performance of the different sensor types is conducted. This evaluation is based on some of their characteristics, which are detailed in this section.

Firstly, the variable resistors meant slow sampling due to the need for manual manipulation after each sample. However, they could serve as a viable option if the finger material possessed enough weight to return these resistors to their

initial resting position after each sample autonomously, without requiring user intervention.

On the other hand, the microswitches showed problems with activation due to slight movements of the sensors. The fingers pushing the sensors backward while taking samples can result in activation loss, requiring a restart of the sampling process, thereby affecting calibration offsets. Thus, it can be concluded that these sensors are not practical for our implementation.

Finally, the optical sensors emerged as the best option, as they do not require physical contact and do not show the frequent activation issues observed with microswitches. Additionally, they do not need frequent recalibration and the gravity force enables the fingers to descend after every sample without requiring user intervention.

Consequently, the optical sensors have been the ones employed for the analysis and comparison between the real and the virtual environments.

2.2.4 Microcontroller

The microcontroller used is the Arduino Nano 33 BLE [2]. It has an operating voltage of 3.3V, a clock speed of 64MHz, a CPU flash memory of 1 MB, and fourteen digital pins which can work as external interrupts.

The microcontroller is powered through the USB Micro cable connected to a computer, and it utilizes nine pins for sensor connections. Digital interrupt pins are used for the optical sensors and the micro switches, while analog pins are used for the linear variable resistors. Data is transmitted from the microcontroller to the computer through a UART cable.

2.3 Software

This section provides an explanation of the two scripts utilized in implementing the system's logic, both the microcontroller and main script.

The main software script in our system is written in Python and runs on the computer. It provides the user with a menu that contains all available options for controlling the robotic arm, and is responsible for the communication with both the robot and the microcontroller. It communicates with the robot through sockets, which is a type of network communication protocol, and it communicates with the microcontroller through the serial port.

On the other hand, the microcontroller script is responsible for the data acquisition logic, where the sensors connected are monitored when interacting with stimuli. The samples taken are sent through the serial port to the main script located in the computer, which stores them in a text file for later processing and analysis.

For the serial communication between the microcontroller and the computer, a protocol of orders has been implemented so that computer's script can indicate the microcontroller what action to take. Figure 7 shows a summary of the different commands used.

Command (*X*, *a*)	Outcome
L,0	Turns the micro's led on.
L,1	Turns the micro's led off.
H, -	The *a* value is not relevant for this command. It will send a "Moly says: Hello World!" through the serial port.
O, t	The system takes a t number of samples.
A, -	The *a* value is not relevant for this command. The system takes samples until the user sends a Ctrl+C.
E, -	The *a* value is not relevant for this command. It exits the program.

Figure 7: Summary table of the different commands that the main script can send to the microcontroller script for the correct operation of the system.

2.3.1 Microcontroller script

We have developed two scripts for the microcontroller: one for using the micro switches or the optical sensors and another for the linear resistors. Both scripts store the data of the individual samples in an array with three positions.

2.3.2 Main script

This script controls the robotic arm and thus the movement of the whole system, while communicating at the same time with the microcontroller to obtain the sampled data. To achieve the movements for data sampling, the script has two main functions: "moveJ" to configure an exact position by indicating the coordinates in radians of the six joints that conform the robotic arm, and "moveZ" to rotate the robotic arm in the Z axis a certain amount of degrees.

The script has been developed in Python 3 and has a menu with six functionalities, each of them associated to a command that the computer sends to the microcontroller as shown in Figure 7.

2.3.3 Preprocessing data

When samples are to be processed to obtain results, it is important to ensure uniformity of physical quantities and units across all samples before they are analysed. In the case of our study, our pre-processing pipeline consists of two main steps: calibration and data conversion.

The calibration technique used to mitigate system offsets caused by sensor irregularities or inherent inaccuracies involves capturing 100 samples from a flat surface to determine the deviation for each finger, which is then subtracted from subsequent object samples. The automation is done via a Python 3 script that uses the determined offset values.

On the other hand, data conversion ensures standardisation when comparing samples in different systems. This involves, for example, the conversion of units between time and distance, which is particularly important for sensors such as linear variable resistors. Another Python 3 script facilitates this conversion process.

3 Virtual robot capture system

A virtual robot capture system was developed to efficiently collect a larger number of haptic samples for training[4]. The virtual robot was modeled based on the specifications of a real robot. It employs a ray tracing technique to simulate the movement and collision of the robot's fingers with the 3D objects. The system features a three-finger configuration with adjustable speeds, allowing for flexibility in sampling. Furthermore, the software allows the change to the time and spatial measurements between different systems, facilitating comparisons between the virtual and real robots even when their specifications do not align perfectly.

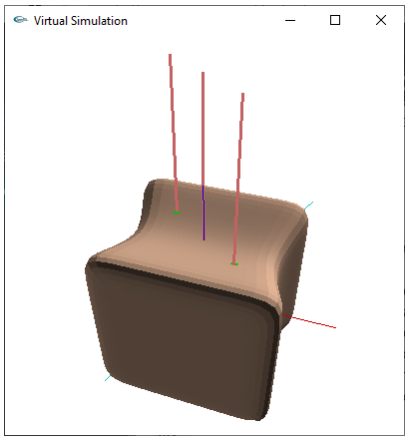


Figure 8: Screenshot of the Virtual robot capture systems where the rays represent the path of the fingers of the robot towards the object.

To improve the realism of the virtual samples, Gaussian noise can be introduced to reflect the imperfections present in real-world robotics due to hardware and software limitations[15]. The virtual dataset consists of digitally modeled figures with ideal shapes, which are also 3D printed using PLA for real-world testing. While the virtual and physical models are proportionally similar, the 3D printing process introduces certain inaccuracies, such as layer thickness, which prevent the real figures from achieving perfect curvature.

4 Stimuli and sampling

4.1 Stimuli

As we have already mentioned, our robot is trained with a virtual system. Therefore, we will use the same stimuli in both environments, i.e. we have created a set of visual (digital) stimuli to train the system with and then 3D printed them to evaluate with real data.

The creation of the stimuli is done with the tool SuperShape [4]. Five surfaces were designed where only the curvature parameter was varied to validate if our system really captures curvature differences. There are two concave surfaces, two convex surfaces and one flat surface (see Figure 9).

These stimuli were also 3D printed with PLA to test them with the real robot. The dimensions of these stimuli are 16.5 cm along the longest side of the base, maintaining the aspect ratio.

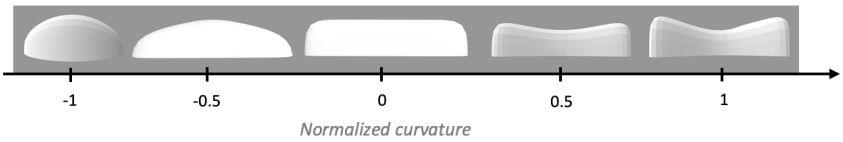


Figure 9: 1-axis representation of the final dataset figures'

4.2 Virtual and real sampling

Our virtual and real systems must take haptic samples of the stimuli. The virtual samples are used to train the artificial agent and the real samples to evaluate whether this agent is able to classify them. he samples are taken according to a 45-degree step pattern, which simply consists of taking a measurement every 45 degrees, as shown in Figure 10.

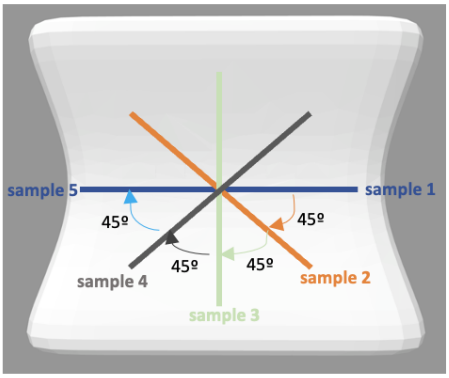


Figure 10: Image that represents how samples are taken on stimuli in both real and virtual environments. For each round, a total of 8 samples are taken.

A first look at the data reveals a crucial difference in the nature of the systems: The samples of the virtual system have no noise, i.e. two samples taken at the same point have the same value, while the real system has noise due to sampling inaccuracies and sensor imprecision. As shown in Figure 11, virtual histograms represent samples that are exactly centred on certain values, without any scatter.

To enable a comprehensive comparison of the samples from both environments using Probability Density Functions (PDFs), we introduced Gaussian noise with a variance of 0.04992 into the virtual sampling process. This adjustment was made to align the virtual measurements more closely with those from the real environment. The variance value was determined through an analysis aimed at maximizing the similarity between real-world and haptic samples, with this specific value yielding the highest accuracy in the recognition model used for training. The PDF comparison between the noisy virtual and real environments is included in Figure 11.

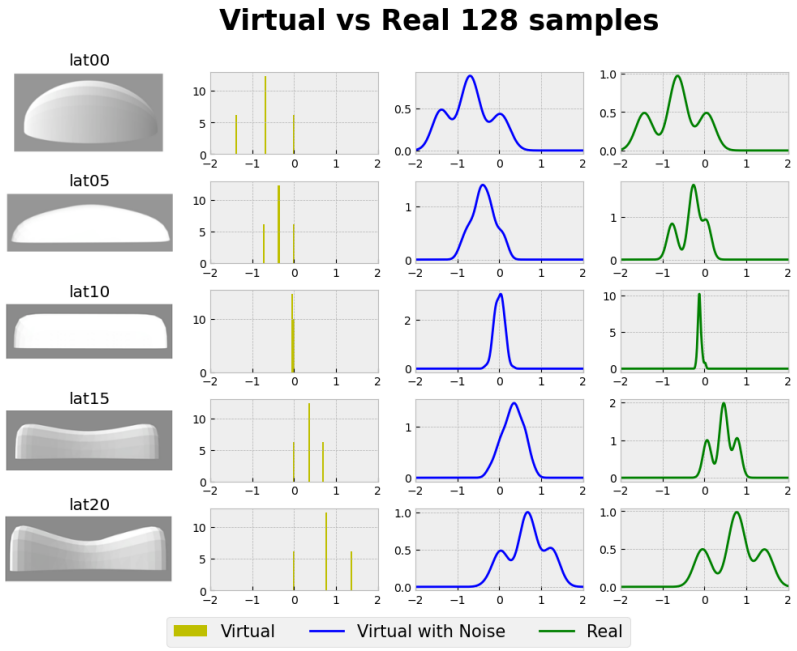


Figure 11: Probability Density Functions (PDFs) and histograms of all stimuli from the dataset (both in real and virtual world). The figure illustrates several key aspects of the PDFs and histograms. The peaks of the PDFs represent the average values of different sample groups. Three main sections are visible: flat samples centered at 0, convex samples centered on positive values, and concave samples centered on negative values. The width of the peaks reflects sample noise, with virtual samples showing wider peaks due to higher noise levels compared to real-world samples.

5 Haptic classification

In order to classify haptic data, our approach is to train a simple learning model on the obtained noisy virtual PDFs and test the model on real samples.

The algorithm used to train the model is a Probability Density Function Based Classifier [5], where the noisy digital PDFs of Figure 11 are the ones used as reference for prediction. To measure the match of one test probability distribution, the Kullback-Leibler divergence [8] measures the difference with all the PDFs of the stimuli. In order to predict the object associated to the real test samples, the algorithm measures the divergence between the test PDF and the 5 virtual train PDFs. The predicted stimulus is the one whose divergence is minimum, i.e.,

$$\widehat{y} = \underset{y \in [1,5]}{argmin}[D_{KL}(P||Q_y)] \tag{1}$$

where P is the probability distribution of the test curvatures and Q_y is the training PDF corresponding to stimuli y.

As depicted in Figure 12, the recognition accuracy of the model increases when the amount of test samples used for prediction is greater.

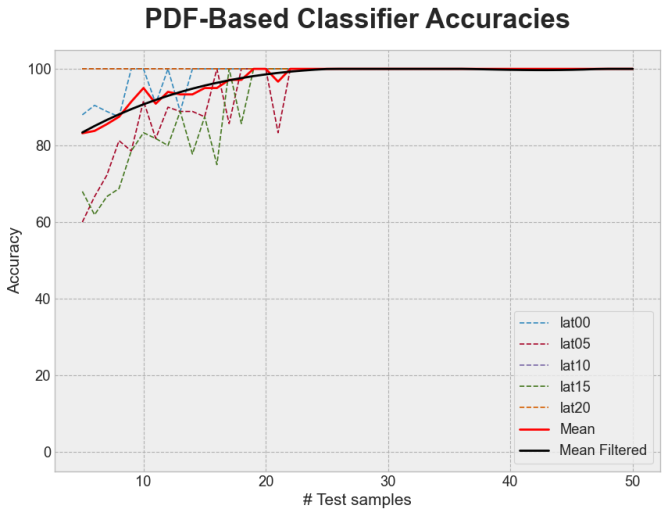


Figure 12: Graph that represents the stimuli recognition accuracy of the Kullback-Leibler Classifier depending on the amount of text samples used for prediction. The graph included an accuracy curve for each stimulus, a global accuracy curve and a smoothed global accuracy curve. When we increase the amount of tests samples used for a stimulus' prediction, the recognition accuracy gets bigger.

Apart from the model's accuracy, it is also interesting to evaluate the confusion matrix to determine the capacity of the model to recognize each stimulus. In Figure 13, a confusion matrix obtained with groups of 10 test samples is provided, showcasing a global accuracy of 95%.

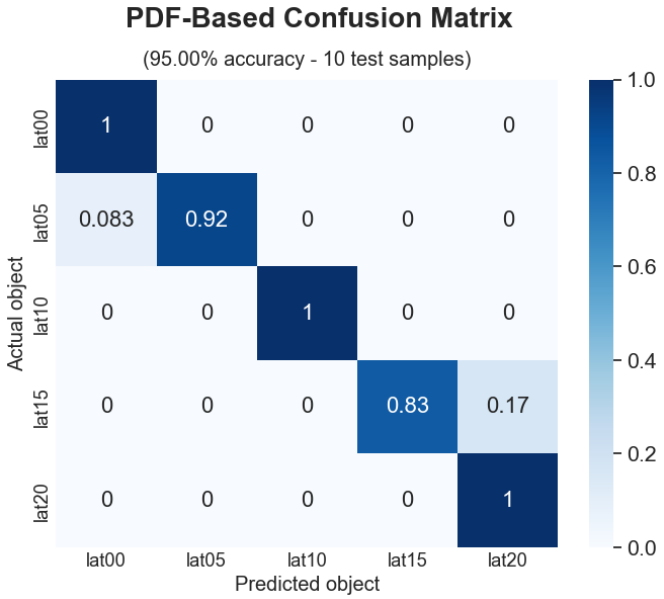


Figure 13: Confusion matrix that represents stimulus recognition accuracy of the Kullback-Leibler Classifier for each of the dataset stimuli when the tests are conducted with 10 test samples of the stimulus.

A Hand structure design

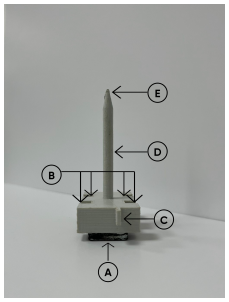
A.1 Fingers' design

The fingers have a rectangular base with a 30 mm side, a 60 mm side, and a height of 12 mm, and its base has two 9 per 6 mm orifices on each side (see B in Fig.15) so the fingers fit into the guide rails. Also, a rectangular hole of 2 per 5 mm and 16 mm depth has been included on one side so the bar of the linear variable resistor can be inserted into the finger and follow its movement.

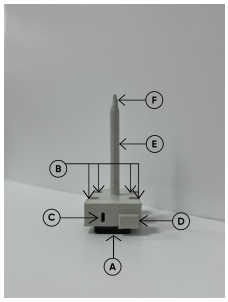
Next to this cavity, there is a rectangular protruding extension of 6 per 10 per 4 mm dimensions responsible for pressing the micro switches (see D's in Fig.14a and in Fig.15). On the other side, there is also a protruding extension that contributes to the activation of the optical sensor (see Cs in Fig.14b and Fig.15).

Sticking out of the center of the rectangular base, we find the body of the circular finger, see. The radius of the body is 3 mm, the finger is shown in D of Figure 18, and E of 14a and Fig.15. The radius of the fingertip is 1 mm, the fingertip is found in E in Fig.14b and in F in Fig.14a and in Fig.15.

It is important to highlight that five iron weights, of 5 grams each (a total of 25 grams), have been affixed to the rectangular base of each finger (refer to A's in Figures 14a, 14b and 15). This addition is intended to facilitate the finger's descent and return to its initial resting position by enhancing the force of gravity.



(a) Photo of the finger's side that faces the optical sensor. A are the 5-gram weights, B are the side orifices for the guide rails, C is the protruding extension to activate the optical sensor, D is the finger and E is the tip of the finger.



(b) Photo of the finger's side that faces the variable resistor and the microswitch. A are the 5-gram weights, B are the side orifices for the guide rails, C is the orifice in which the variable resistor's lever is inserted, D is the protruding extension to activate the microswitch, E is the finger and F is the finger's tip.

Figure 14: Different perspectives of one of the three fingers used for the system.

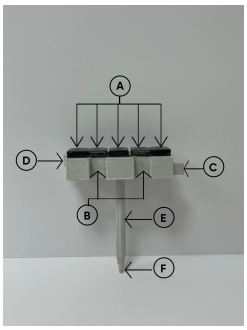


Figure 15: Picture of one of the sides of the fingers. A are the 5-gram weights, B are the orifices for the guide rails, C is the protruding extension.

A.2 Base of the box design

In the base of the box there are three equidistant holes with a 3 mm radius for the correct sliding of the fingers (see B's in Fig.3), and it also contains eight guide rails that go from the base to a height of 60 mm (see C's in Fig.16 and A's in Fig.3) so the fingers follow a straight vertical trajectory perpendicular to the surface without wobbling.

The base of the box was partitioned into three identical sections, as illustrated by the A's in Fig. 16 and Fig. 17. Each section contains the same structure and accommodates one of the three fingers, with a 3 mm radius hole located at the center through which each finger will pass.

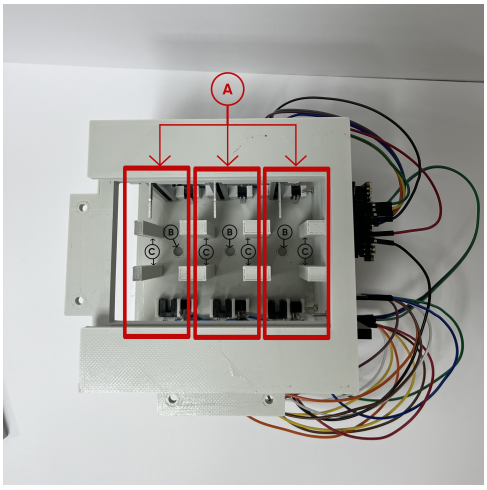


Figure 16: Photo from the top of the box, the fingers have been removed so the base of the box can be seen. A represents each of the sections that host each of the 3 fingers, B are the 3mm equidistant holes and C are the 60mm guide rails.

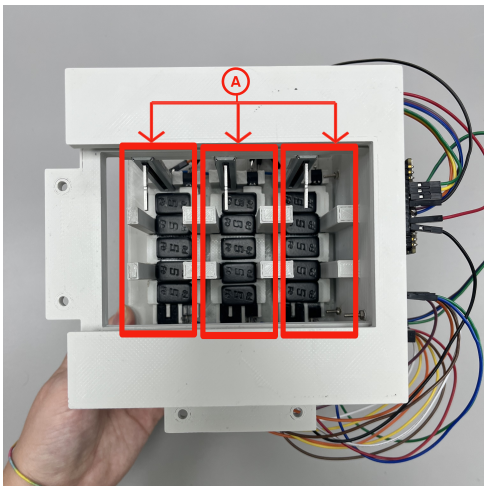


Figure 17: Photo from the top of the box, with the three fingers in it. A are the three sections that host each one of the fingers.

A.3 Box lid design

The sliding lid designed has a rectangular base and a 32 mm radius circumference in the middle with four equidistant 3 mm radius holes which have their centers on a concentric circumference of 25 mm radius (see A's in Fig.18). These four holes go through the 17 centimeters of the lid and will allow the screws to pass through in order to screw the hand structure to the robotic arm.

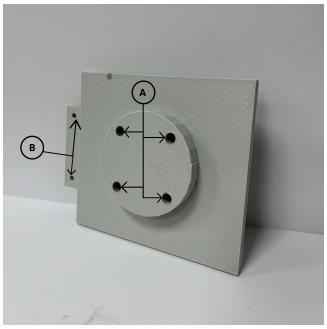


Figure 18: Picture of the box's lid, where A are the four 3 mm holes and B are the holes in the lid's tap, that allow us to fit the box and the lid.

References

[1] Peter K Allen and Kenneth S Roberts. Haptic object recognition using a multi-fingered dextrous hand. Technical report, Columbia University, 1988.

[2] Arduino, 2022.

[3] Siyuan Dong, Wenzhen Yuan, and Edward H Adelson. Improved gelsight tactile sensor for measuring geometry and slip. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 137–144. IEEE, 2017.

[4] Guillem Garrofé, Carlota Parés, Anna Gutiérrez, Conrado Ruiz, Gerard Serra, and David Miralles. Virtual haptic system for shape recognition based on local curvatures. In Nadia Magnenat-Thalmann, Victoria Interrante, Daniel Thalmann, George Papagiannakis, Bin Sheng, Jinman Kim, and Marina Gavrilova, editors, *Advances in Computer Graphics*, pages 41–53, Cham, 2021. Springer International Publishing.

[5] Guillem Garrofé, Carlota Parés, Anna Gutiérrez, Conrado Ruiz, Gerard Serra, and David Miralles. Virtual haptic system for shape recognition based on local curvatures. In Nadia Magnenat-Thalmann, Victoria Interrante, Daniel Thalmann, George Papagiannakis, Bin Sheng, Jinman Kim, and Marina Gavrilova, editors, *Advances in Computer Graphics*, pages 41–53, Cham, 2021. Springer International Publishing.

[6] Nicolas Gorges, Stefan Escaida Navarro, and Heinz Wörn. Haptic object recognition using statistical point cloud features. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 15–20. IEEE, 2011.

[7] Shan Luo Jiaqi Jiang. Robotic perception of object properties using tactile sensing. in tactile sensing, skill learning, and robotic dexterous manipulation. *Academic Press*, pages 23–44, 2021.

[8] Solomon Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[9] Shan Luo, Wenxuan Mou, Kaspar Althoefer, and Hongbin Liu. Iterative closest labeled point for tactile object shape recognition. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3137–3142. IEEE, 2016.

[10] Mouser electronics, 2022.

[11] Mouser electronics, 2022.

[12] Stefan Escaida Navarro, Nicolas Gorges, Heinz Wörn, Julian Schill, Tamim Asfour, and Rüdiger Dillmann. Haptic object recognition for multi-fingered robot hands. In *2012 IEEE haptics symposium (HAPTICS)*, pages 497–502. IEEE, 2012.

[13] Marsela Polic, Ivona Krajacic, Nathan Lepora, and Matko Orsag. Convolutional autoencoder for feature extraction in tactile sensing. *IEEE Robotics and Automation Letters*, 4(4):3671–3678, 2019.

[14] Rs components, 2022.

[15] Conrado Ruiz, Òscar de Jesús, Claudia Serrano, Alejandro González, Pau Nonell, Arnau Metaute, and David Miralles. Bridging realities: training visuo-haptic object recognition models for robots using 3d virtual simulations. *Vis. Comput.*, 40(7):4661–4673, May 2024.

[16] Harold Soh and Yiannis Demiris. Incrementally learning objects by touch: Online discriminative and generative models for tactile-based recognition. *IEEE transactions on haptics*, 7(4):512–525, 2014.

[17] Adam J Spiers, Minas V Liarokapis, Berk Calli, and Aaron M Dollar. Single-grasp object classification and feature extraction with simple robot hands and tactile sensors. *IEEE transactions on haptics*, 9(2):207–220, 2016.

[18] Universal robots, 2022.

[19] Vishay semiconductors, 2022.

[20] Robert J. Woodham. Haptic object recognition for multi-fingered robot hands. In *Photometric method for determining surface orientation.*, pages 139–144, 1980.

[21] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[22] Mabel M Zhang, Monroe D Kennedy, M Ani Hsieh, and Kostas Daniilidis. A triangle histogram for object classification by tactile sensing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4931–4938. IEEE, 2016.