COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
(ICAI)

MSc. Industrial Engineering

Vision-tactile object classification through Faster R-CNN and Machine Learning algorithms for robotic grasping

Author
Belén Castellote López

Supervised by
Silvia Tolu

Lyngby, Denmark
30 May, 2023

Belén Castellote López, declara bajo su responsabilidad, que el Proyecto con título **Vision-tactile object classification through Faster R-CNN and Machine Learning algorithms for robotic grasping** presentado en la ETS de Ingeniería (ICAI) de la Universidad Pontificia Comillas en el curso académico 2020/21 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Belén Castellote .... Fecha: ..07.. / ..06.. / ..2023..

Autoriza la entrega:

El Director del Proyecto

Silvia Tolu

Fdo.: [illegible] .... Fecha: ..08.. / ..06.. / ..2023..

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
(ICAI)

Máster en Big Data: Tecnología y Analítica Avanzada

MSc. Industrial Engineering

Autor
Belén Castellote López

Dirigido por
Silvia Tolu

Lyngby, Denmark
30 May, 2023

Clasificación de objetos por visión y tacto a través de algoritmos Faster R-CNN y Machine Learning para agarre robótico

Autor: Belén Castellote López

Director: Silvia Tolu

Entidad Colaboradora: ICAI Universidad Pontificia Comillas.

Resumen del proyecto

El informe discute el concepto de percepción multimodal, que sugiere que los humanos dependen de múltiples sentidos, como el audiovisual, el tacto y la visión, para comprender el entorno. En relación a estos sentidos, destaca la interdependencia entre tacto y vista y es importante incorporar entradas sensoriales diversas para los procesos de toma de decisiones. En la robótica industrial, comprender las capacidades de percepción de los robots se vuelve crucial, ya que éstos carecen de habilidades innatas y dependen de sistemas implementados. El enfoque se centra en mejorar la percepción de los robots de agarre fijo para mejorar la calidad, la localización y la identificación de objetos, así como la seguridad humana. La clasificación de objetos plantea desafíos, especialmente cuando los objetos son similares pero están hechos de diferentes materiales, y utilizar como única herramienta la clasificación visual puede resultar computacionalmente costoso.

En este proyecto, se ha llevado a cabo una investigación para combinar sensores visuales y táctiles con el objetivo de mejorar la categorización de objetos. Se plantean varias preguntas relacionadas con el uso de un gripper rígido para la clasificación de objetos. Entre estas preguntas se incluyen si es posible clasificar objetos con la

misma apariencia pero con durezas diferentes, el potencial de mejorar significativamente la precisión combinando datos visuales y táctiles, y las situaciones en las que un protocolo que combine ambos sentidos sería beneficioso para la clasificación de objetos.

La finalidad del proyecto es desarrollar un protocolo que combine información visual de una cámara e información táctil obtenida mediante los sensores de presión de un *gripper*. La idea principal es construir un modelo de clasificación utilizando Region-Based Convolutional Neural Network (R-CNN) y Machine Learning (ML). El propósito es diferenciar eficazmente objetos con características visuales similares pero con características táctiles variables. Los principales objetivos de investigación son los siguientes:

- Diseñar un modelo táctil capaz de distinguir entre diferentes niveles de dureza (suave y duro).
- Generar múltiples modelos de ML para determinar el algoritmo más preciso entre ellos.
- Evaluar el rendimiento de R-CNN cuando los datos sensitivos son desconocidos y compararlo con la salida del protocolo cuando se combina con la percepción táctil.
- Verificar que el modelo seleccionado logra una clasificación precisa en un escenario de caso de uso.

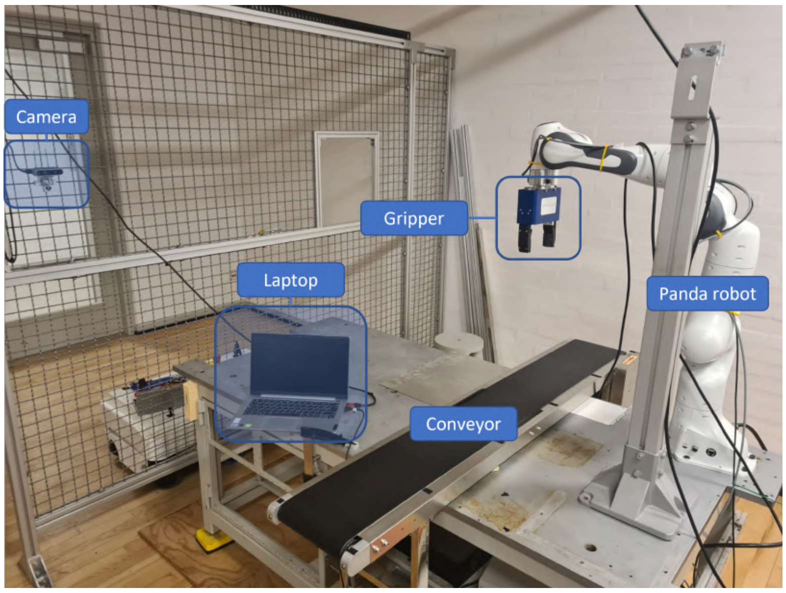


Figure 1: Imágen de la salida del laboratorio.

Este proyecto es significativo no solo en términos de sus contribuciones teóricas, sino también a través de su experimentación práctica realizada en el Laboratorio de Sistemas Autónomos en DTU (Universidad Técnica de Dinamarca). La configuración experimental, como se muestra en la Fig. 1, consta de una cámara y un *gripper* equipado con siete sensores de presión en cada dedo, que funcionan como sensores visuales y táctiles, respectivamente.

Considerando la metodología para obtener el algoritmo de clasificación, es necesario estudiar la clasificación táctil por separado, seguida de la clasificación visual y, finalmente, la integración de ambas. Sin embargo, antes de esto, es esencial comprender el proceso de adquisición de datos del *gripper*.

En este contexto dado, una prueba táctil se refiere al proceso de extraer información de un objeto utilizando el *gripper*. El enfoque específico empleado depende de las características del *gripper* utilizado. Si fuera factible recopilar toda la información requerida utilizando ciertos sensores, una posible prueba táctil podría involucrar deslizar la herramienta sobre el objeto para capturar su forma en forma de imagen. Alternativamente, en el caso de los grippers blandos, se podría crear una imagen sensorial cuando se agarra el objeto. Sin embargo, cabe señalar que el *gripper* seleccionado para este proyecto posee sensores limitados, capaces solo de adquirir información de contacto en forma de evento y fuerza. Por tanto, los datos obtenidos por cada medición serán parecidos a los mostrados en la Fig. 2.

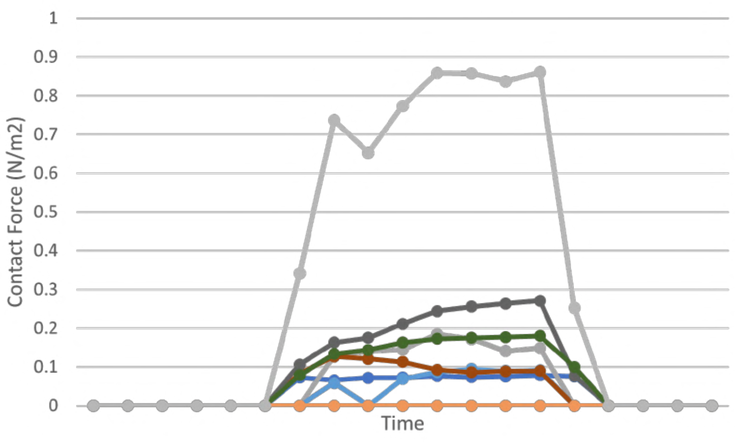


Figure 2: Gráfico de las medidas de fuerzas de contacto una vez realizado el test táctil en una botella vacía.

Una vez que se comprende el proceso de percepción de la información, los algoritmos asumirán la responsabilidad de clasificar el objeto en función de su forma y dureza, utilizando indicaciones visuales y táctiles, respectivamente. La coordinación entre estos algoritmos culminará en un algoritmo visión-táctil como el que se representa esquemáticamente en la Fig. 3.

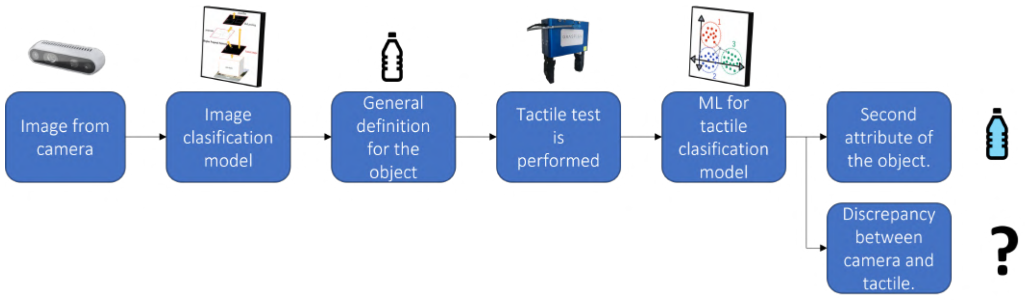


Image from camera
Image clasification model
General definition for the object
Tactile test is performed
ML for tactile clasification model
Second attribute of the object.
Discrepancy between camera and tactile.
?

Figure 3: Diseño del algoritmo de clasificación de objetos.

Una vez establecida la coordinación de los sentidos, el siguiente paso conlleva entrenar y probar los algoritmos. Se realizan experimentos con varios objetos caracterizados por su forma y dureza, como botellas llenas y vacías. Si bien el conjunto de datos para entrenar el Faster R-CNN se puede obtener de Internet, los datos de percepción táctil deben obtenerse directamente del *gripper*. Por lo tanto, se debe crear un conjunto de datos para cada objeto para permitir el análisis de la información táctil. A continuación, se eligen los modelos de ML seleccionados: árboles de decisión, K-Nearest Neighbour (KNN) y Support Vector Machine (SVM).

Una vez que se realizan el entrenamiento y las pruebas, se lleva a cabo un caso de uso para verificar los resultados obtenidos. Estos resultados se muestran en la Fig. 4.

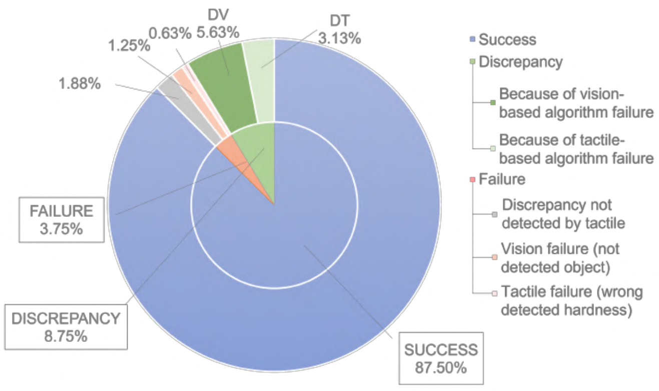


Figure 4: Resultados del caso de uso de precisión en el algoritmo visión-táctil.

La detección táctil de discrepancias fue evaluada inicialmente como un fracaso cuando se evaluó de forma independiente. Sin embargo, al considerar el sistema en su conjunto, se encontró que la detección de diferencias puede considerarse un éxito potencial. El sistema demuestra una alta probabilidad de detectar y clasificar correctamente objetos con lecturas de sensores diferentes al repetir el proceso de verificación. Como resultado, la tasa de éxito general del algoritmo se calcula en un 95%. Cabe destacar que el componente visual del algoritmo muestra una tasa de fracaso más alta del 8.75% en comparación con el componente basado en tacto, que tiene una tasa de fracaso del 5.7%.

En general, los hallazgos de la tesis destacan la implementación exitosa de un sistema de clasificación de objetos utilizando entradas visuales y táctiles. El enfoque visión-táctil superó al enfoque solo de visión y mostró promesas para aplicaciones prácticas, especialmente en la industria alimentaria. Sin embargo, la refinación adicional y la integración con sistemas robóticos son áreas potenciales para investigaciones futuras.

El informe de la tesis está estructurado en varios capítulos. El Capítulo 2 proporciona una visión general del contexto, discutiendo la percepción humana y revisando enfoques anteriores en el campo. En el Capítulo 3, el enfoque se centra en la teoría detrás de los algoritmos de clasificación para el reconocimiento de objetos basado en visión y táctil. El Capítulo 4 entra en detalle sobre los métodos y herramientas utilizadas en el proyecto, comenzando desde los requisitos y cubriendo todo el proceso de implementación. El Capítulo 5 está dedicado al diseño de la prueba táctil, incluyendo las métricas utilizadas, la recopilación y procesamiento de datos, y la implementación de los algoritmos. También presenta los resultados finales y el análisis dentro de un escenario específico de caso de uso. Los capítulos restantes sirven como discusión y conclusión, resumiendo los logros del proyecto. El informe incluye apéndices al final, que proporcionan información adicional para complementar el contenido.

Vision-tactile object classification through Faster R-CNN and Machine Learning algorithms for robotic grasping

Author: Belén Castellote López

Director: Silvia Tolu

Collaborating Entity: ICAI Universidad Pontificia Comillas.

Project summary

The report discusses the concept of multi-modal perception, which suggests that humans rely on multiple senses, such as audiovisual, touch, and vision, to understand the environment. It highlights the interdependence between touch and sight and emphasizes the significance of incorporating diverse sensory inputs for decision-making processes. In industrial robotics, understanding robots' perception capabilities becomes crucial as they lack innate abilities and rely on implemented systems. The focus is on enhancing perception for fixed grasping robots to improve quality, object localization, identification, and human safety. Object classification poses challenges, especially when objects are similar but made of different materials, and visual classification can be computationally expensive.

An investigation has been conducted into combining visual and tactile sensors to improve object categorization. It raises several questions related to utilizing a hard gripper for object classification. These questions include whether it is possible to classify objects with the same appearance but different hardness, the potential for significantly improved accuracy by combining vision and tactile data, and the situations in which a protocol combining both senses would be beneficial for object

classification.

The project's goal is to develop a protocol that combines visual information from a camera and tactile information from a gripper. The overarching idea is to construct a classification model utilizing R-CNN and ML. The objective is to effectively differentiate objects with similar visual characteristics but varying touch characteristics through classification. The main research objectives are as follows:

- Design a tactile model capable of discerning between different levels of hardness (soft and hard).
- Generate multiple ML models to determine the most accurate algorithm among them.
- Evaluate the performance of glsrcnn when sensitive data is unknown and compare it to the protocol's output when combined with tactile perception.
- Verify that the selected model achieves precise classification across diverse use case scenarios.

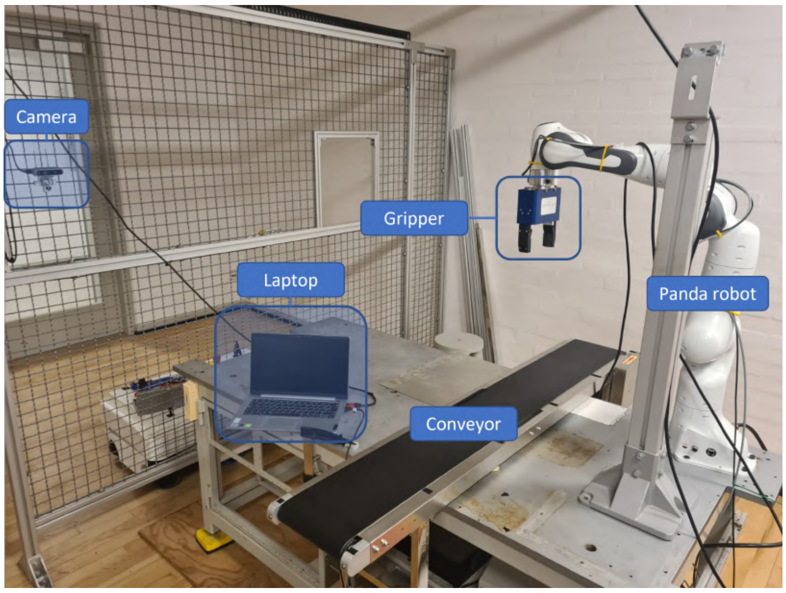


Figure 5: Picture of the laboratory output.

This project holds significance not only in terms of its theoretical contributions but also through its practical experimentation conducted at the Autonomous Systems Laboratory at DTU (Technical University of Denmark). The experimental setup, as depicted in Fig. 5, comprises a camera and a gripper equipped with seven pressure sensors on each finger, functioning as vision and tactile sensors, respectively.

Considering the methodology for obtaining the classification algorithm, it is necessary to study tactile classification separately, followed by visual classification, and finally the integration of both. However, prior to this, it is essential to understand the data acquisition process from the gripper.

In this given context, a tactile test refers to the process of extracting information from an object using the gripper. The specific approach employed depends on the characteristics of the gripper utilized. If it were feasible to gather all the required information using certain sensors, a potential tactile test could involve sliding the gripper across the object to capture its shape in the form of an image. Alternatively, in the case of soft grippers, a sensory image could be created when the object is grasped. However, it should be noted that the gripper selected for this project possesses limited sensors, capable only of acquiring contact information in the form of an event and force. Therefore, the data obtained from each measurement will be similar to those shown in Fig. 6

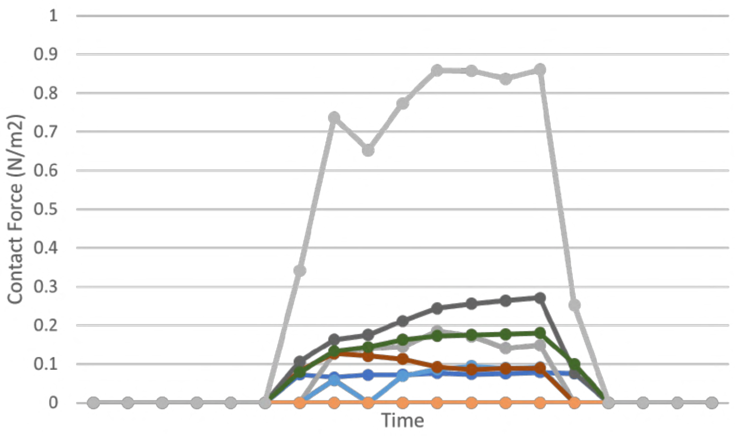


Figure 6: Plot of contact forces measures once tactile test is performed on empty bottle.

Once the process of information perception is comprehended, the algorithms will assume the responsibility of classifying the object based on its shape and hardness, using visual and tactile cues, respectively. The coordination between these algorithms will culminate in a vision-tactile algorithm, which is depicted schematically in the Fig. 7.

Once the coordination of the senses is established, the next step involves training and testing the algorithms. Experiments are conducted on various objects characterized by their shape and hardness, such as full bottles and empty bottles.

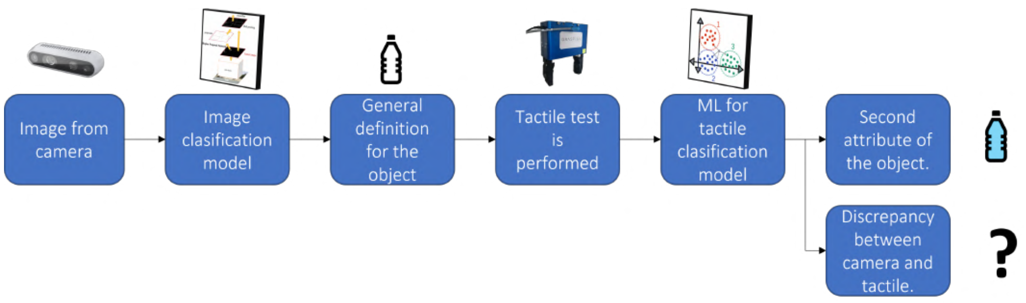


Image from camera
Image clasification model
General definition for the object
Tactile test is performed
ML for tactile clasification model
Second attribute of the object.
Discrepancy between camera and tactile.
?

Figure 7: Design of the object classification algorithm.

While the dataset for training the Faster glsrcnn can be sourced from the internet, the tactile sensing data must be obtained directly from the gripper. Therefore, a dataset needs to be created for each object to enable analysis of the tactile information. Afterward, the selected glsml are chosen: decision trees, KNN and SVM.

Once the training and tests are performed, a use case takes place in order to check the results obtained. This is shown in the Fig. 8.

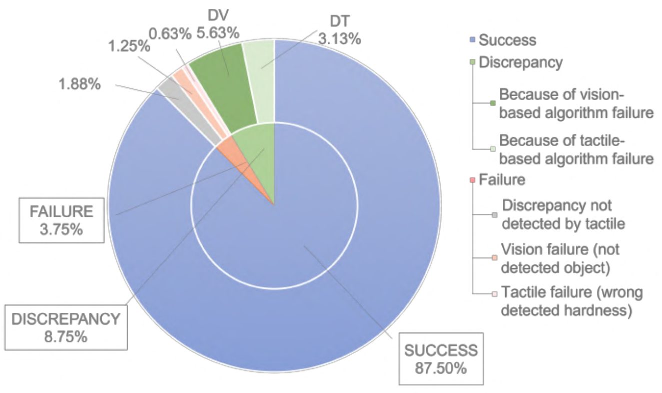


Figure 8: Accuracy use-case results on the vision-tactile algorithm.

The tactile-based detection of discrepancies was initially assessed as a failure when evaluated independently. However, when considering the system as a whole, it was found that the detection of mismatches can be deemed a potential success. The system demonstrates a high likelihood of correctly detecting and classifying objects with differing sensor readings upon repeating the verification process. As a result, the overall success rate of the algorithm is calculated to be 95%. Notably, the visual component of the algorithm exhibits a higher failure rate of 8.75% compared to the tactile-based component, which has a failure rate of 5.7%.

Overall, the thesis findings highlight the successful implementation of an object classification system using both visual and tactile inputs. The vision-tactile approach outperformed the vision-only approach and showed promise for practical applications, particularly in the food industry. However, further refinement and integration with robotic systems are potential areas for future research.

The thesis report is structured into several chapters. Chapter 2 provides an overview of the context, discussing human perception and reviewing previous approaches in the field. In Chapter 3, the focus is on the theory behind classification algorithms for vision and tactile-based object recognition. Chapter 4 goes into detail about the methods and tools employed in the project, starting from the requirements and covering the entire implementation process. Chapter 5 is dedicated to the design of the tactile test, including the metrics used, data collection and processing, and the implementation of the algorithms. It also presents the final results and analysis within a specific use case scenario. The remaining chapters serve as a discussion and conclusion, summarizing the project's achievements. The report includes appendices at the end, which provide additional information to complement the content.

Agradecimientos

I am deeply grateful to Silvia Tolu, who served as my supervisor throughout the development of this project and provided invaluable guidance along the way. Her expertise and support have been instrumental to the success of this thesis.

Additionally, I would like to express my gratitude to Jacob Fiskaali for sharing his knowledge and insights from the thesis and project he developed over the course of two years. His contribution has been immensely valuable to my own work.

I am also thankful to Jacob Benchmann Pederssen for providing mechanical solutions to the challenges posed by the gripper. His expertise has been essential to the success of this project.

I am grateful to Carlos Marcos for sharing his knowledge on working with HPCs. His input has been invaluable to my understanding of this aspect of the project.

I would like to extend my thanks to Nils Meile for his vision and contributions to the project, and for collaborating with me in seeking a common solution for our projects in the future.

Finally I am grateful to my family and boyfriend for their constant patience, motivation and suggestions throughout my academic journey. Their encouragement and belief in me has been invaluable and has made all the difference in the completion of this thesis.

Belén Castellote

Lyngby, Denmark

30 January, 2023

Contents

List of Figures

List of Tables

Acronyms

ANN	Artificial Neural Network.
CNN	Convolutional Neural Network.
HPC	High Power Computer.
IoU	Intersection-over-Union.
KNN	K-Nearest Neighbour.
mAP	Mean Average Precision.
ML	Machine Learning.
MTP	Mean Time Performance.
ND	Not detected object.
PCA	Principal Component Analysis.
PIO	PlatformIO.
POI	Point Of Interest.
R-CNN	Region-Based Convolutional Neural Network.
ROS	Robot Operating System.
RPN	Region Proposal Network.
SDG	Sustainable Developments Goals.
SIFT	Scale-Invariant Feature Transform.
SLAM	Simultaneous Localization And Mapping.
SVM	Support Vector Machine.

TD	Discrepancy due to tactile-based algorithm failure.
TF	Tactile-based algorithm failure in classifying the object hardness.
VD	Discrepancy due to vision-based algorithm failure.
VF	Vision-based algorithm failure in classifying the object type.
VSCode	Visual Studio Code.
VTF	Tactile-based and vision-based algorithm failure in classifying the object type.

Chapter 1

Introduction

Several studies suggest that human perception can be multi-modal [1], meaning that more than one sense participates in understanding the environment. Most of the research for this multi-modal perception is done in the audiovisual scenario [1], as there are various daily tasks where these two senses are intertwined; for example when a friend is talking to another in a loud noise environment and it is possible to interpret the sounds by reading the lips or when a calisthenics move should be done just by following the beat of a song [2].

Furthermore, it has been established that the senses of touch and vision possess a significant interdependence [3]. By incorporating both touch and visual sensory input, the brain receives diverse information about the environment, allowing for the integration of the most relevant information in decision-making processes.

In the context of industrial robotics, it is crucial to understand the background of robots' perception capabilities. Unlike humans, robots do not possess innate perception abilities and must rely on implementations that have been developed for various types of robots, such as fixed-base manipulators, collaborative robots (cobots), mobile robots, and mobile manipulators [4]. The main focus of perception enhancement for fixed grasping robots is to improve quality, accurately locate objects, identify objects, and enhance human safety during the industrial process.

Object classification presents a significant challenge in the context of fixed grasping robots operating in an industrial setting. The ability to classify objects is crucial for detecting misplaced objects or products of poor quality. While significant research has been conducted on visual object classification, there are scenarios where visual discrimination is difficult, such as when objects are similar but constructed from differing materials or hardness. Additionally, visual classification is computationally expensive and becomes increasingly so as the number of objects considered in the classification increases.

1.1 Problem statement

Investigation has been conducted regarding the combination of visual and tactile sensors to enhance object categorization. However, when it comes to utilizing a hard gripper for object classification, several queries arise:

- The human sense of touch usually helps us determine what kind of material an object is made of. *Would it be possible to classify different objects that look the same into two different groups according to their hardness?*
- As mentioned before, vision is sometimes enough for classifying objects, but it might not be the most accurate way. *Will the model be significantly more accurate when we use both data from vision and tactile than just using vision?*
- *What are the situations in which a protocol that combines both vision and tactile information to classify objects would be useful?*

1.2 Motivation of the project, goals and methods

The aim of the project is to create a protocol that fusions both the vision information from the camera and the tactile information from the gripper. The general idea is to create a classification model based on Convolutional Neural Network (CNN) and ML. Objects with similar visual characteristics but distinct touch characteristics, should be separated through classification. Among the main research objectives are the following:

- Design a tactile model which is able to distinguish between different levels of hardness (soft and hard).
- Create different models (ML) to find the most accurate algorithm among them.
- Examine the performance of the R-CNN without knowing the sensitive data and compare it to the output of the protocol when it is working together with the tactile perception.
- Check that the selected model is capable of performing accurate classification under different use case scenarios.

This project also aligns with sustainability goals as outlined by the Sustainable Developments Goals (SDG) of 2020. Specifically, it actively contributes to two of the main goals, including goal 8: Decent Work and Economic Growth, as it seeks to reduce manual labor and increase income. Additionally, it contributes to goal 9: Industry, Innovation, and Infrastructure, through its utilization of cutting-edge algorithms and technologies in the research.

In the design chapter, more information will be provided in order to show the methods used to achieve those goals.

1.3 Thesis overview

The structure of the thesis report is as follows:

- Chapter 2 gives the context overview by describing the human perception and reviewing the previous approaches in the field.
- Chapter 3 focuses on the theory and the selected classification algorithms for vision and tactile-based object recognition.
- Chapter 4 details the methods and tools used in this project, from the requirements until the implementation.
- Chapter 5 presents the design of the tactile test, the metrics used, the data collection and processing, and the implementation of the algorithms. It also showcases the final results and analysis for a use case scenario.
- The final two chapters discuss and conclude the project achievements.

Chapter 2

Literature review

2.1 Perception in humans

Humans have served as inspiration for the robotic industry in terms of perception. Therefore, it is crucial to understand the behavior of the human body and its perceptual abilities before explaining the approaches made in robot's perception. The project focuses on the touch and sight senses.

As humans, we count with different perceptual abilities that help getting information from the environment, together with interacting with ourselves. These senses take information thanks to the receptors, which are the sensory systems that detect the world from the outside (exteroceptors), those that get data from internal organs (interoceptors) and those in charge of detecting position and load (proprioceptors) [5]. Both vision and touch are considered exterosenses. Their performance to get information from the environment is different. Although vision does not require direct interaction with the environment, touch does. Most research on the senses emphasizes the importance of visual acuity, but this dominance of vision is socially and culturally reinforced, not a law of nature [6].

Sight sense

The visual interpretation of humans is processed in the visual cortex, which receives, integrates, and processes visual information from the retinas [8]. Fig. 2.1 shows an explanatory image of the parts of the human visual system. In the image, information is taken (1), visual information is processed (2) information is extracted and sent to V1 (primary visual cortex) and then to V2 until V5. As this happens, the layers communicate and give feedback to each other. Simultaneously, neurons in layers communicate and get the missing information [7].

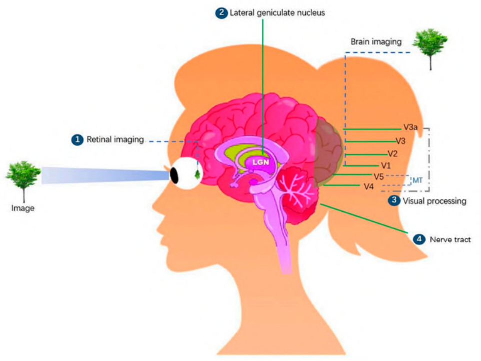


Figure 2.1: Human visual system schematic. Fig. from [7].

Touch sense

The touch perception is done through the exteroceptors, neurons, and the somatosensory cortex. This last one mentioned, is responsible for transforming information from the exteroceptors into human body sensations [10]. Fig. 2.2 explains the touch pathway with the nervous system. First-order neuron connects exteroceptors with the dorsal column and medulla. From there second order take the information from the Medial lemniscus until the thalamus. Third-order neurons take the information from there to the cerebral cortex [9].

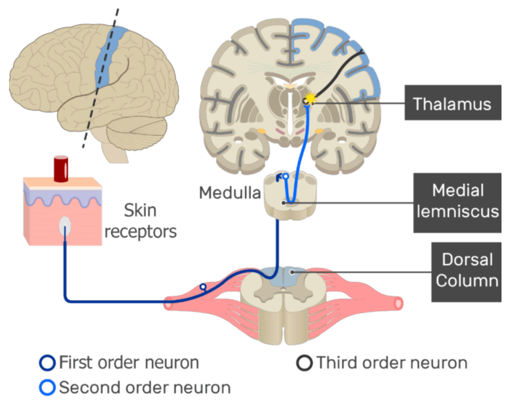


Figure 2.2: Touch Pathway of the nervous system. Fig. from [9].

Tactile sensing is divided into two modalities: intrinsic and extrinsic tactile sensing, with the respective physiological concepts being kinesthetic (limb movement and position, force perception [11]) and cutaneous sensing (vibration, temperature and pain). Haptic perception refers to perception based on cutaneous and kinesthetic sensing. This perception is required to extract information including an object's overall shape or hardness [12].

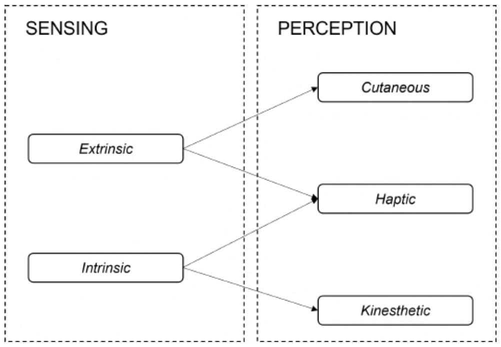


Figure 2.3: Classification and connection between tactile sensing and perception.

Thanks to the tactile sense, humans are able to identify an object, its shape, temperature, roughness and hardness of the material, etc.

2.2 Perception in robotics

Despite the advances in robotics today, robots used to have limited understanding of the environment, which made them dangerous and limited. Perception has played a crucial role in helping these machines understand the world and behave more like humans.

Visual Perception

With vision and robot technology combined, intelligence and reliability are enhanced. The purpose of adding vision perception to a task varies depending on the way it is included. For example, by using a moving camera or two or more cameras at the same time we can create different 3D images of the environment. This type of images are practical in cases where depth is needed. For instance, they can be used to recreate a room or an object. Besides, 3D images can be used in mobile robotics to orientate the machine with visual odometry and mapping algorithms.

According to 2D images, gray-scale images are useful for object detection, while the addition of RGB information improves the results on object classification and identification.

Tactile Perception

In these days, tactile sensing involves the use of multiple types of sensors in different robotic tasks. In particular, manipulator robots focus on haptic perception, which is the aspect of tactile sensing concerned with grasping an object. To this end, various sensors are used for different tasks, such as:

Measure the temperature: in order to determine if another machinery/human can safely handle the object.

Shape and texture: different sensors are used to gather this information. Some approaches have been done with normal pressure sensors, describing this tactile information as image pixel [13]. Soft grippers and dexterous robotic hands are also suitable sensors for these tasks.

Sliding detection: to determine if the object grasp is still on position or moving. This was previously accomplish by using an accelerometer [14].

2.3 Multi-sensing perception for multiobject classification

Previous studies have investigated various methods for object identification and classification. Several of them used tactile sensors. For example, by using an under-actuated robotic hand performing both single-exploratory and short-exploratory movements [15]. Other research done in touch-sensitive fingers gripper, was based on creating images for each finger and performing a K-means clustering [16]. As an outcome, both systems were able to identify objects with different shapes and materials.

Some of them used a more advanced tactile sensors such as a dexterous robotic hand [17]. The approach done a few years ago also involved computer vision to determine the material and shape of a specific ball. At the end of the article, the authors discussed the potential for deep learning-based implementation to improve the results.

Besides the type of tactile sensor used, the data collection process and the tactile test performed to gather data can vary, with three general approaches to grasping: single-grasping [15, 16, 18], short-exploratory grasping movements [15] and sliding grasping [19, 20, 21].

Even though single-grasping is less precise, it is often the preferred method due to its speediness, as demonstrated in [15]. However, combining it with another sense, such as vision, can result in noticeable improvements in the identification accuracy.

Not many experiments have combined vision and tactile elements for their approach. However, those that have merged them have focused on different object characteristics. For example, the approach explained in [13] tries to classify the object according to shape, size and visual characteristics. However, another experiment [21] defined the approach for textures and materials.

2.3.1 Object classification algorithms

Many projects have been carried out using a variety of algorithms to achieve different objectives with both the data obtained from cameras and the information obtained from tactile sensors. Object classification using both tactile and visual information is one of the tasks that has been tackled using these algorithms. Some techniques use the algorithms separately and then combine the results.

If we focus on the methods for object identification in images in the context of robotic manipulators, the most commonly used methods are usually 2D. On the other hand, 3D techniques are typically employed in methods such as Simultaneous Localization And Mapping (SLAM) to create maps based on the information collected by mobile robots, or to create a descriptive model of an object for further analysis.

Nevertheless, paying attention to previous experiments, it is possible to highlight different algorithms at the visual level. The Scale-Invariant Feature Transform (SIFT) algorithm is one of the simplest ones. It only detects a clearly defined object and find it in a image [22]. On the other hand, Artificial Neural Networks (ANNs) are discarded in processes that need to work in real-time, as they are particularly slow. In this regard, the most common options for object classification are the CNNs. These neural networks reduce the dimensionality of images, making them faster and more efficient. Besides, CNNs are unsupervised, so they require less information to perform well. To achieve not just classification, but also identification of the object in space, one needs to work with R-CNN. These CNNs are able to not only extract features from the image to classify objects, but also identify them within the image by searching for the correct bounding box.

Furthermore, when it comes to data interpretation of tactile sensors using the single-grasp method, two techniques stand out. The first one involves analyzing the sensor information and applying various ML techniques, such as KNN, trees, SVM, boosted trees, ANN, etc. (referenced in [13, 15, 18, 19]) with the objective of finding similarities and differences between objects. The other widely used technique is creating a sensory map that is represented in an image format, which can then be analyzed using the image-based classification methods discussed earlier (referenced in [16, 21]).

Although some previous experiments did the approach for each sense and then fused them [13], there is at least one project where they use the fusion of both as a method, as in [23]. In this approach, a neural network is implemented to fuse information from both sensors in a single algorithm. The researchers created a neural network model for the camera images and another one for the tactile map images. Afterwards, they fused the results with another neural network, resulting in a vector indicating the similarity of a tested object to the 26 classes of items it had to classify. This algorithm can label the objects, but it does not have the ability to identify a given object in space.

In the framework of a MSc thesis, the Graspian company, in collaboration with DTU, built a gripper with two fingers and pressure sensors (tactile sensors). The focus of Jacob Fiskaali's MSc thesis was to integrate the pressure sensors into a robotic loop [24].

The thesis project explained in this report implements a classification model that uses both visual and tactile (pressure) information to classify objects with different characteristics. The visual information is used to classify the shape of the objects, and the pressure sensors, which are type of tactile sensor assembled on a rigid gripper, are used to identify the hardness of the object. This approach, which relies on bio-inspired algorithms utilizing both visual and pressure information, brings the identification process closer to human performance, where multiple senses are employed.

Chapter 3

Theory

This chapter explains the theory behind the selected algorithms. This will help to understand the later explanation of the implementation in software for both the vision-based algorithm (Faster R-CNN) and the tactile-based algorithm, based on ML (Decision trees, KNN and SVM).

3.1 Vision-based algorithm - Faster Region-Based Convolutional Neural Network

Among the different eligible vision algorithms, the Faster R-CNN is chosen because it is a fast algorithm that can identify where the object is located. This algorithm is a CNN developed based on the indications in the 2015 paper: *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks* [25]

This paper introduced the concept of Faster R-CNN and its functioning, and there have been several implementations, with the most commonly used one being the *Pytorch* implementation (currently in beta stage).

This algorithm is based on a previous algorithm called Fast R-CNN [26]. Both of them belong to the R-CNN familly. The explanation diagram for the Faster R-CNN is presented in Fig. 3.1. This section outlines the various parts of the algorithm.

3.1.1 Convolutional Neural Network

This CNN is commonly referred to as the backbone and it is the most computationally intensive component of the algorithm. The input to the first layer is the

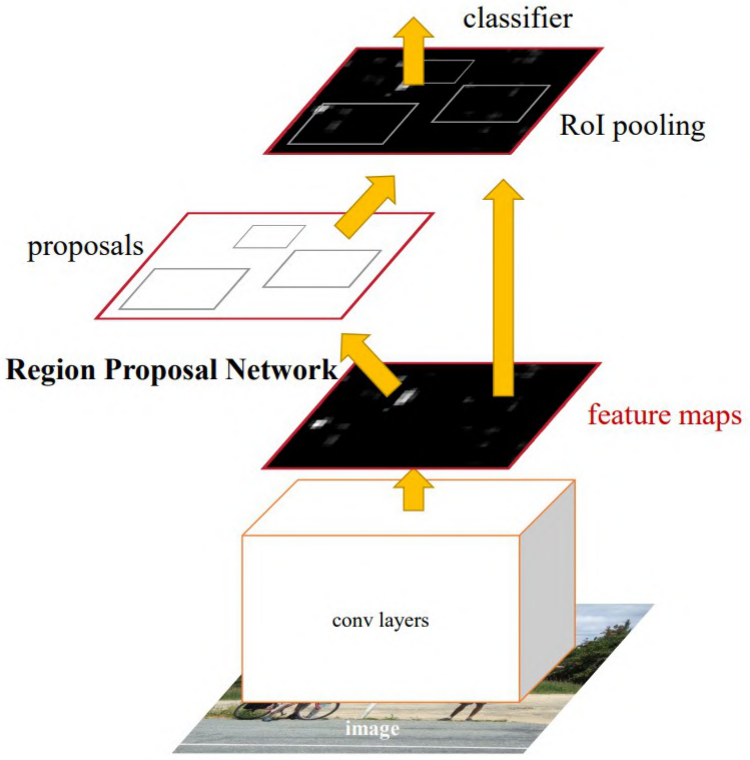


Figure 3.1: Contents of the Faster R-CNN algorithm, from original paper [25].

image data and a filter or feature detector. The output of the final layer is a feature map. The filter is a 2D array of weights that represents a portion of the image. In this project, a 3x3 matrix is used since we are working with RGB images (one dimension for each reference color). The filter is applied to small sections of the image (as shown in Fig. 3.2) and the result is a feature map. It's worth mentioning that after each intermediate convolutional layer, a ReLU activation function is applied.

Different CNN have been tested in conjunction with the Faster R-CNN algorithm. However, in this project, the chosen CNN is ResNet 50 (shown in Fig. 3.3). Compared to other possible options (VGG16, VGG19), it is a CNN with fewer parameters and reduces the dimension of the image faster in each of the layers used. On the other hand, there are more advanced backbones, such as ResNet101, which has double the number of layers. This makes the training more difficult and the results more accurate, which makes sense to implement when the objects to be identified are similar.

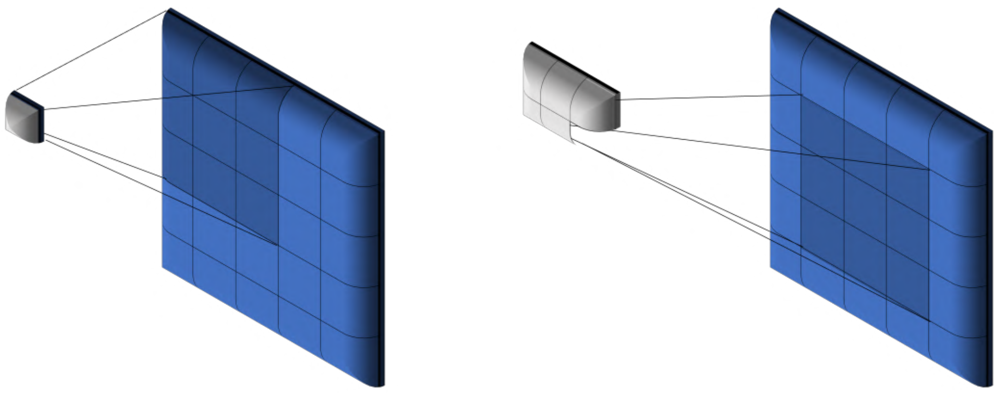


Figure 3.2: Inside work of a convolutional layer.

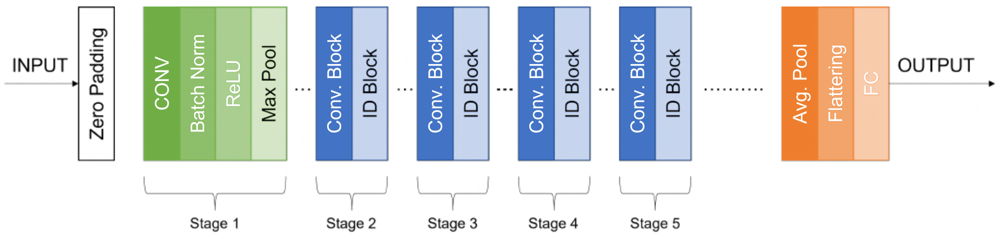


Figure 3.3: Resnet 50 backbone algorithm.

3.1.2 Region proposal network

The Region Proposal Network (RPN) is a CNN and it is a new component in the Faster R-CNN algorithm compared to Fast R-CNN. With the feature map as an input, the RPN aims to produce a set of rectangular object proposals, known as anchors, each with an objectness score. To generate these region proposals, a small network is slid over the final shared convolutional layer's feature maps, resulting in multiple sliding windows. The dimensions of each sliding window are reduced based on the backbone CNN used. An anchor is centered at the analysed sliding window and it is assigned a predetermined scale and aspect ratio. output will provide the calculated bounding box according to its coordinates and a vector indicating whether the box represents the background or an object. Before using the RPN in the detection task, it needs to be trained. The anchor is given a positive or negative value based on its Intersection-over-Union (IoU) with the ground truth bounding boxes.

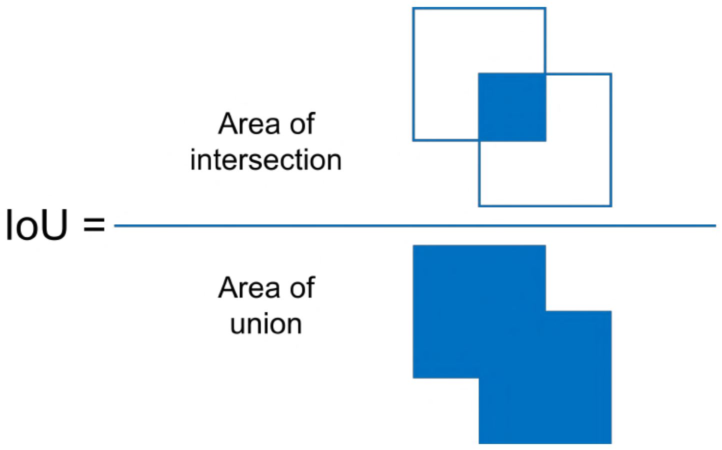


Figure 3.4: Region Proposal Network for Faster R-CNN [25].

3.1.3 Region of Interest (RoI) Pooling and Loss function

This layer of the algorithm is responsible for merging the results obtained by Fast-R-CNN with the outputs of the RPN to make it more efficient. Once the proposed regions have been obtained, these are combined with the feature map to identify the object with a softmax classifier and select the bounding box. Fig. 3.5 explains the last step in more detail. The classification and regression are performed using a loss function. This loss function, which incorporates the RPN classification loss, RPN regression loss, and Fast-R-CNN classification and regression loss, is applied to the output of the Fast-R-CNN branch. The loss is calculated for each proposal, and the gradients are backpropagated through the network to update the model's parameters. For further information on loss calculation, refer to Appendix B.

Traditionally, the Mean Average Precision (mAP) is the evaluation metric used in object detection tasks for Faster R-CNN. It calculates the average precision of the model across all object classes. It is a way to measure the model's ability to correctly identify objects while avoiding false positives. It takes into account both precision and recall of the model, which are important indicators of the model's performance. mAP provides a single number summary of the model's performance and it can be used to compare different models or track the performance of a model over time.

3.2 Tactile-based algorithm - Machine Learning

In this project, the output of the tactile test is a vector of 14 positions, each indicating the contact force measured by each of the sensors of the gripper. Regarding

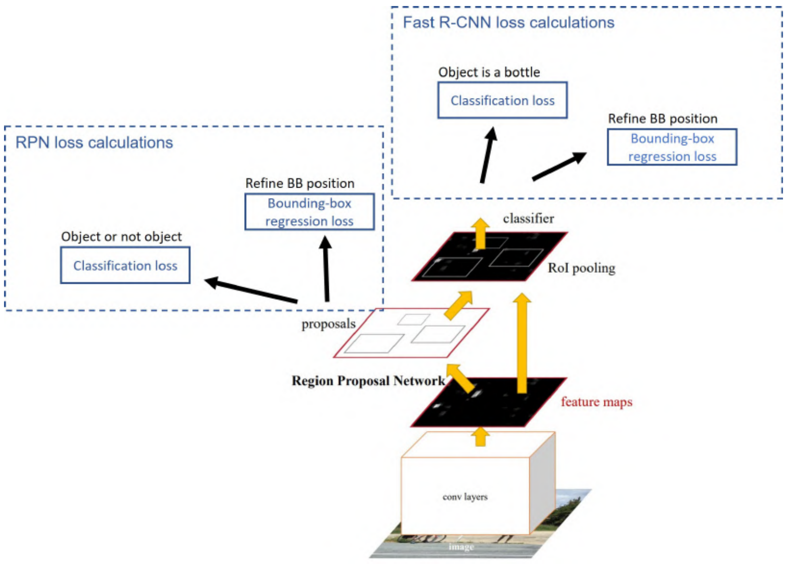


Figure 3.5: Faster R-CNN algorithm extended explanation and example.

this, it has been determined that the best approach would be to implement ML algorithms. Among the various ML algorithms available, it is important to mention the two main categories:

Supervised: Here, a training dataset is labeled with the expected class or outcome for each row of data.

Unsupervised: In this case, there is no prior information about the outcome, and the algorithm attempts to group the data into clusters for better understanding.

Moreover, supervised ML relies on using previous data to predict the results, making it the preferred option for this project. Therefore, among the many possible algorithms, three of them have been selected for this project and are explained in the following sections.

3.2.1 Decision Tree

The idea of using a Decision Tree for classification is to predict a discrete output label y_i for observation i based on features x_i [27]. The goal is to build a training model that can predict the class or value of the target variable by learning simple decision rules inferred from prior data (training data).

This algorithm starts from the root, which is referenced to all data contemplated for the training. From this features, a decision is taken according to the values of the different variables, where the dataset is split in two or more nodes depending on the possible values of a certain input variable. Fig. 3.6 shows the different terminology of nodes after splitting.

Decision nodes are those that are split in different nodes.

Terminal nodes are those that are not split again. This can happen because of two reasons: there is just one class in this node or there is a limitation on depth so the algorithm doesn't get too complex.

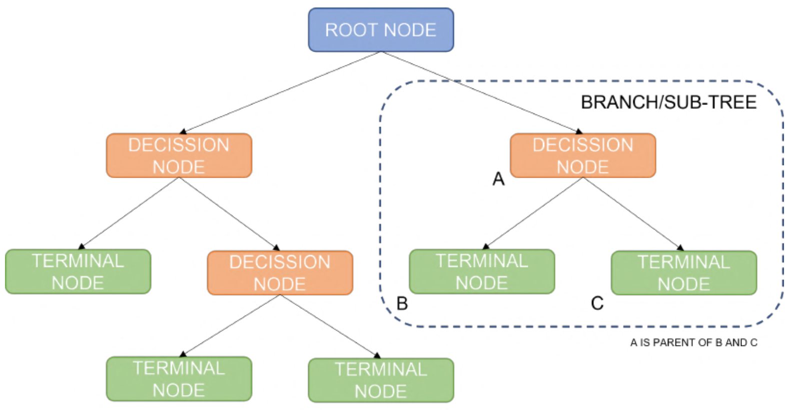


Figure 3.6: Decision tree simple graph.

The algorithm has an iterative procedure. On each iteration of the algorithm, it goes through the attribute of the set and calculates impurity. Impurity defines the percentage of mix classes in the set being processed. The meaning of having an impurity close to 0 is a better split class. There are many ways of calculating the impurity: entropy, Gini index (just for binary split), class error, etc. Besides of this case, just entropy is explained because is the selected one for this project.

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2 p_i$$

Being p_i the amount of values of a certain class among the total data in a node.

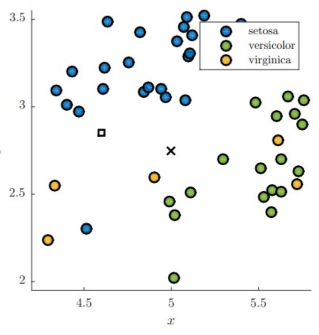
In order to work with continuous input variables, the split is performed looking for the smallest entropy when selecting a specific input and changing its split value. Then, the smallest entropy calculated among all the input variables, would be the one that will perform the split.

3.2.2 K-Nearest Neighbour

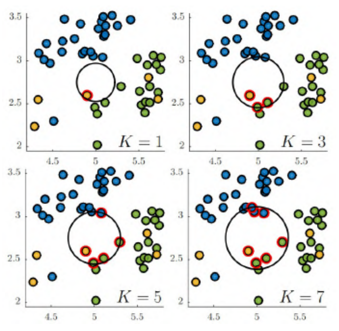
KNN is an instance-based ML algorithm type, meaning that the algorithm does not explicitly learn a model. Instead, it memorises training instances that are used as a "knowledge base" for the prediction phase. It essentially classify values by looking for the "most similar" (by closeness) data points learned in the training stage (see seven steps to create your ML) and making guesses of new points based on that classification. This method merely looks at the observations nearest to the one it is trying to predict and classifies the Point Of Interest (POI) based on the majority of the surrounding data.

To explain this algorithm, it could be broken down into 3 steps:

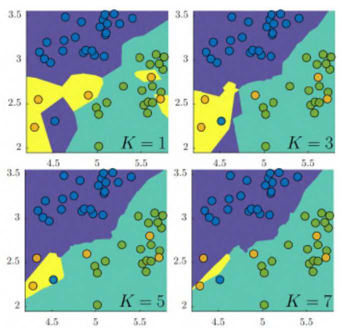
- Calculate the distance between the item to be classified and the rest of the items in the training dataset. The measure used for calculating the distance is a parameter to define in this algorithm.
- Select the "k" closest items (with a smaller distance, depending on the function used). When the k is bigger, the algorithm is less restricted and less overfitted.
- Perform a "majority vote" among the k items: those of a class/label that dominate will decide their final ranking.



(a) Subset of a dataset where only two features are consider.



(b) Illustration of the KNN of the black cross defined in a). The circle determines the included data points for determining the class.



(c) KNN classification boundary example. As K increases, the boundary becomes more smooth.

Figure 3.7: KNN example for explanation [27].

3.2.3 Support Vector Machine

SVM is a model-based machine learning algorithm that separates sample points in space into two classes as widely as possible by means of a hyperplane separator. The X-dimensional plane is defined as the vector between the two closest points of the two classes, known as the support vector. When new samples are mapped to this model, they can be classified into one of the two classes based on the spaces they belong to. SVM can also map data to a space with multiple dimensions, even more than three.

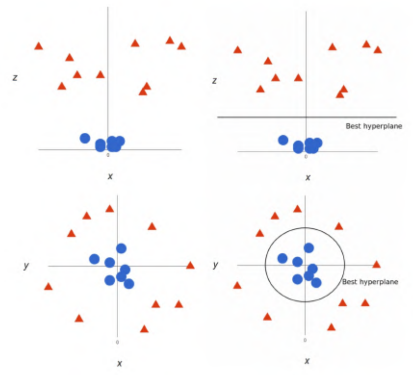


Figure 3.8: SVM graphic example for not easy to separate data.

To select the best separator, the SVM algorithm chooses the boundary that maximizes the distance to the closest elements of each class. Different boundary types can be used depending on the data distribution, such as linear, Gaussian, polynomial, etc. Fig. 3.8 shows a brief explanation on how the separation is done in linear way.

Principal Component Analysis

Principal Component Analysis (PCA) is a method of simplifying large datasets by transforming the original set of features into a new set of features that capture the most variance in the data. This process reduces the number of features while preserving the information that is most important for analysis. The result is a set of features that are uncorrelated and more interpretable, making it easier to visualize patterns in the data.

PCA can also be used in conjunction with SVM, as it can improve the accuracy of the SVM model by simplifying the dataset and reducing noise in the data. Additionally, PCA can help to visualize high dimensional data, making it easier to understand the structure of the data and interpret the results of SVM. By simplifying the dataset and reducing noise, PCA can lead to more robust and accurate SVM models, making it an important tool for data analysis and modeling.

Chapter 4

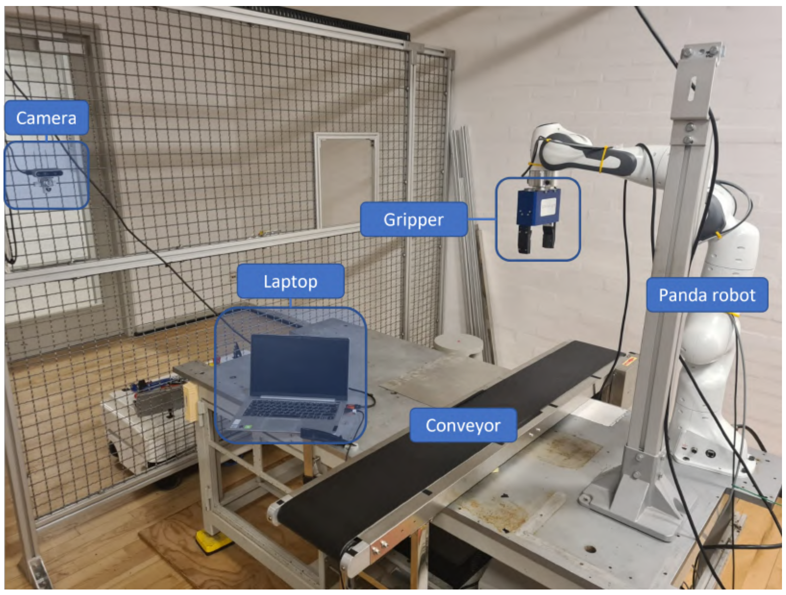
Methods and Tools

The project aims to develop an algorithm that utilizes information from a camera and touch sensors to identify an object. The object is considered to be "identified" when the algorithm can recognize its general category (e.g. bottle) and state (e.g. empty) through the combination of both visual and tactile information. The use of the tactile sensor enhances the object classification compared to using vision alone.

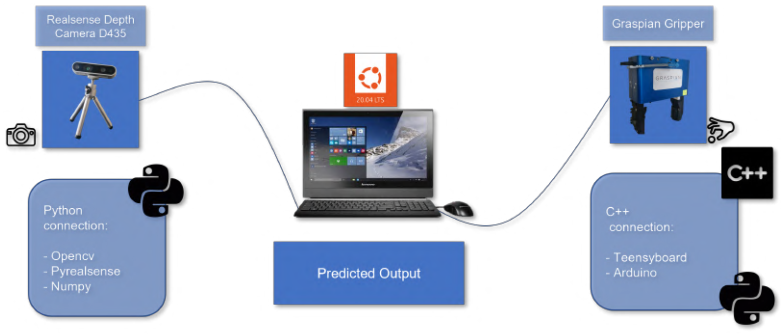
The goal of the project is to validate the results obtained from the vision algorithm using the information received from the tactile sensor.

4.1 Setup of the scene

The experiment takes place in the DTU electrical laboratory. The image in Fig. 4.1 shows a corner view of the room and a schematic representation of the setup. The setup consists of a camera acting as a visual sensor and a Gripper serving as a tactile sensor.



(a) Picture of the laboratory output.



(b) Schematic representation of the setup

Figure 4.1: Setup of the project.

4.2 Requirements

The project aims to develop an algorithm that fuses information from a camera and touch sensors to identify objects and their states in real-time. The following sections outline the requirements for the gripper, camera, vision-based algorithm, tactile-based algorithm and software used in the project. The ultimate goal is to obtain a reliable and accurate algorithm that can classify objects based on both visual and tactile information.

Gripper requirements

- The pressure sensors of the fingers must maintain relatively stable answers no matter the time passed by and the number of times the gripper is performed.
- Test done by the gripper must be able to detect sensitive objects, like food, and act over them without changing their shape or state (not squeezing them).

Camera requirements

- Able to get images where the objects can be classified also by the human eye.
- Get images in real-time.
- The camera has to work correctly with the usual light working conditions of the area.

Vision-based algorithm requirements

- The algorithm must be trained without any economic cost, from a GPU at the university.
- Identify the objects positioned on the right side of the conveyor, even if there is more than one.

Tactile-based algorithm requirements

- Get the model, run the gripper test for tactile information, apply the model and get the output in real-time.
- Get more than 3 working models in order to compare them.
- Able to identify that the object tested on the gripper is not the same as the one seen on camera.
- Accuracy over 85% to consider it valid.

Software

- Python libraries related to ML and Faster R-CNN.
- Python libraries related to camera.

- All the programs must run together since the camera detects an object on the conveyor until this object is classified with the two main attributes.
- The output of this algorithm must show also if there is a discrepancy between the two sensors (camera and pressure sensors).

4.3 Tools

4.3.1 Camera

This camera is an Intel RealSense Depth Camera D435, as shown in Fig. 4.2. The main features of the device are depicted in the figure. Despite the presence of multiple cameras, only the right camera is used in this project. Thus, the information obtained from the environment does not take into account depth vision, which can be obtained by using two cameras placed at a constant distance, such as the right and left cameras. The camera is connected to the computer through a 2-meter-long cable with USB C input for the camera and USB 3.1 input for the processor element. This arrangement may be a limiting factor for real-time image acquisition.

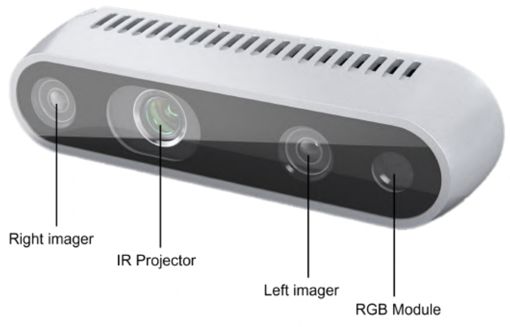


Figure 4.2: Elements of RealSense D435 Camera.

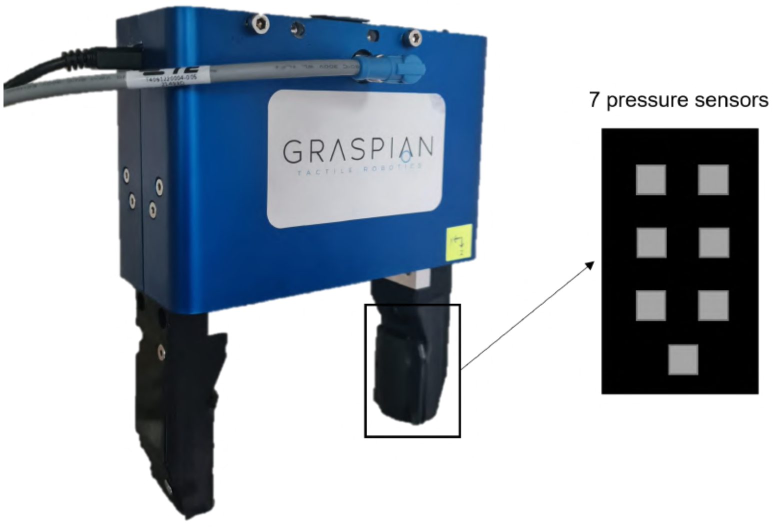
To access the camera, the camera control software must be installed to obtain the device identification number. In addition, the Python library *pyrealsense* must be installed to work with the vision classification models.

4.3.2 Gripper

The gripper was developed as part of a master's thesis by a former master's student in collaboration with a company named Graspian[24]. The student worked on various aspects of the gripper:

- Test the pressure sensors.
- Calibrate the sensors.
- Design the hardware circuit for the gripper.
- Transform the output information from the sensors to understand it as a contact force perceived by the sensors.
- Build an interface based on Robot Operating System (ROS) to obtain the responses from the sensors and send some commands for the gripper to actuate the motors in a specific way.

Fig. 4.3 shows the components of the gripper and a screenshot of the interface.



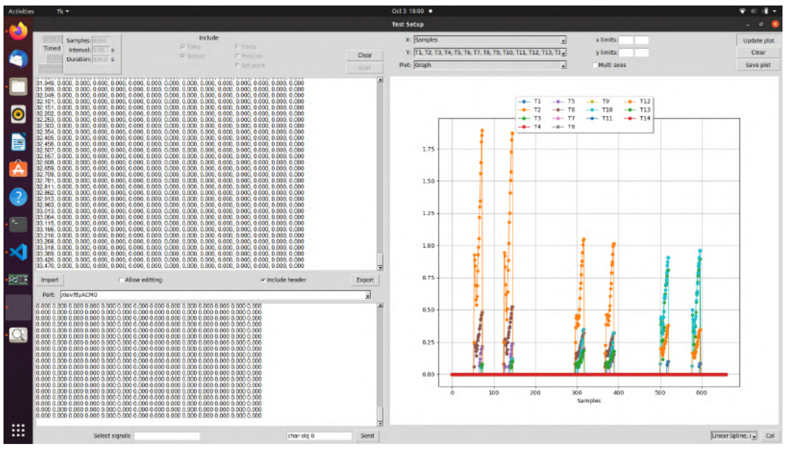


Figure 4.3: Elements and interface of gripper from Graspian.

It is important to understand the software that underlies the gripper to work with it. The gripper uses a Teensyduino, a programmable microcontroller, as its main board. The programming language used for the Teensyduino is Arduino. For its implementation, it is necessary to install the compatible Arduino software on the computer. The Teensyduino also has a "Serial.print()" function that allows access to the information inside the microcontroller and a "Serial.write()" function that enables sending information to the gripper to perform a specific action.

Additionally, when using Visual Studio Code (VSCode), it is necessary to download a PlatformIO (PIO) extension. This extension is a professional development environment that supports different operating systems, including Teensy.

4.3.3 Computer

The project is developed using their own Lenovo MT 81YH IdeaPad 5 14IIL05 computer for this project. The computer specifications include an Intel(R) Core(TM) i7-1065G7 CPU running at 1.30GHz with 1498 MHz and 4 physical and 8 logical processors. The operating system used is Ubuntu 20.04 and the code editor used is Visual Studio Code (VSCode). The main programming languages used in the project are C++ and Python.

Robot Franka Emika

Although the manipulator's arm is beyond the scope of the intention of this project, it is relevant to mention it as it appears in the image obtained from the camera. Also, it could be used in future works.

Conveyor

This element is a black rubber conveyor on which the different objects to classify are. This conveyor is the reference scene in the image. Only those objects positioned in this area are considered objects to classify.

Chapter 5

Methodology and Results

5.1 Tactile test design

There are different methods for gaining knowledge of an object through tactile interaction. In the field of robotics, the goal is to approximate human perception of actions performed by robots in the most appropriate way possible. To understand the previous approaches, it is necessary to visualize the various types of information that can be obtained through tactile contact, as depicted in Fig. 5.1, where an outline represents this information.

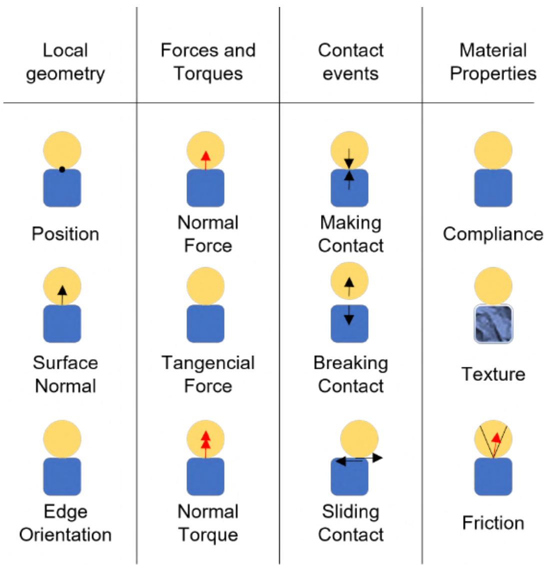


Figure 5.1: Different types of contact-level data that can be extracted from sensor-level tactile signals are illustrated. Figure inspired on [28].

In this context, a tactile test is the action of using the gripper to get the information from the object. There are different approaches depending on the characteristics of the gripper used. If it was possible to obtain all this information with some sensors, a possible tactile test would be, for example, to slide the gripper over the object to create an image of its shape. Another possibility, in the case of soft grippers, would be to create a sensory image when the object is grasp. However, the gripper selected for this project has limited sensors. These are only capable of getting contact information in the form of an event and a force. Fig. 5.2 shows the chosen test for the project.

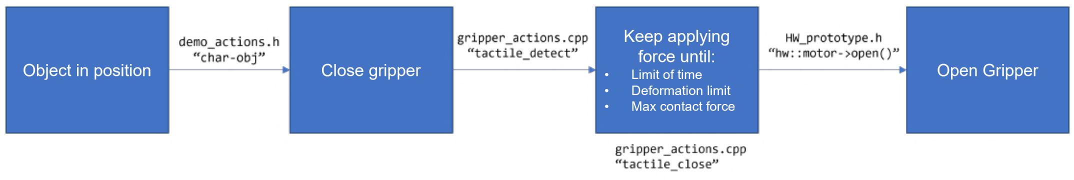

Object in position
demo_actions.h "char-obj"
Close gripper
gripper_actions.cpp "tactile_detect"
Keep applying force until:
Limit of time
Deformation limit
Max contact force
gripper_actions.cpp "tactile_close"
HW_prototype.h "hw::motor->open()"
Open Gripper

Figure 5.2: Gripper test description.

The data obtained from a single object can vary greatly depending on its position during the test. For example, if the object is positioned on the outer edge of the gripper, the pressure sensors located deeper within the gripper may not detect any contact force. Conversely, if the object is positioned closer to the center of the gripper, all sensors are more likely to register a contact force.

5.2 Vision-tactile-based algorithm

The implementation used seeks to exploit the characteristics of the sensors. As the camera is positioned outside the working area, it can classify the objects before working with the touch sensor. Therefore, it is decided to proceed with a linear implementation, as shown in the Fig. 5.3.

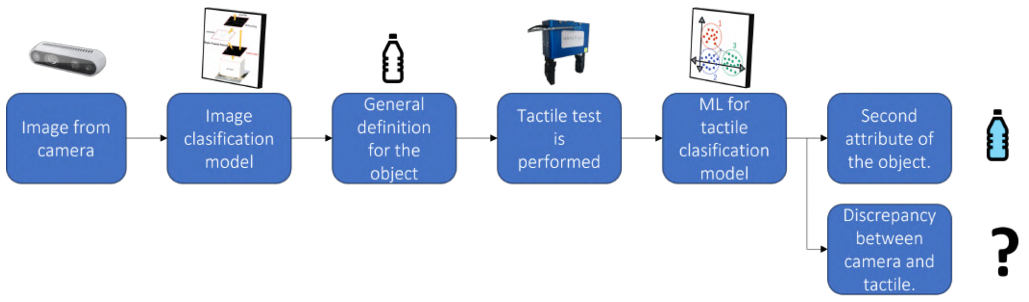


Image from camera
Image clasification model
General definition for the object
Tactile test is performed
ML for tactile clasification model
Second attribute of the object.
Discrepancy between camera and tactile.
?

Figure 5.3: Design of the object classification algorithm.

Regarding the operation and characteristics obtained by the sensors, the camera is in charge of finding the object type (related to the object's shape). For example, if it is a full plastic bottle, the Faster R-CNN should detect that there is a bottle in position. The signal is then sent indicating that an object has been found. Information on the type of object found is also sent. It is at this point that the gripper is moved to the position of the object. This movement and calibration of the robot are out of the scope of this project, but it is considered that this task could be done automatically in the future. Therefore, after the gripper is placed in the correct position and performs the tactile test, we get the second feature. Following the example, the result obtained would be, f.e. "full". The touch test aims to obtain a more specific tactile feature. This characteristic is not appreciable by the vision-based algorithm. Fig. 5.4 shows two pictures of a full an an empty plastic bottle. This objects are not easy to differentiate by a picture.

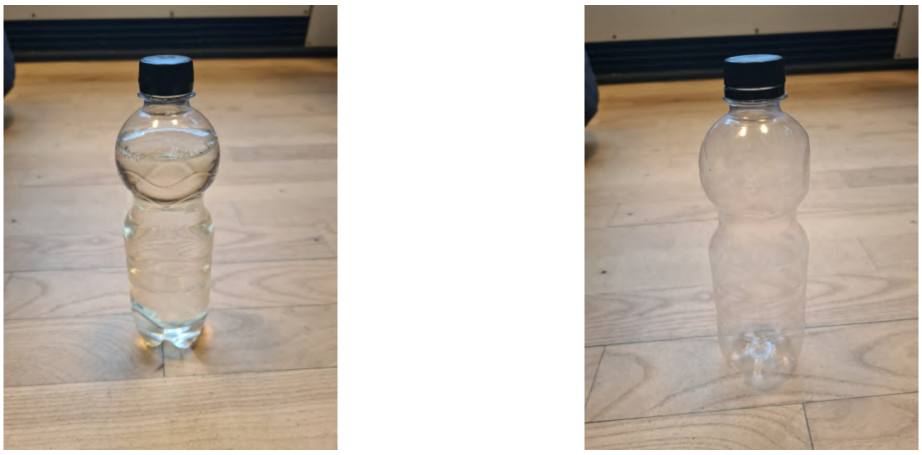

(a) Full bottle. (b) Empty bottle.

Figure 5.4: Images of the two different bottles.

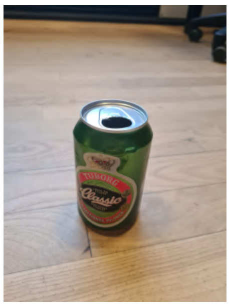
Additionally, to enhance the self-diagnosing capabilities of the algorithm, certain implementations have been made. As this is an experimental test and the external conditions of the environment may vary, the possibility of a discrepancy between the camera and touch sensor detections is taken into account. Furthermore, the camera may not detect the object once it has been placed on the conveyor. The idea would be to implement a loop for the system to work until it detects an object.

5.3 Metrics and experiments for algorithms

The metrics used will be based on accuracy. That means that it will give a percentage of the number of times that the algorithm has performed meeting the requirements out of the total amount of times that it was tested. Nevertheless, as the project develops two algorithms and the implementation of both together, the experiment and the succeed requirements are different for each.

To carry out these experiments, 6 objects are selected. These objects are shown in the Table 5.1 Objects with different physical characteristics (bottle, ball, can) and discriminating hardness characteristics (full-empty, hard-soft) have been chosen. The experiments will perform the test 20 times for each of the objects.

Table 5.1: Selected objects for the experiments.

	Bottle	Can	Ball
Hard			
Soft			

Faster R-CNN requirements and experiment.

The experiment will start the camera, look for an object an identify it. For this experiment, the test is consider to be a success if:

- The mAP value is over 70 %.
- The test takes less than 10 seconds.
- The object is detected in 3 or less tries.

ML requirements and experiment.

The experiment will be divided in two parts: first giving the right type of object detected to the algorithm (14 time per object) and then giving a wrong one (6 times per object). For this experiment, the test is consider to be a success if:

- The test takes less than 10 seconds.
- In case of sending the corresponding type of object, the object is detected correctly in the first try.
- In case of sending the a different type of object, the object is detected as "something else".

Thanks to this experiment, the best ML algorithm will be chosen for the experiment with all the algorithm.

Vision-tactile-based algorithm requirements and experiment

The whole implementation explained in 5.3 is performed for each of the objects. For this experiment, the test is consider to be a success if:

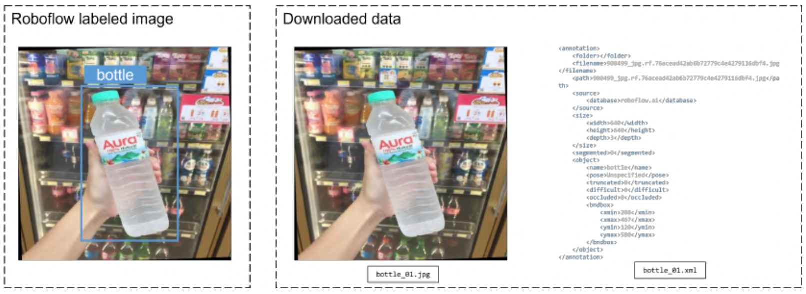
- The object is detected by the camera in 3 or less tries.
- The test takes less than 30 seconds since the command is sent.
- In case of disagreement between tactile and visual sensors, the object is detected correctly when is tested again.

5.4 Data preparation

Once the objects to be worked with are selected, data is collected about these same objects.

Visual data preparation

The visual data preparation is divided in two process. The first step is to group 100 images of each of the objects (bottle, can and ball). This data has to be large and diverse in order to get more accurate results and less biased model. Once the images are selected it is necessary to mark the box and label each of the images. In this case, this task was done thanks to the Roboflow app. The advantages of this app are that it allows to label the images, add the corresponding boxes and perform different transformations on the images in order to have a not over-fitted model. Once the dataset is created, it is necessary to download the data in the required form, in this case in *.xml* file. In Fig. 5.5 it is shown an example of the *.xml* file. Each of the images used has a correspondant *.xml* file with the label information.



```
<annotation>
    <folder></folder>
    <filename>900499_jpg.rf.76acead42ab6b72779c4e4279116dbf4.jpg
</filename>
    <path>900499_jpg.rf.76acead42ab6b72779c4e4279116dbf4.jpg</pa
th>
    <source>
        <database>roboflow.ai</database>
    </source>
    <size>
        <width>640</width>
        <height>640</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>bottle</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <occluded>0</occluded>
        <bndbox>
            <xmin>288</xmin>
            <xmax>467</xmax>
            <ymin>120</ymin>
            <ymax>580</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 5.5: Labeled data and labeled data file.

Tactile data preparation

It is necessary to perform the tactile test (Fig. 5.2) on the different objects to get the tactile dataset. This test is performed 100 times for each of them. The results of each of these give rise to a graph similar to the Fig. 5.6. This plot represents the contact force measured by each of the pressure sensors throughout the experiment. Every 300 ms new measurement is taken.

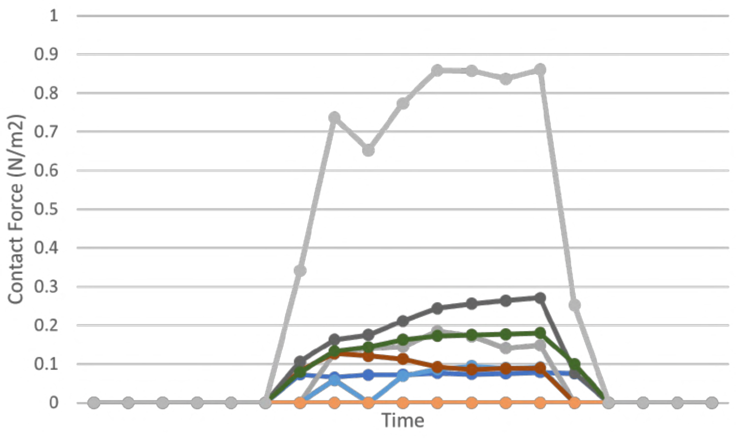
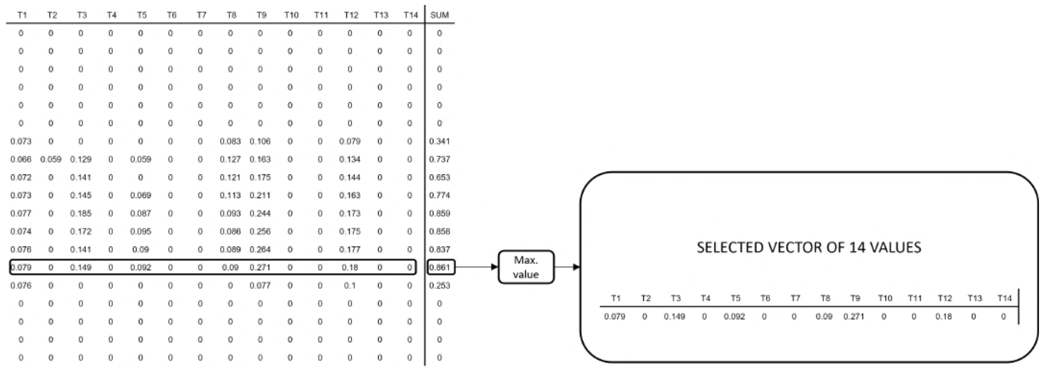


Figure 5.6: Plot of contact forces measures once tactile test is performed on empty bottle.

The data must be simplified to work with tactile data. The present form of the data is a matrix. This matrix counts with 14 columns and as many rows as the amount of data taken over time. It is decided to simplify this form by choosing as representative data the row whose values add up to the highest number. This vector will be the data representing an object. This data cleaning is explained in Fig. 5.7.



T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	SUM
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.073	0	0	0	0	0	0	0.083	0.106	0	0	0.079	0	0	0.341
0.066	0.059	0.129	0	0.059	0	0	0.127	0.163	0	0	0.134	0	0	0.737
0.072	0	0.141	0	0	0	0	0.121	0.175	0	0	0.144	0	0	0.653
0.073	0	0.145	0	0.069	0	0	0.113	0.211	0	0	0.163	0	0	0.774
0.077	0	0.185	0	0.087	0	0	0.093	0.244	0	0	0.173	0	0	0.859
0.074	0	0.172	0	0.095	0	0	0.086	0.256	0	0	0.175	0	0	0.858
0.076	0	0.141	0	0.09	0	0	0.089	0.264	0	0	0.177	0	0	0.837
0.079	0	0.149	0	0.092	0	0	0.09	0.271	0	0	0.18	0	0	0.861
0.076	0	0	0	0	0	0	0	0.077	0	0	0.1	0	0	0.253
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
0.079	0	0.149	0	0.092	0	0	0.09	0.271	0	0	0.18	0	0

Figure 5.7: Cleaning tactile data descriptive diagram.

The training of the tactile-based algorithm could be done in different ways. One choice could be to train the algorithms with the experimental data of all objects. This option would lead to more errors, as it would be necessary to classify according to object type and secondary features, resulting in more clusters. Another option would be to use the information obtained from the vision-based algorithm. Moreover, only the tactile data of the objects classified as that object type could be used. That option does not allow checking if there is a discrepancy between what the tactile sensor and the camera understand, so another approach is sought.

Therefore, it is decided to create a different dataset for each object type. The dataset for each object type is divided into three classes. For example, if the first algorithm detects a bottle, a dataset will be created with the following classes: full bottle, empty bottle, and "Something else" (other object type). The first two classes will contain all of the previously obtained data (100 data vectors per class), and the "Something else" class will consist of a set of data from other previously tested objects (the same number of data vectors for each object). This dataset aims to have the same weight as the others, so it is created with 100 data vectors to avoid overfitting in the system and to ensure balance.

5.5 Implementation

This section explains the implementation carried out to develop the project. This part is divided into three components: file overview, libraries used and selection and training of the algorithms.

Files overview

The outline below shows the structure of the files used for this project. Each of them is described next to their names.The files are uploaded in Github. Just some of them that are explained here are not uploaded due to large memory requirements.

```
object_classification_vision_tactile/
|- .gitignore                  large files are ignored on github,
                               but will appear for the explanation
|- README.md                   explanation on how to work with the
                               code and possible requirements
|- object_classification.py    main code, conection with gripper
                               and camera
|- Camera/
| |- __init__.py
| |- change_xml.ipynb          change the name of the object on xml
                               files for labeling images
| |- connect_realsense.py      test that the camera is connected
                               and cable working
| |- connect_to_hpc.txt        text file to explain the steps to
                               connect to HPC
| |- results_camera/
| | |- __init__.py
| | |- model.pth               saved model from the training (not
                               uploaded on Github)
| | |- inference.py            connect to camera, get image, get
                               the image classification
| | |- models/
| | | |- __init__.py
| | | |-                       description of number of classes
create_fasterrcnn_model.py
| | | |-                       description of the model
fasterrcnn_resnet50_fpn.py
| | |- utils/
| | | |- __init__py
| | | |- annotations.py        change the image and add the
                               information from the classification,
                               thus class and mAP
| | | |- general.py            training oriented
| | | |- transforms.py           change image from camera to the
                                 right format for the classification

|
|- Gripper/
| |- GraspianGripperSoftware/    interface to work with Gripper -
                                 developed by Jacob Fiskaali
| |- gripper-software/           PIO software to implement on
                                 teensyboard.  Deeper explanation
                                 on C
| |- object_classification/
| | |- __init__.py
| | |- cleaning/                 used in not real time to get the
                                 dataset
| | |- Classifiers/
| | | |- KNN.py
| | | |- SVM.py
| | | |- tree_classifier.py
| | | |- import_serial.py        send command to perform the test and
                                 returns the vector value
```

The connection between the explained files for object-classification by vision-tactile-based algorithm is shown in Fig. 5.8

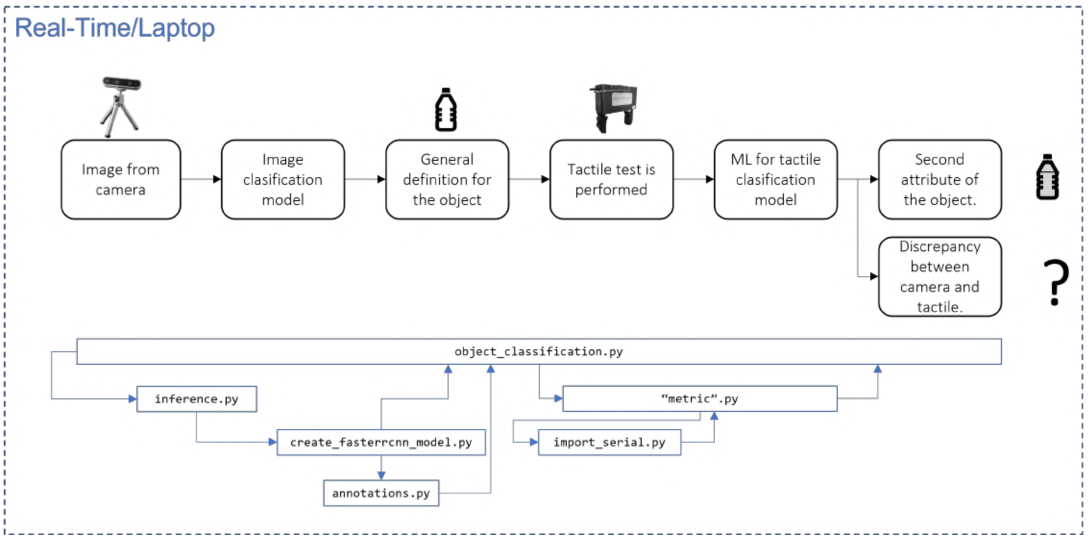

Real-Time/Laptop
Image from camera
Image clasification model
General definition for the object
Tactile test is performed
ML for tactile clasification model
Second attribute of the object.
Discrepancy between camera and tactile.
?
object_classification.py
inference.py
create_fasterrcnn_model.py
annotations.py
"metric".py
import_serial.py

Figure 5.8: Diagram of connection between files for object-classification.

Python Libraries

Since multiple algorithms are used for this project, it is necessary to download different libraries to work with Python. Moreover, when working with elements such as the camera and the gripper, it is necessary to have special libraries to communicate with it. The main used libraries for this project are shown and explain below.

pyrealsense2 it is the python library to connect with the cammera. It is developed by Intel in order to work with the camera that they designed.

cv2 also known as opencv for python. This is a computer vision library. In this project this is used to show the images obtained from the camera and modify them by adding the results from the Faster R-CNN model.

serial this is the library to connect with serial port. The projects needs this one for the connection with the gripper. Through this library, commands are sent to the Gripper to perform in a specific way.

pandas it is the most known library for data analysis and manipulation. This projects takes the advantage of using it to clean the tactile data in not real time. Furthermore, it is also used to access the tactile dataset to train the ML algorithms.

numpy work with arrays. In the project, it is used for the processing of tactile data, as well as for processing the results from the visual classification algorithm.

Pytorch: torchvision torch library for computer vision modeling. In specific, oriented to image classification modeling. It is used to work with the predefined model for faster_rcnn_resnet50_fpn.

albumentations image increasing the size and diversity of labeled trainings.

wandb this library is an API to connect to the web Weights & Biases. This web allows to save the deep learning models that are trained on GPUs.

sklearn tool for predictive data analysis. Decission trees, K-Nearest Neighbours and Support Vector Machines are algorithms that are already predefine by this library.

Others some other libraries used in this project: *matplotlib, jupyter, Pillow, pyYAML, scipy, tensorboard, torchinfo, xml...*

Selection and training of the algorithms

Tactile-based algorithms

The selected algorithms to implement are previously explained in Section 3.2. But in addition to picking the algorithms, we have to choose parameters.

In the case of the Decision Tree,the method used to calculate impurity of nodes is entropy. It is decided that there should not be a minimum number of samples to separate the nodes. Thus, a complex decision tree.

In the case of KNN, the distance used to find the closest point should be taken as a parameter. In the implementation, three different distances will be used, depicted in Fig. 5.9.

Euclidean

$$D_E(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_{14} - q_{14})^2}$$

Cosine

$$D_c(p,q) = \frac{\sum_{i=1}^{n} p_i * q_i}{\sqrt{\sum_{i=1}^{n} {p_i}^2} * \sqrt{\sum_{i=1}^{n} {q_i}^2}}$$

Mahalanobis

$$D_M(p,q)^2 = (p - q)^T * C^{-1} * (p - q)$$

C as covariance matrix of independent variables

Figure 5.9: Equations to calculate the three proposed distances for KNN.

For the last ML algorithm SVM, three options are selected to take the lines that will separate the data (linear, cubic and quadratic).

There are different models depending on the information given by the class obtained by vision. Since the tactile models are quick to create, it is decided to train these in real-time. To select the most appropiate algorithm, the idea is to get the percentage of the acccuracy when it is performing with the correct type of object given and than calculate the percentage when it is working with the wrong type given. The total accuracy would be the mean of those two values.

It is important to have in mind that the algorithms are design attending to a dataset of 3 classes: hard object type, soft object type and another object type.

Vision-based algorithms

Although only the Faster R-CNN is used for the classification by vision, there are different parameters to specify for its training:

- Number of epochs: number times that the learning algorithm will work through the entire training dataset.
- Number of batches: number of samples to work through before updating the internal model parameters.

Although the backbone and the RPN could have been trained, it was decided to use a pre-trained model as a starting point. These neural networks are big enough and the type of project developed suggests starting from pre-trained models. Nonetheless, this training is computationally expensive. Therefore, in this project the decision was to connect to the DTU's High Power Computers (HPCs). A HPC is a technology that uses clusters of powerful processors working in parallel to process massive multidimensional data sets (big data) and solve complex problems at extremely high speeds. In addition, they typically run on GPUs rather than CPUs, which makes this training faster. The Fig. 5.10 shows how the connection is made in this case and how the model is obtained.

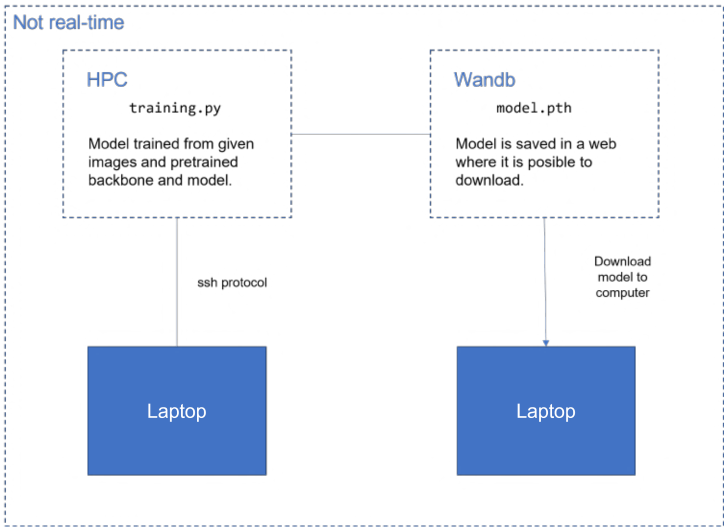

Not real-time
HPC
training.py
Model trained from given images and pretrained backbone and model.
Wandb
model.pth
Model is saved in a web where it is posible to download.
ssh protocol
Download model to computer
Laptop
Laptop

Figure 5.10: Training of Faster R-CNN through connection with HPC.

Every time that it is needed to add new objects to be identified by the neural network, it is necessary to re-train it.

Chapter 6

Results

Having previously outlined the format in which the tests are conducted, the software structure of the project, and the experiments that were performed, we proceed to present the results.

6.1 Test on laboratory objects

Faster R-CNN results

To train the algorithms, images of cans, bottles, and balls are utilized. The dimensions of the images are also a parameter that is set by the user. In this case, the same images (only with different size) are used to create three models:

Algorithm 64_100_32 Image size: 64x64, Number of epochs: 100, Batch size: 32

Algorithm 640_100_16 Image size: 640x640, Number of epochs: 100, Batch size: 16

Algorithm 640_200_32 Image size: 640x640, Number of epochs: 200, Batch size: 32

Once the model is trained and implemented, it is tested with different images from the Internet, distinguishing between can, ball and bottle. The results are shown in Fig. 6.1

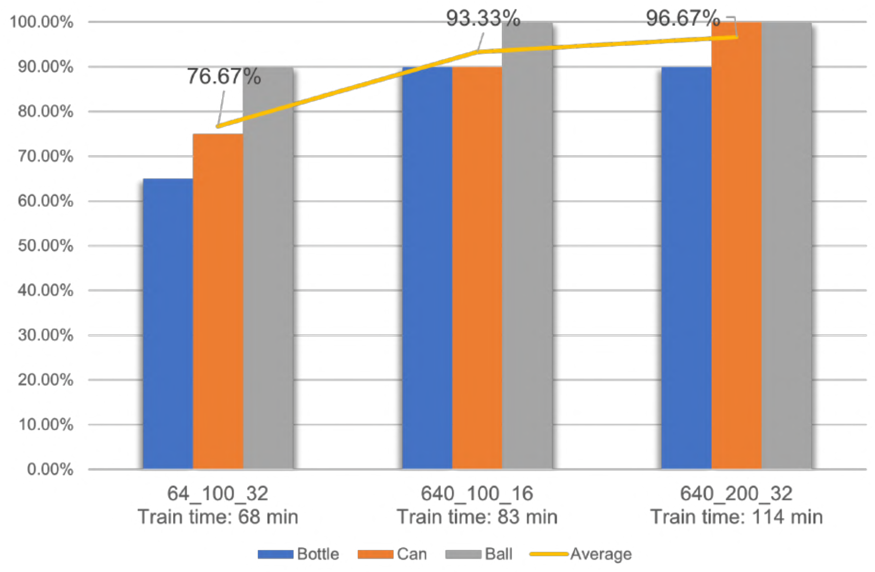


Figure 6.1: Faster-RCNN accuracy results on test images for the three trained models.

After this results are shown, apparently it seems to work better for the model that was tested with larger images and a higher number of batches and epochs. However we perform another test but now with the objects from the lab. The image test described in 5.3 is performed, and the results are shown in Table 6.1. Image results of each modle are shown in Appendix E.4.

Table 6.1: Faster R-CNN models performance on laboratory images.

Algorithm	Score	MTP (s)	mAP
64_100_32	98.33%	3.222	91.78%
640_100_16	100.00%	3.137	97.75%
640_200_32	100.00%	3.149	97.95%

The results indicate that using larger images improves the accuracy of the algorithm. Furthermore, increasing the number of training epochs and batches improves the algorithm's performance but increases the computational load during training. Despite this, the response time of the algorithm remains unchanged. Results for each of the tested objects are shown in Appendix E.1.

Machine Learning results

Upon implementation of the algorithms, an evaluation of their accuracy was conducted as previously outlined in 5.5. The results of this evaluation, utilizing objects in a laboratory setting, are presented in the accompanying Fig. 6.2. The algorithms depicted in the image are:

Tree	Decision tree with entropy
KNN1	KNN with euclidean distance.
KNN2	KNN with cosine distance.
KNN3	KNN with mahalanobis dist.
SVM1	SVM with linear metric.
SVM2	SVM with quadratic metric.
SVM3	SVM with cubic metric.

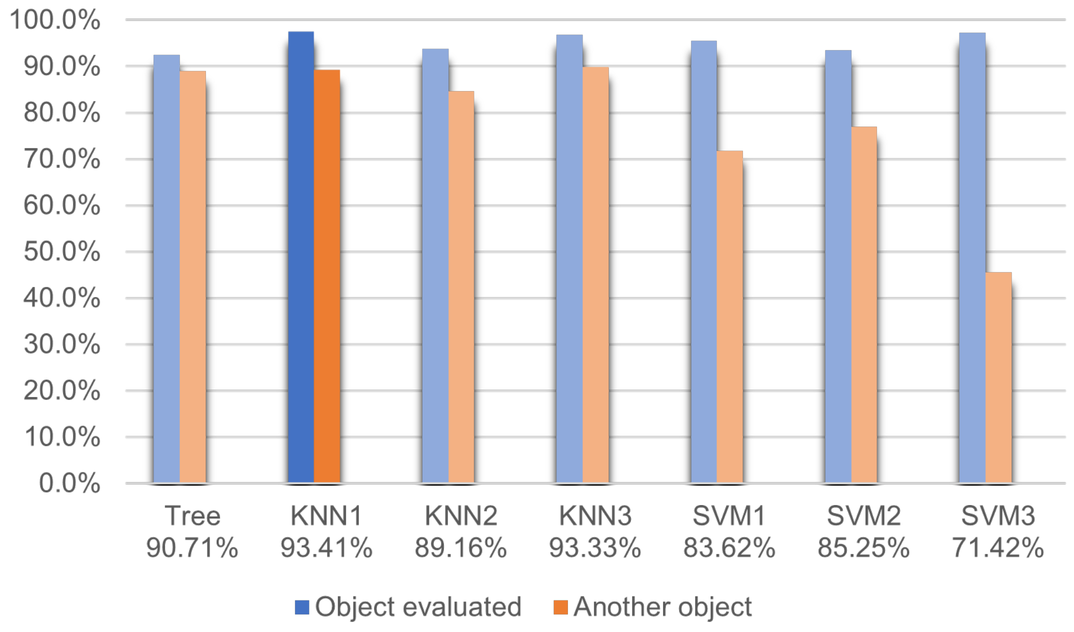


Figure 6.2: Accuracy results for Machine Learning algorithms.

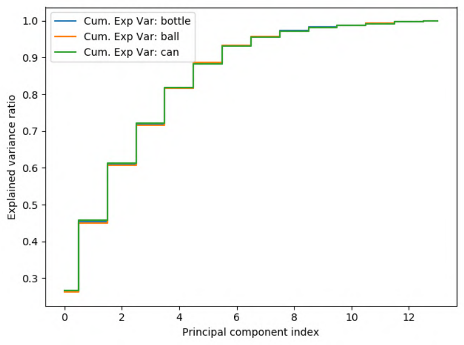


Figure 6.3: PCA exp. variance.

As the results demonstrate, the KNN1 algorithm, which uses a KNN approach with Euclidean distance as its distance metric, was the most successful. It was observed that the SVM algorithm performed significantly worse in comparison. This is due to the fact that SVM is typically used in two or three dimensions, and working with fourteen dimensions makes data division more complex. As a result, the possibility of implementing PCA on the data was considered, however, after analyzing the explained variance, this option was ultimately dismissed (as depicted in the Fig. 6.3). The figure illustrates three plots that pertain to the different datasets used to evaluate the algorithm's performance.

Vision-tactile-based algorithm results

Once the most appropriate models and algorithms for each sense were selected, the results of the combined algorithm were evaluated. Fig. 6.4 shows the results distinguishing between three different outputs in accuracy: success, failure and discrepancy. For the purpose of this project, we consider discrepancy as a successful result, because the system understand that the object is not define if there is discrepancy between the visual and the tactile-based algorithms. This would mean that both visual and tactile test must be performed again in order to identify the object correctly. The results obtained indicate that the system reduces errors in

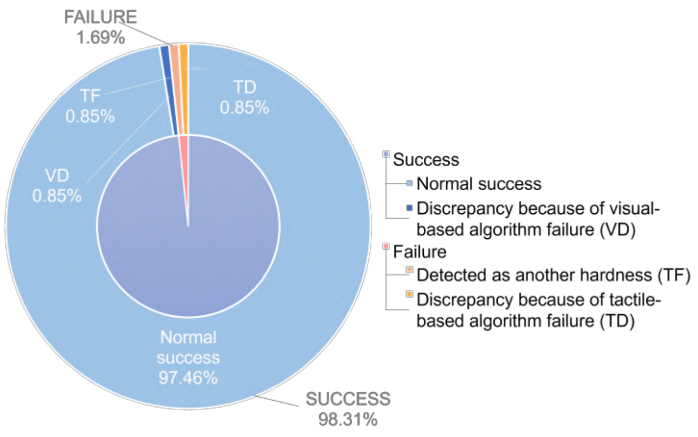


Figure 6.4: Accuracy test results on the vision-tactile-based algorithm.

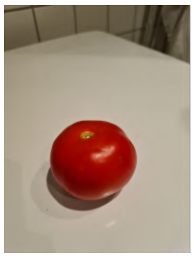
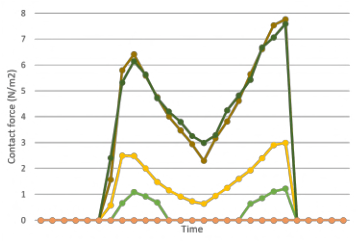
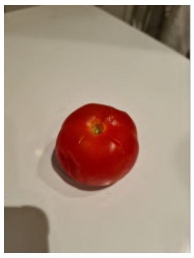
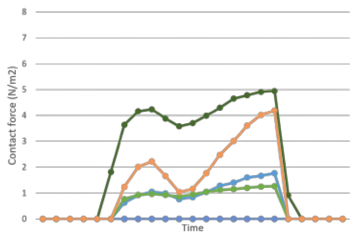
vision due to the use of touch. As cases of discrepancy will be re-analyzed, it can be considered that the system as a whole functions better due to the inclusion of the tactile sensor. Additionally, the characteristics that are not obtained through the vision algorithm have a small percentage of errors (1.5%). If a more specific analysis of the errors obtained in the experiments due to the vision and tactile algorithms is desired, a more detailed explanation can be found in the Appendix E.

6.2 Use case

To validate the algorithm's performance in an industrial setting, a use case involving the testing of tomatoes was proposed. This exercise was a reduced way of validating the project. Moreover, this experiment tested the selected algorithm ability of classifying tomatoes as being in good or bad condition. The ripeness of tomato classification was carried out solely by the tactile sensor. This means that even if the tomato looked ripped, it would be just classified as tomato by the camera. The aim of this experiment was to test that the system was able to detect the state of the tomato independently of its appearance.

Because of using an object that was not included in the test, the former visual and tactile datasets (Table 5.1) had to be enlarged. We added tomato images as visual data and previously taken information by the gripper as tactile data. Therefore, the dataset used for Faster R-CNN training now included 400 images and the dataset for the tactile-algorithm counted with 800 tactile data from eight different objects. Table 6.2 shows the tomatoes used for the experiment and the results of a tactile-data collecting test performed on them. As can be seen in the images, these tomatoes had different tactile characteristics, but very similar visual appearance.

Table 6.2: Selected added objects for use case experiment and contact force results when the test is applied on them.

	Tomato
Hard	 
Soft	 

Faster R-CNN results

The earlier chosen model (640_200_32) needed to be retrained because it included an object that was not considered before. Once the model has been trained, it is validated by the laboratory objects. The results obtained on this occasion are shown in the Fig. 6.5

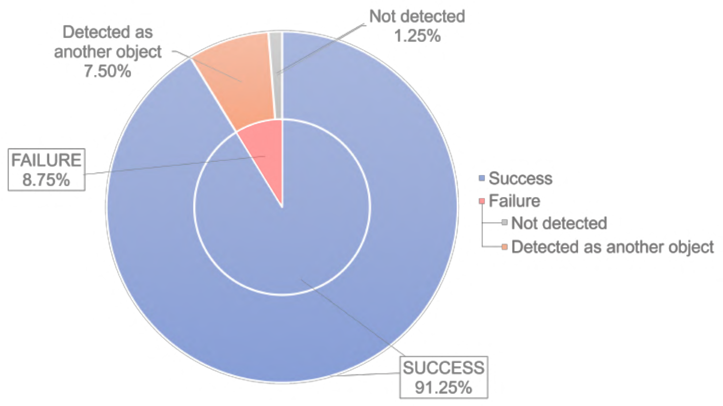


Figure 6.5: Accuracy use-case results on the vision-based algorithm.

As can be observed, even though the algorithm has been trained in the same way, when considering two visually similar objects (tomatoes and hard ball, which are both round and red), the algorithm was not able to distinguish them correctly. In some cases, the algorithm did not detect the object. This last happen because when the mAP is less than 70%, the classification given by the algorithm was not considered.

Machine Learning results

Since the algorithm selected during the test is the KNN with Euclidean distance, we proceeded to validate through the use-case with the same algorithm. The results are shown in Fig. 6.6

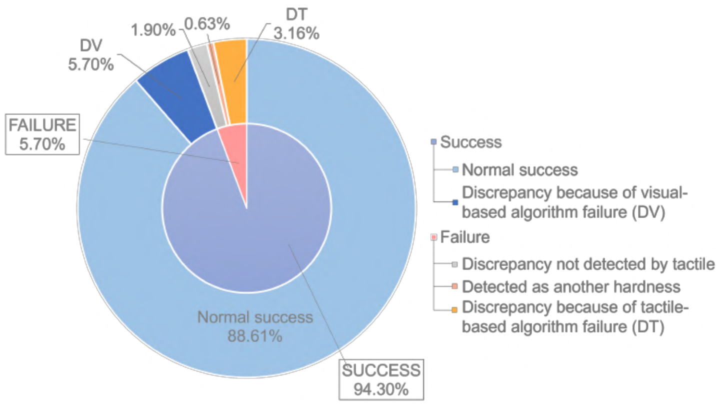


Figure 6.6: Accuracy use-case results on the tactile-based algorithm.

This test show a similar accuracy than the one obtained with the test done without considering the tomatoes in the dataset (94.3% against 93.41% that it was before). Nevertheless, according to the results, when the vision fails on misclassifying the target and the tactile is not detecting discrepancy, it results in a tactile-based algorithm failure, which should have detected discrepancy. In this use-case, this percentage is much higher than the test's (almost 90% before and 1.27%/5.7% = 77.7% now). This happened because the tactile test sometimes confused the response of the soft-ball with tomatoes and the other way around. Despite this misclassification, the success rate of the tactile-based algorithm classification is achieved with a decent accuracy (94.3%).

Vision-tactile-based algorithm results

The performance can be summarized with the diagram in Fig. 6.7

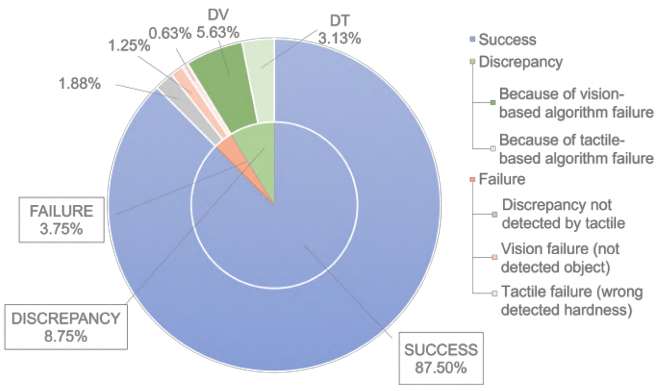


Figure 6.7: Accuracy use-case results on the vision-tactile algorithm.

In the previous test, when the tactile-based detection of discrepancy was assessed by itself, it was deemed a failure. However, upon evaluating the system as an integrated whole, it was determined that the detection of mismatch by the system can be considered a potential success. This is because if the system were to repeat the verification process, it is likely that the objects where the sensors differed would be correctly detected and classified. Therefore, the overall algorithm's success rate is calculated to be 95%. It is important to note that the visual component of the algorithm has a higher failure rate at 8.75%, compared to the tactile-based component which has a failure rate of 5.7%.

Chapter 7

Discussion

This chapter presents a discussion about the results obtained throughout the thesis. It is done by covering the aspects that have functioned correctly and pointing out the limitations that have arisen in the development of the project. This chapter describes the potential uses of the implementation. Moreover. it comments on possible solutions to the flaws that regard hardware elements (camera and gripper), and the design and usage of the algorithms.

7.1 Impact of vision-tactile based object classification

The project work consisted on implementing and testing different algorithms for object classification. The aim was to find the most appropriate solution toward a future robotic grasping task. The final solution consists of two object classification algorithms: Faster R-CNN and KNN that process vision and tactile input, respectively. The tests showed that the system accurately identify four types of objects and two object hardness classes.

The integration of both vision and tactile sensing is beneficial in multiple scenarios. Including a tactile-based object classification to the traditional vision-based object classification enhance the data. This implementation has three main advantages.

The first one is the ability to adapt to an environment of disparate items. In this sense, the classification aims to identify the objects and their characteristics. The results of the experiments showed that the system employed in this project can distinguish between different types of items, with an accuracy of 96.25%. In addition, it discriminates among two types of hardness, thus giving extra information that is not available from the camera.

The second main advantage is the possibility of choosing how to grasp the object depending on shape and material characteristics. Although we did not test the following objective, after performing the vision-based algorithm, we could send the type of object information to the robot, for example, a bottle. With this first attribute characteristic (bottle), some pre-defined shape characteristics would be sent to the robot in order to position the gripper in the right place to perform tactile test. Once the tactile test is performed, the tactile-based algorithm could get the hardness results. With this results, we could send to the gripper the force that it would have to exert to grasp the object without losing or damaging it.

The third advantage is to be able to identify that the vision-based algorithm has not gotten false positive results. If the objects to be classified have a similar shape, it is possible that the camera gives the wrong results. In our selected use case, the results were worse than they could have been because of using two different round objects (tomatoes and balls). The camera was sometimes confused, but the tactile-based algorithm allowed to determine that the grasped object belonged to a different category than the one indicated by the vision-based algorithm.

Therefore, thanks to such a system, a robot could handle various items (hard or soft) in a more precise and compliant manner. Because of this reason, the system could be applicable to different industrial sectors, for example (recycling plants, food industry).

However, when it comes to handling identical objects, this type of arrangement does not seem to have much in its favor. This system involves an economic and computational overhead that can sometimes be solved more efficiently. A good example where it would not be useful would be industries manufacturing identical and not very fragile objects. In this case, the way of picking up the object and identifying whether or not it is defective can be done by selecting specific gripper for the industry, like 3D printed grippers [1].

7.1.1 Possible applications

Based on the understanding of the pros and cons of the system, the following features can be identified as necessary for any potential application to make the developed system practical and usable.

- Application counts with different objects according to its shape.
- Application needs information of the hardness of the object

[1] `https://www.ennomotive.com/robot-grippers-industrial-applications/`

- Application needs to understand which object is the target to leave it in a determined position.

According to this features, one application could be food sorting. In the food sector, products have many different shapes. It could classify the food into good or bad condition depending on its hardness. In the use case development, the system is able to distinguish between two tomatoes with similar physical characteristics, but one of which has the properties of overripe food. According to the results from the test, when vision-based algorithm correctly identify the tomatoes, the tactile-based algorithm can confuse it with a tennis ball, but is never confused distinguishing between the tomatoes hardness (See Table F.1). In the same way, handling different vegetables could be carried out in a realistic scenario. This could be done by depositing them in the determined area. It is a good idea to use a fixed gripper for this tasks, as one of the main advantages of using IT instead of a soft gripper is precision tasks.

Picking up unknown objects is one of the most complicated handling tasks, specifically if this task is performed outdoors, where spatial conditions may change. Therefore, a second application could be the recycling sector. The carried out tests through this project have validated the identification of objects such as cans and bottles. For this purpose, it would be possible to identify such items and determine whether they are full or empty to recycle or store for later consumption.

Finally, a third application could be in contexts where the camera can not be placed in the workspace and the gripper is able to access these areas. A good example would be an oven in which the manipulator arm could to work but the camera could not withstand such extreme temperatures. Therefore, thanks to the tactile sensor, we could determine whether the gripped object is the one previously seen by the camera or if there is a delay or malfunction of the system.

7.2 Gripper performance and limitations

Thanks to the experiments, it has been observed that the gripper is a simple element in terms of handling and control.It is possible to send force commands to be applied to the motors, with the aim of picking up objects with a specific force. The tactile sensors attached to the gripper provide simple measurements. Since the readings are straightforward, good results have been obtained even using simple algorithms. This information is enough to classify the same type of object into two classes according to its hardness and to indicate when the camera has failed to classify it.

Despite the fact that this element has worked and fulfilled its objective, this project rules out the possibility of industrial use, as two main problems have been detected during the development of the project. Firstly, the sensor measurements have changed their values over time, due to a more intensively used and changes in temperature. Accordingly, the results obtained by the subsequent algorithm were not trusted. The only way to rely them would be to create the data set shortly before the experiment and under similar conditions. On top of that, during the tests, the gripper broke due to a motor disengaged on two occasions from the gear, slowing down the course of the project.

In addition, if we wished to create more classes based on the hardness presented in the objects, this gripper would not have been able to distinguish between cases. To improve the outcome, a possible experiment would be to use a soft-gripper. The soft nature of the gripper allows the adaptation of the grasping to the object shape. Otherwise, defining a map of the contact forces received from each part of the object could allow a fusion of deep learning by images between vision and tactile senses [23].

7.3 Camera performance and limitations

At the beginning of the project, we tried a camera without an RGB module (BlueFox3). The connection to this camera was quite complicated, and the images were not clear enough to work with classification. There were complications with the set of this camera because it was looking at the objects from the top, meaning that the objects were hard to identify (See G.3). Moreover, we could not change the position because another work was using this camera. Therefore, the decision was to change the camera and work with the one finally implemented, the Intel Depth camera. Understanding the operation of this camera was a much simpler process, as it had direct indications from the company that created it (Intel RealSense Depth Camera). In addition, this camera does have the RGB module, which is useful in classification problems. On the other hand, the obtained images from this camera are of good quality, and the camera calibrates itself automatically. Moreover, regardless of the time of day and independently of the light in the laboratory, the images change only slightly so that the obtained data is robust.

However, this camera presents some disadvantages. Images with more information are more susceptible in real-time, thus resulting in an increased communication time with the PC. That is the reason why, with a 5-meter cable connection, the system started to crash. However, we could solve this issue by using a shorter wire.

In order to get more benefits for the project, we could have exploited even more the Intel depth camera. Because of the two incorporated cameras, we could have worked with depth and made a deep learning model using a dataset that includes this information (RGBD) [29]. The objective of analyzing depth would be to determine at what distance the objects we want to work with should be. This distance measure would help to avoid identifying objects belonging to the background as objects we are interested in this.

Another possible improvement for the project would be to re-design the camera position within the workspace. This change is linked to the future implementation of a project with the robot because, from the current location, it is difficult to estimate the position to which the robot should go. Adding depth information could help to find a more accurate arrangement. A better option to get the information for the robot would be to have the camera above the conveyor. However, this option would not be feasible because the objects would be difficult to recognize from this perspective.

7.4 General issues and further improvements

One of the main problems faced in this project is the training of the Faster R-CNN. As this has been done with the university's HPCs, it is not always possible to allocate memory to run the batch. This problem has delayed the tests. In addition, due to the computational load involved, it has not been possible to train the model with a batch size of 64, which would probably have improved performance.

On the other hand, a laptop via the CPU drives the system. According to previous experiments, the processing of a deep learning model on a CPU is approximately three times slower than if it were processed on a GPU [2]. The support of a GPU would improve the performance of the implementation and bring it closer to real-time application.

The used tactile-based algorithms are easy and work successfully. Therefore, with the sensors currently available and for the purpose of separating into two classes according to the object hardness, handling a neural network is ruled out. The reason for doing this is that it would increase the computational load and processing time unnecessarily. In case of modifying the scope and dividing the classes according to more than two types of hardness, this could be a suitable option.

The thesis work presented in this report has been carried out with the intention of merging it with the work developed at the same time by Nils Meile, a Master's

[2]`https://deci.ai/blog/close-gap-cpu-performance-gpu-deep-learning-models/`

student of Autonomous Systems at DTU. His project consisted of the development of a grasping control that rely on fused vision and tactile sensing. A camera is placed above the workspace, so the robot is able to move towards and pick up a target. In fact, the camera is only used to detect an object and then the tactile sensing allows the gripper to determine when it grasps or not the object. Both sensing clues are fused to determine an optimal grasping based on the object position and measured pressure. The object classification provided by this thesis would allow a more intelligent control grasping in the future.

Previous research [23] has suggested combining visual and tactile information through neural networks. However, the advantage of the current project, which incorporates a controlled robot, is that visual information is obtained in advance. Therefore, merging both senses is unnecessary in this setup as it would negate the benefit of obtaining the images beforehand. The purpose of obtaining these images is to determine the object's coordinates while the classification is in progress. This approach would be beneficial if the camera was placed on the manipulator rather than in the current setup.

Chapter 8

Conclusion

In this master thesis project, an object classification system based on shape and object hardness was implemented. To classify the shape, a Faster R-CNN algorithm was employed by providing visual sensing input while ML methods were applied to distinguish the hardness by means of tactile sensing input. With this aim, we used an RGB camera and a gripper endowed with pressure sensors.

Besides the object classification algorithms, we designed an appropriate test to obtain the tactile information. We opted for a single grasping test. This test stops when it detects an item and applies an extra force until it acquires the contact force of the object. The gripper applies the pressure without exceeding a determined deformation nor force. The test results format is a simple vector that allows the modeling of not complicated ML algorithms that can be used to interpret the hardness.

Regarding the tactile recognition, we implemented seven ML algorithms. After calculating their accuracy, the KNN with Euclidean distance was selected as the most fitting algorithm. Additionally, throughout the experiments, we discovered that the employed pressure sensors are not reliable, given that the measure is not replicable in a constant scenario. Hence, we conclude that they are not capable of being used in a non-academic context.

The Faster R-CNN has been trained at HPCs of the university. A different training was carried out by changing the image dataset and the quality of the data. The best training performance was obtained with the dataset comprising 640x640 pictures with applied image transformations (brightness, size, rotations, etc.). When the algorithm was tested, it was found to be accurate (91.25%) but also slow (around 3 s to classify and identify). The algorithm can not be used in real-time, at least not if it is run from a CPU. In addition, bringing in two types of visually similar objects leads to a large amount of error compared to having classes that are very different in shape.

The vision-tactile implementation has shown slightly better results than using only vision (96.25% compared to 91.25% accuracy from vision-based algorithm). The tactile-based algorithm gives better results than vision-based algorithm (94.3%) and is able to identify hardness differences, especially when dealing with an object of similar shape characteristics. All the implementation could performed iteratively even when a discrepancy between vision-based and tactile-based algorithms is detected.

Finally, the use case has shown that the system is suitable for the food sector. The validation results showed that the accuracy is good at identifying vegetables and identifying their current state condition. Moreover, a proper tactile test procedure ensure not to damage the product.

The conclusion of this thesis suggests that future work could involve integrating this project with another one to control a robot for grasping and releasing objects based on their characteristics. This has already been explored by a previous master's thesis student, Nils Meile, who utilized a top-view camera and the Panda robot from the lab. A potential solution could be to position the camera at an angle that enables object classification while also recognizing the object's position and orientation.

Bibliography

[1] Lorin Lachs. *Multi-Modal Perception.* 2022. URL: http://noba.to/cezw4qyn.

[2] Uehara T Takehana A and Sakaguchi Y. *Audiovisual synchrony perception in observing human motion to music.* 2019. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0221584#sec002.

[3] Tabrik S Behroozi M Schlaffke L Heba S Lenz M Lissek S Güntürkün O Dinse HR Tegenthoff M. *Visual and Tactile Sensory Systems Share Common Features in Object Recognition.* 2021. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8493885/.

[4] Bonci Andrea et al. *Human-Robot Perception in Industrial Environments: A Survey.* 2021. URL: https://www.mdpi.com/1424-8220/21/5/1571.

[5] Alhawaj AFz Marzvanyan A. *Physiology, Sensory Receptor.* 2022. URL: https://www.ncbi.nlm.nih.gov/books/NBK539861/.

[6] Hutmacher Fabian. *Why Is There So Much More Research on Vision Than on Any Other Sensory Modality?* 2019. URL: https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02246.

[7] Hongxin Zhang and Suan Lee. *Robot Bionic Vision Technologies: A Review.* 2022. URL: https://doi.org/10.3390/app12167970.

[8] Tadi P Huff T Mahabadi N. *Neuroanatomy, Visual Cortex.* 2022. URL: https://www.ncbi.nlm.nih.gov/books/NBK482504/.

[9] Scott A. Sheffield MS. *Touch Pathway.* 2022. URL: https://www.getbodysmart.com/sensory-system/touch-pathway/.

[10] Mounia Ziat Catherine L. Reed. *Haptic Perception: From the Skin to the Brain.* 2018. URL: https://www.sciencedirect.com/referencework/9780128093245/reference-module-in-neuroscience-and-biobehavioral-psychology.

[11] Lynette A. Jones. *Kinesthetic Sensing.* 2000. URL: http://bdml.stanford.edu/twiki/pub/Haptics/PapersInProgress/jones00.pdf.

[12] M. Valle R. S. Dahiya G. Metta and G. Sandini. *Tactile SensingFrom Humans to Humanoids.* 2009. URL: https://ieeexplore.ieee.org/document/5339133.

[13] Jiao C. Lian B. Wang Z. Song Y. Sun T. *Visualtactile object recognition of a soft gripper based on faster Region-based Convolutional Neural Network and machining learning algorithm.* 2020. URL: https://journals.sagepub.com/doi/full/10.1177/1729881420948727.

[14] Trinh Trinh & Iwamoto Yuki & Shibuya Koji. *Localization of Sliding Movements Using Soft Tactile Sensing Systems with Three-axis Accelerometers.* 2019. URL: http://dx.doi.org/10.3390/s19092036.

[15] Vinicius Prado da Fonseca et al. *Tactile object recognition in early phases of grasping using underactuated robotic hands.* 2022. URL: https://link.springer.com/article/10.1007/s11370-022-00433-7.

[16] Schneider et al. *Object Identification with Tactile Sensors using Bag-of-Features.* 2009. URL: http://dx.doi.org/10.1109/IROS.2009.5354648.

[17] MinKyo Lee Binayak Bhandari. *Haptic identification of objects using tactile sensing and computer vision.* 2019. URL: https://journals.sagepub.com/home/ade.

[18] B. Calli A. J. Spiers M. V. Liarokapis and A. M. Dollar. *Single-Grasp Object Classification and Feature Extraction with Simple Robot Hands and Tactile Sensors.* 2016. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7390277.

[19] Wang S-a et al. *Fabric Classification Using a Finger-Shaped Tactile Sensor via Robotic Sliding.* 2022. URL: https://doi.org/10.3389/fnbot.2022.808222.

[20] S. S. Baishya and B. Bäuml. *Robust material classification with a tactile skin using deep learning.* 2016. URL: https://ieeexplore.ieee.org/document/7758088.

[21] Yang Gao et al. *Deep Learning for Tactile Understanding From Visual and Haptic Data.* 2015. URL: http://arxiv.org/abs/1511.06065.

[22] Cagri KAYMAK and Aysegul UCAR. *Implementation of Object Detection and Recognition Algorithms on a Robotic Arm Platform Using Raspberry Pi.* 2018. URL: https://doi.org/10.1109/IDAP.2018.86209162.

[23] Reza Pebdani Babadian et al. *Fusion of Tactile and Visual Information in Deep Learning Models for Object Recognition.* 2022. URL: https://doi.org/10.1016/j.inffus.2022.11.032.

[24] Jacob Fiskaali Hertz. *Design and Implementation of Tactile Sensing for Robotic Gripper Applications.* 2020.

[25] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* 2015. URL: https://dblp.org/rec/journals/corr/RenHG015.bib.

[26] Ross Girshick. *Fast R-CNN.* 2015. URL: https://doi.org/10.1109/ICCV.2015.169.

[27] Tue Herlau, Mikkel N. Schmidt, and Morten Mørup. *Introduction to Machine Learning and Data Mining*. 2021.

[28] Q. Li et al. *A Review of Tactile Information: Perception and Action Through Touch*. 2020. URL: `https://doi.org/10.1109/TRO.2020.3003230`.

[29] Yihui He. *Estimated Depth Map Helps Image Classification*. 2017. URL: `https://doi.org/10.48550/arxiv.1709.07077`.

Appendix A

Sustainability

The Sustainable Development Goals (SDGs), also known as the Global Goals, are a set of 17 interconnected goals adopted by the United Nations in 2015. They serve as a blueprint for achieving a more sustainable and equitable world by 2030. The SDGs address a wide range of social, economic, and environmental challenges faced by countries worldwide.

The goals cover a broad spectrum of issues, including eradicating poverty and hunger, ensuring quality education and healthcare, achieving gender equality, promoting clean energy and sustainable cities, combating climate change, protecting ecosystems and biodiversity, and fostering peaceful and inclusive societies.

This project aligns with multiple SDGs, primarily SDG 9: Industry, Innovation, and Infrastructure, and SDG 12: Responsible Consumption and Production. Additionally, it has implications for SDG 8: Decent Work and Economic Growth.

SDG 9 focuses on promoting sustainable industrialization, innovation, and infrastructure. The development of a vision-tactile object classification system contributes to this goal by advancing the field of robotics and automation. By leveraging machine learning algorithms and computer vision techniques, the system enhances the capabilities of robots, leading to increased efficiency and productivity in industries. This technology can support the development of advanced manufacturing processes, helping create sustainable and resilient infrastructure.

SDG 12 emphasizes responsible consumption and production patterns. The vision-tactile object classification system contributes to this goal by optimizing the robotic grasping process. Accurate object classification enables robots to handle items more effectively, reducing errors, damages, and waste in production and logistics operations. By improving the precision of grasping, this technology promotes sustainable consumption by minimizing resource wastage and product losses during manufacturing and supply chains.

SDG 8 aims to promote sustained, inclusive, and sustainable economic growth. The implementation of a vision-tactile object classification system creates opportunities for job growth and skills development. It can lead to the creation of new employment opportunities in robotics and automation industries, as well as related fields such as machine learning and computer vision. This technology empowers workers by augmenting their abilities and enabling them to focus on higher-level tasks, leading to improved productivity and economic growth.

Overall, this project demonstrates the potential to address multiple SDG simultaneously by leveraging technological advancements to enhance industrial processes, promote responsible consumption, and foster economic growth. By aligning with the SDGs, it contributes to the broader agenda of sustainable development and helps create a more sustainable and prosperous future for all.

Appendix B

Faster R-CNN Loss function

In this section, we shall discuss the loss function utilized in the Faster R-CNN algorithm.The algorithm utilizes a multi-task loss function which is made up of four components: RPN classification loss (L_{cls}^{RPN}), RPN regression loss (L_{reg}^{RPN}), Fast R-CNN classification loss ($L_{cls}^{Fast-RCNN}$) and Fast-RCNN regression loss ($L_{reg}^{Fast-RCNN}$). The RPN classification loss is calculated as the cross-entropy loss between predicted objectness scores and true objectness labels. The RPN regression loss is calculated as the smooth L1 loss between predicted and true bounding box coordinates. Similarly, the Fast R-CNN classification loss is also calculated as the cross-entropy loss between predicted class labels and true class labels, and the Fast R-CNN regression loss is calculated as the smooth L1 loss between predicted and true bounding box coordinates. The formulas for each of the loss are shown below.

$$L = L_{cls}^{RPN} + L_{reg}^{RPN} + L_{cls}^{Fast-RCNN} + L_{reg}^{Fast-RCNN} \tag{B.1}$$

$$L_{cls}^{RPN} = -\frac{1}{N_{cls}} \sum_{i}^{N_{cls}} (y_i * log(p_i) + (1 - y_i) * log(1 - p_i)) \tag{B.2}$$

Where N_{cls} is the number of region proposals, y_i is the true objectness label and p_i is the predicted objectness score.

$$L_{cls}^{Fast-RCNN} = -\frac{1}{N_{cls}} \sum_{i}^{N_{cls}} (y_i * log(p_i) + (1 - y_i) * log(1 - p_i)) \tag{B.3}$$

Where N_{cls} is the number of region proposals, y_i is the true class label and p_i is the predicted class label.

$$L_{reg}^{RPN} = smoothL1(t_i - t_i^*) \tag{B.4}$$

$$L_{reg}^{Fast-RCNN} = smoothL1(t_i - t_i^*) \tag{B.5}$$

Where t_i is the true bounding box coordinates and t_i^* is the predicted bounding box coordinates. The smooth L1 loss is defined as:

$$smoothL1(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{B.6}$$

Appendix C

Gripper-software code description

Below is given a short description of each file or class file-pair in within the project. This description was developed by Jacob Fiskaali during his project development at Graspian company.

```
Gripper-software/
|- archive/                 old unused code
|- include/
| |-                        old user interface - still used by some tests
Publihser(deprecated)/
| |-                        old user interface - still used by some tests
UserCom(deprechated)/
| |- UserInterface/
| | |- Publishing/
| | | |- publishers.h       implementation of data that can be published to
                            user
| | | |- pubsub.h           class defining the subscription service
| | |- UserCommands/
| | | |-                    implementation of user commands regarding force
control_actions.h           control
| | | |- motor_actions.h    implementation of user commands regarding motor
                            control
| | | |-                    implementation of user commands regarding
optical_actions.h           optical sensor
| | | |-                    implementation of user commands regarding
tactile_actions.h           tactile sensor
| |-                        methods used for calibrating tactile sensors
tactile_calibration.h
```

```
| |- tactile_perception.h  methods used for computing higher-level tactile
                           information
|
|- lib/
| |- Executor/             wrapper for Teensy-thread
| |- ForceControl/         abstraction class for force controller
| |- ForceControllers/     implementations of force controller
| | |-                     controller used for detecting objects within
DetectionController/       gripper
| | |- PID_Controller/     controller used for applying constant grip
                           force
| |- Hardware/             definitions of hardware
| |- Led/                  class for controlling Tennsy led
| |- PushButton/           class for reading push button
| |- RS485_Module/         class for reading/writing through RS485 module
| |- TactilePerception/    methods used for computing higher-level tactile
                           information
| |- UserInput/
| | |- Listener            class for logging user input
| | |- Parser              class for parsing strings to user command
                           arguments
| | |- UserAction.h        abstraction class for user commands
| | |- variant.h           storage for variant variable types
| |- UserStream/
| | |-                     definitions regarding serial port and baud rate
serial_definitions.h
| | |- UserStream          warpper class for writing annotated outputs
|
|- src/
| |- main/                 files included when selcting env:main
| | |- gripper_actions     methods defining higher-level gripper
                           executions
| | |- main.cpp            file containing main methods setup() and loop()
| |- robotbrag/            files included when selcting env:robotbrag
| |- tests/                files included when selcting div.  test
                           environments
|
|- extra_script.py         script executed after upload.
|- platformio.ini          file describing environments and build options

Libraries/                 hardware-related libraries
|- Fingers/                definition of different finger designs
| |- F2221/                finger with 7 taxels (unused)
| |- F2221_2/              finger with 7 taxels (env:tactile_tests)
| |- F2221_demo/           finger with 7 taxels (unused)
| |- OPT_demo/             finger w.  7 taxels + opt.  sensor (env:main /
                           env:robotbrag)
| |- OPT_v1/               finger w.  7 taxels + opt.  mouse sensor
                           (env:optical_tests)
| |- PlusFingers_v1/       finger with 4 taxels
|- MotorControl/
| |- MC3001_Interface      definition regarding communication with MC3001
                           board
| |- Motor                 high-level class for controlling the motor
| |- MotorCom              class handling communication with MC3001 board
| |- MotorSpecs.h          motor-specific constants
|- OpticalSensor_mouse/    optical sensor class based on optical mouse
|- OpticalSensor_PAA5100/  optical sensor class based on PAA5100 module
                           (current)
| |- Bitcraze_PMW3901_mod  modified arduino library (unused)
| |- OpticalSensor         used implmentation
|- Streaming/              arduino library for using '«' operator
|- TactileCircuit/
| |- Circuit_2x7           class regarding tactile meassuring
| |- CircuitBoard          definition of circuit version and signal order
| |-                       analog definitions related to Teensy
hw_analog_definitions.h
| |- SamplingCircuit       abstraction class for tactile circuit
|- TactileSensor/
| |- RowSensor_impl.h      implementation of tactile sensor template class
| |- RowSensor.h           template class definition for tactile sensor
                           with n taxels
| |- SingleSensor          class for single taxel (unused)
| |-                       abstraction class for tactile sensor
TactileSensorInterface.h
```

Note: if the internal wiring of taxels are changed within the finger, the order should be updated in the file *libraries/Fingers/OPT_demo/fingers.cpp*.

Appendix D

Tactile description on tested objects

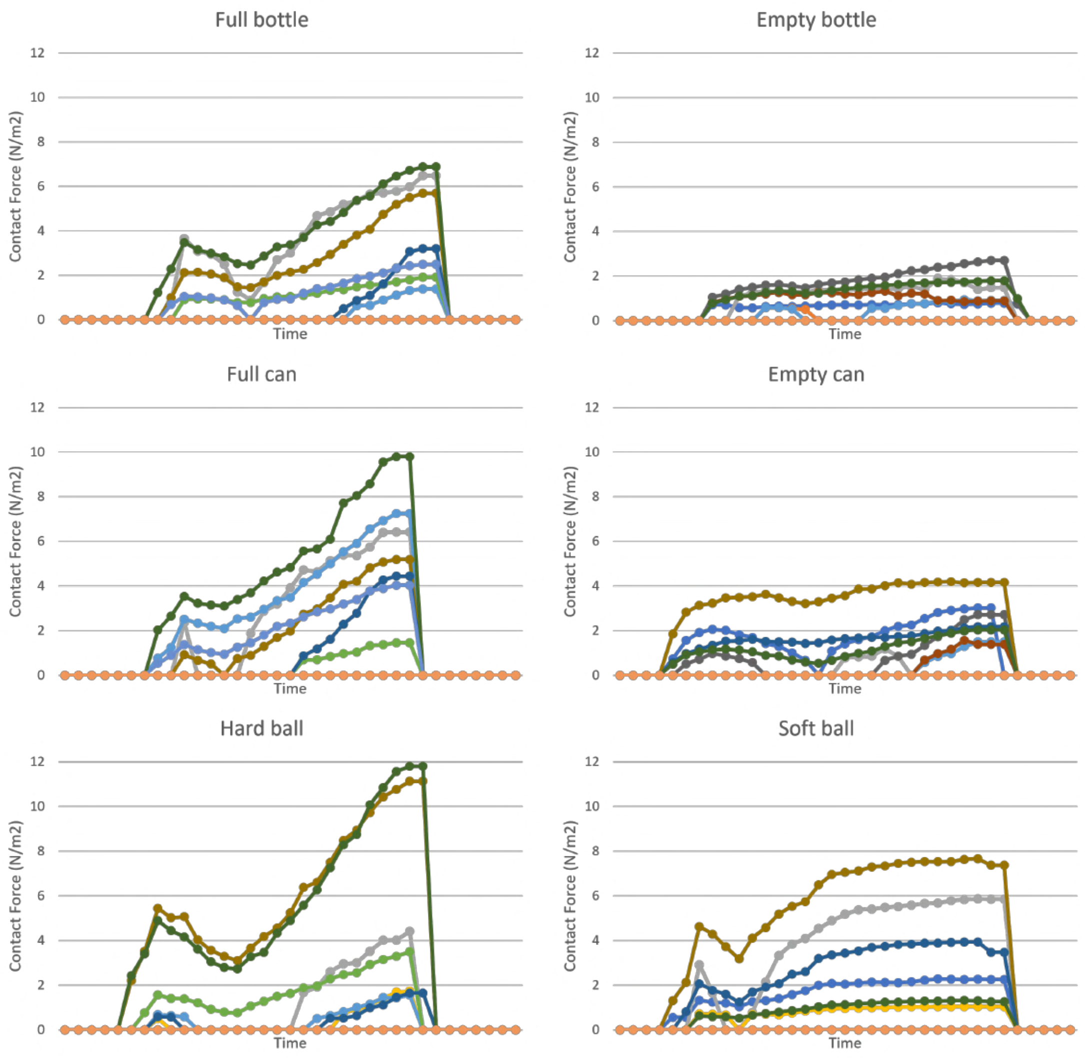


Figure D.1: Classification and connection between tactile sensing and perception.

Appendix E

Testing accuracy on the different objects

Faster R-CNN results

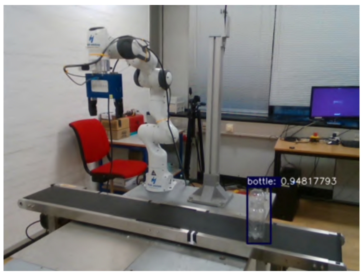


Figure E.1: 64_10_32 algorithm.

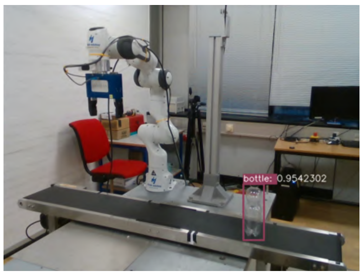


Figure E.2: 640_100_16 algorithm.

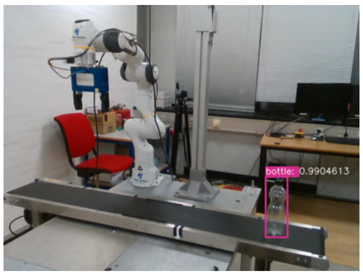


Figure E.3: 640_200_32 algorithm.

Figure E.4: Bottle classification by using three different models of Faster R-CNN.

Table E.1: Results of test on objects from the setup.

	64_100_32			**640_100_16**			**640_200_32**		
Objects	Acc.	MTP	mAP	Acc.	MTP	mAP	Acc.	MTP	mAP
Full bottle	100%	3.28	91.58%	100%	3.22	95.42%	100%	3.08	99.02%
Empty bottle	100%	3.21	93.64%	100%	3.14	98.76%	100%	3.09	98.98%
Full can	100%	3.30	89.30%	100%	3.23	99.13%	100%	3.26	97.74%
Empty can	90%	3.14	87.42%	100%	3.07	99.73%	100%	3.18	98.96%
Hard ball	100%	3.21	97.43%	100%	3.08	95.09%	100%	3.10	96.09%
Soft ball	100%	3.20	91.29%	100%	3.08	98.35%	100%	3.17	96.91%
Average	98%	3.22	91.78%	100%	3.14	97.75%	100%	3.15	97.95%

ML results

Table E.2: ML. Training performance.

Objects	Tree	KNN1	KNN2	KNN3	SVM1	SVM2	SVM3
Full bottle	95.0%	100.0%	91.7%	94.1%	100.0%	87.8%	88.2%
Empty bottle	100.0%	100.0%	96.4%	100.0%	100.0%	100.0%	100.0%
Full can	96.8%	97.1%	100.0%	100.0%	96.8%	97.0%	95.0%
Empty can	88.9%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Hard ball	91.7%	100.0%	100.0%	94.3%	100.0%	97.1%	100.0%
Soft ball	82.1%	87.9%	74.1%	92.3%	75.9%	79.3%	100.0%
Something else	89.0%	89.3%	84.6%	89.9%	71.8%	77.0%	45.6%

Vision-tactile-based algorithm results from test

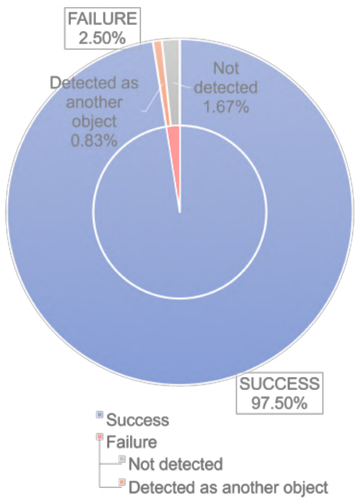


Figure E.5: Results on visual test with selected model (640_200_32).

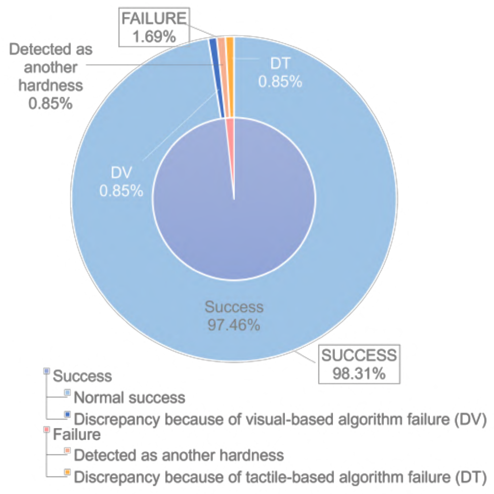


Figure E.6: Results on tactile test with selected algorithm (KNN1).

Figure E.7: Independent test results for the two algorithms.

Table E.3: Global algorithm test performance.

Object	Visual				Tactile					
						Visual failed		Visual succ-		
	Succ.	*VF*	*ND*	*Cause*	*Succ.*	*VD*	*VTF*	*TF*	*TD*	*Cause*
Full bottle	20	-	-		19	-	-	-	1	(TD) Tact. Som.Else
Empty bottle	20	-	-		19	-	-	1	-	(TF) Dcted. as FB
Full can	20	-	-		20	-	-	-	-	
Empty can	18	-	2	(ND) Not dcted. (ND) Not dcted.	18	-	-	-	-	
Hard ball	19	1	-	(VF) Dcted. can	19	1	-	-	-	(VD) Disc.
Soft ball	20	-	-		20	-	-	-	-	

Appendix F

Use case accuracy results on the different objects

Vision-tactile-based algorithm results from use case

Table F.1: Vision-tactile-based algorithm result on use-case.

Object	Visual component				Tactile component					
						Vis. failed		Vis. succ.		
	Succ.	*VF*	*ND*	*Cause*	*Succ.*	*VD*	*VTF*	*TF*	*TD*	*Cause*
Full bottle	20	-	-		18	-	-	1	1	(TF) Dcted. EB. (TD) Tact. Som. Else.
Empty bottle	20	-	-		19	-	-	-	1	(TD) Tact. Som. Else.
Full can	20	-	-		20	-	-	-	-	
Empty can	19	-	1	(ND) Pos. limit. area	18	-	-	-	-	
Hard ball	17	3	-	(VF) Dcted. tomato	17	3	-	-	-	(VD) Disc.
Soft ball	20	-	-		20	-	-	-	-	
Hard Tomato	15	5	-	(VF) Dcted. ball	15	3	2	-	-	(VD) Disc. (VTF) Misclass. / SB
Soft Tomato	15	4	1	(VF) Dcted. ball (ND) Alg. confus. ball	15	3	1			(VD) Disc. (VTF) Misclass. / SB

Appendix G

Others

Tactile test on other objects

Before selecting the objects, the gripper tested other objects. As can be seen in Fig. G.1 and Fig. G.2 some objects where discarded because with this gripper and the approach made with ML algorithms it was hard to get good results. The behaviour of the empty bottle is really similar to the half-full bottle. It happened the same with the two types of tennis balls, one completely new and one that had lost some pressure already.

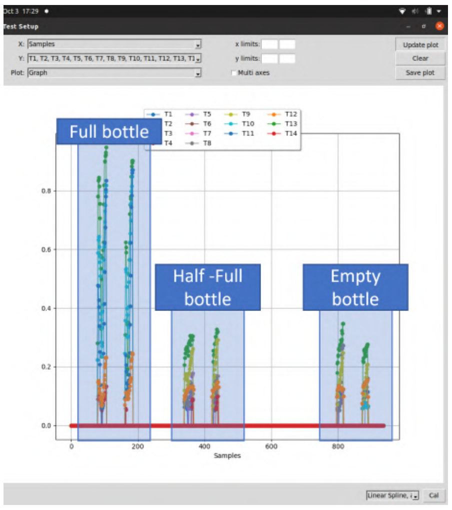


Figure G.1: Response of three different bottles for tactile test.

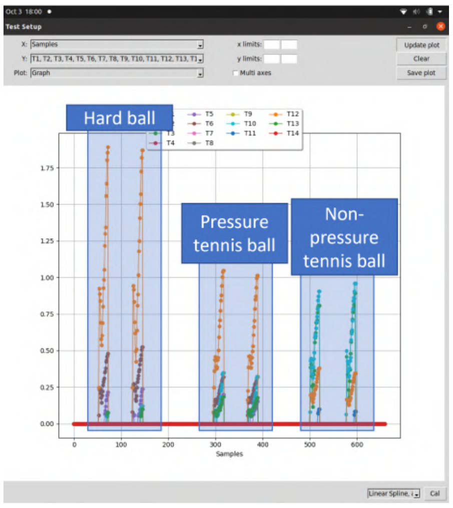
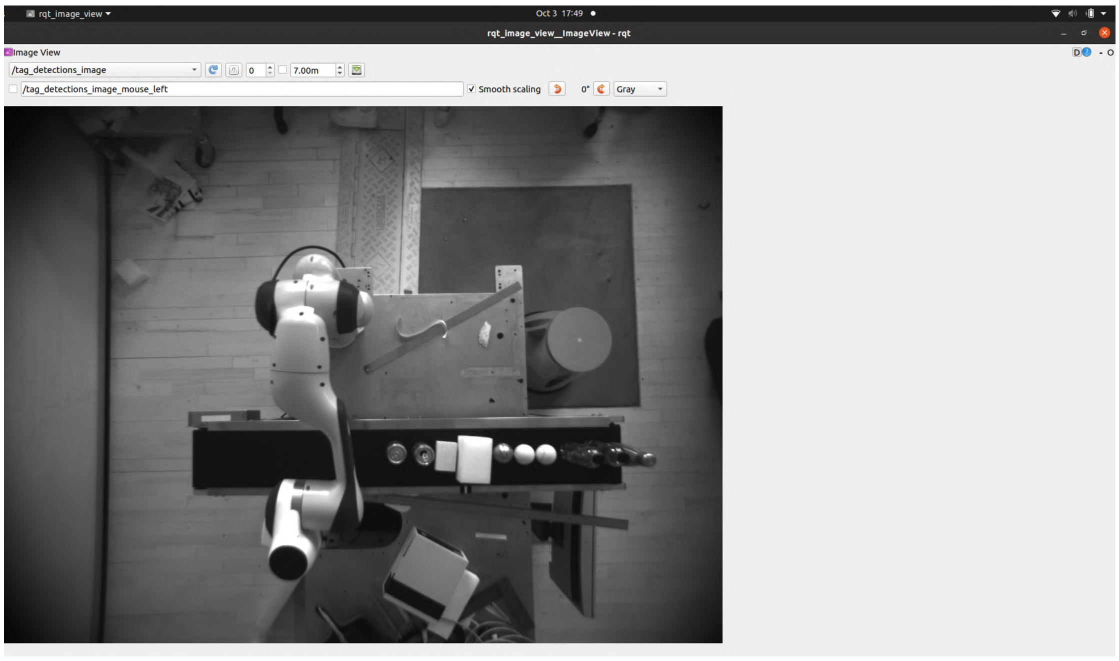


Figure G.2: Response of three different balls for tactile test.

BlueFox3 image results



/tag_detections_image

0

7.00m

/tag_detections_image_mouse_left

✓ Smooth scaling

0°

Gray

Figure G.3: Image obtained from BlueFox3 camera at the lab.