Universidade do Minho
Escola de Engenharia

José Miguel Maia Ribeiro

Development of an active vision system for robot inspection of complex objects

January 2022

Universidade do Minho
Escola de Engenharia

José Miguel Maia Ribeiro

Development of an active vision system for robot inspection of complex objects

Dissertação de Mestrado

Mestrado Integrado em Engenharia Mecânica

Sistemas Mecatrónicos

Trabalho efetuado sob a orientação do

Professor Doutor José Mendes Machado

e do

Professor Doutor Sérgio Paulo Carvalho Monteiro

January 2022

ACKNOWLEDGMENTS

This dissertation is the culmination of five years of study, which were only possible with the essential support and incentive of several people, to whom I cannot forget to thank.

First, I want to thank my two supervisors, Professor José Machado, and Professor Sérgio Monteiro, for the opportunity to work on this project, for the experience and guidance provided. Their support was crucial in both good and bad times for the success of this work. I would also like to take this opportunity to express my gratitude to Professor Estela and Professor Luis Louro, for their knowledge and experience which were fundamental in some critical moments of the project.

To all my friends, either at the university or in my hometown, a big thanks for all the support, adventures, and company that you gave me not only during this year but throughout this entire journey. It wouldn't have been the same without you.

One very special thanks to my dear friend Carlos Borges, who helped more than anyone during this year. Without his wisdom, calmness, and knowledge this work would not be complete.

I also want to take this opportunity to thank my girlfriend, Maria Sampaio, from the bottom of my heart for her constant support, encouragement and for always believing in me throughout the last five years.

To my parents, Fernando Ribeiro e Ana Paula, and my sister, Sofia Ribeiro, goes the most heartfelt thanks for their love, support and for always believing in me during these years. Without them, none of this would be possible. They were, are, and always will be my greatest example.

Lastly, I want to thank both Professor Sérgio Monteiro and Professor Estela Bicho for the opportunity to work on the project IntVIS4Insp, ref.: POCI-01-0247-FEDER-042778 with the grant support provided by "Fundação para a Ciência e a Tecnologia" with the reference UMINHO/BIM/2021/39.

STATEMENT OF INTEGRITY

Resumo

A dissertação aqui apresentada insere-se no âmbito do projeto IntVis4Insp entre a Universidade do Minho e a empresa Neadavance, e foca-se no desenvolvimento de um sistema para extração da posição e orientação das mãos no espaço para posterior auxílio na manipulação automática de peças de couro, com recurso a manipuladores robóticos.

O trabalho inicia-se com uma revisão literária sobre os dois principais métodos existentes para efetuar a recolha de dados necessária à monitorização da posição e orientação das mãos ao longo do tempo. Estes dividem-se em métodos baseados em luvas ou visão. No caso dos primeiros, estes recorrem normalmente a algum tipo de suporte montado na mão (ex.: luva em tecido), onde estão instalados todos os sensores necessários para a medição dos parâmetros desejados. Relativamente a sistemas de visão estes recorrem a uma câmara ou conjunto delas para capturar as mãos e por via de algoritmos de visão por computador determinam a sua posição e configuração. Foi selecionado para este trabalho um algoritmo de visão por computador denominado por Openpose. Este é capaz de, em cada imagem gravada e para cada mão, localizar 21 pontos pertencentes ao seu esqueleto.

Esta aplicação é inserida no sistema de monitorização desenvolvido, sendo utilizada a sua informação numa arquitetura mais completa onde é efetuada a extração da localização dos pontos chave de cada mão nos vídeos de demonstração dos movimentos de inspeção. A gravação destes vídeos é efetuada com uma câmara RGB-D, a Microsoft Kinect, que fornece um valor de profundidade para cada pixel RGB gravado. Com os dados de profundidade e a localização dos pontos chave nas imagens foi possível obter as coordenadas 3D no mundo destes pontos considerando o modelo *pinhole* para a câmara. No caso da posição da mão é selecionado um ponto de entre os 21 para a definir ao longo do tempo, no entanto, para o cálculo da orientação foi desenvolvido um método auxiliar para estimação da pose tridimensional da mão denominado por "Iterative Pose Estimation Method" (ITP). Este método recorre aos dados 2D do Openpose e às coordenadas 3D do pulso de cada mão para efetuar a correta estimação das coordenadas 3D dos restantes pontos da mão. Isto permite essencialmente resolver problemas com oclusões da mão, muito frequentes com o uso de uma só câmara na gravação dos vídeos. Uma vez estimada corretamente a posição 3D no mundo dos vários pontos da mão, a sua orientação pode ser definida com recurso a quaisquer três pontos que definam um plano.

Palavras-Chave: Inspeção, ITP, Monitorização da Orientação, Monitorização da Posição, Mãos

ABSTRACT

The dissertation presented here is in the scope of the IntVis4Insp project between University of Minho and the company Neadvance. It focuses on the development of a 3D hand tracking system that must be capable of extracting the hand position and orientation to prepare a manipulator for automatic inspection of leather pieces.

This work starts with a literature review about the two main methods for collecting the necessary data to perform 3D hand tracking. These divide into glove-based methods and vision-based methods. The first ones work with some kind of support mounted on the hand that holds all the necessary sensors to measure the desired parameters. While the second ones recur to one or more cameras to capture the hands and through computer vision algorithms track their position and configuration. The selected method for this work was the vision-based method Openpose. For each recorded image, this application can locate 21 hand keypoints on each hand that together form a skeleton of the hands.

This application is used in the tracking system developed throughout this dissertation. Its information is used in a more complete pipeline where the location of those hand keypoints is crucial to track the hands in videos of the demonstrated movements. These videos were recorded with an RGB-D camera, the Microsoft Kinect, which provides a depth value for every RGB pixel recorded. With the depth information and the 2D location of the hand keypoints in the images, it was possible to obtain the 3D world coordinates of these points considering the pinhole camera model.

To define the hand, position a point is selected among the 21 for each hand, but for the hand orientation, it was necessary to develop an auxiliary method called "Iterative Pose Estimation Method" (ITP), which estimates the complete 3D pose of the hands. This method recurs only to the 2D locations of every hand keypoint, and the complete 3D world coordinates of the wrists to estimate the right 3D world coordinates of all the remaining points on the hand. This solution solves the problems related to hand occlusions that a prone to happen due to the use of only one camera to record the inspection videos. Once the world location of all the points in the hands is accurately estimated, their orientation can be defined by selecting three points forming a plane.

KEYWORDS: 3D HAND TRACKING, INSPECTION, ITP, ORIENTATION, POSITION

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Nomenclature

1H One hand

2H Two hands

CCD Charge Coupled Device

CCG Centro de Computação Gráfica

CMC Carpometacarpal

CMOS Complementary Metal Oxide Semiconductor

CNN Convolutional Neural Network

CPM Convulotional Pose Machines

DIP Distal Interphalangeal

DoF Degree of freedom

FCRF Frame Conditioned Regression Forest

FIR Finite Impulse Response

fps frames per second

HCI Human-Computer Interaction

Hz Hertz

ICP Iterative closest Point

IIR Infinite Impulse Response

IK Inverse Kinematics

IMU Inertial Measurement Unit

IP Interphalangeal

IR Infrared

ITP Iterative Pose Estimation

LBS Linear Blend Skinning

LFR Latent Regression Forest

MCP Metacarpophalangeal

MSE Mean Squared Error

PIP Proximal Interphalangeal

PM Periodic Movement

PSO Particle Swarm Optimization

RDF	Randomized Decision Forest
RF	Random Forest
RGB	Red-Green-Blue
RGB-D	Red-Green-Blue-Depth
SMRF	Super-pixel Markov Random Field
SoG	Sum of Gaussian
STR	Semi-supervised Transductive Regression
SVM	Support Vector Machine
TMCP	Trapeziometacarpal

1. INTRODUCTION

1.1 Context

This work was carried out in the scope of the IntVIS4Insp project, between Minho university and the company NEADVANCE, whose goal is to conceive, develop and show new ways of realizing automatic inspection on complex objects with a high degree of flexibility and adaptability using robotic manipulation, computer vision and artificial intelligence. This dissertation focuses on specific type of complex objects: deformable ones.

One of the main goals of robotics is to achieve a more efficient and effective manipulation of objects, resorting on the speed, accuracy, and consistency that its solutions can offer in opposition to human manipulation. On the other hand, humans are more dexterous than any robot on what concerns object manipulation, which is particularly relevant when deformable objects are considered.

This work is focused on a particular deformable object: leather pieces, which present some resistance to compression and tolerate severe extensions. In terms of geometry, it can be considered a biparametric object, since one of its dimensions (thickness) is considerably smaller than the others. According to the classification of deformable objects proposed in section 3.1.1 these are of "type II", also called planar.

The visual inspection of deformable objects typically requires changes in perspective or illumination as well as torsions or extensions, all to highlight possible defects in the leather piece. If most rigid objects can be manipulated with just one arm, most complex objects, or materials (e.g., leather pieces, cables, cloth, ropes) require at least two supports (e.g., two arms) (Figure 1). Therefore, to execute all these movements two active or complementary supports are needed for the leather piece.

Figure 1-Example object inspection with bimanual manipulation [1]-

Consequently, two main problems arise when dealing with automatic inspection of complex/deformable objects: the problem of visual inspection related to the identification of defects; and the problem of manipulation that facilitates the visual analysis of the leather piece. The last being further divided into two different stages: pre-manipulation and during manipulation.

The work presented here tackles the problems of the pre-manipulation stage, where an initial manipulation strategy for the leather piece must be defined, and only then can the system evolve to the actual manipulation with the robot (during manipulation stage).

The mechanical behaviour of the leather piece, namely its numerous degrees of freedom and ability to tolerate severe deformations, makes it very challenging to use conventional motion planning to manipulate the object. One way of simplifying the problem is to rely on what human operators usually do when inspecting these objects and the executed movements can guide the robot on what it should do. With this strategy is also likely that the movements executed by the robot will be the best ones to highlight typical defects on this type of objects.

This kind of solution follows the ideas behind programming by demonstration (PBD) which is a well-known branch within robotics. The main idea behind PBD in robotics is to study and develop teaching/learning strategies for robots to execute a certain task. These can go from mimic a demonstrated action, to more abstract approaches that focus on understanding the underlying goals of the task [2].

With this in mind, the approach defined to perform automatic inspection on the leather pieces was to start by copying as closely as possible the trajectories executed by both hands of the operator during inspection of a leather piece. However, to make the robot copy these paths, first a system capable of obtaining them must be developed.

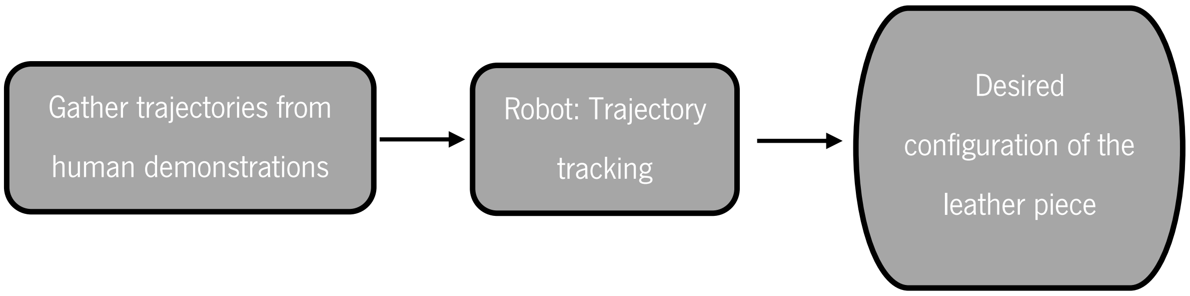

Gather trajectories from human demonstrations
Robot: Trajectory tracking
Desired configuration of the leather piece

Figure 20- Proposed simplified pipeline to manipulate the leather pieces during inspection.

In this work it is assumed that the manipulators that will execute these trajectories use a two-finger gripper (Figure 3), this means that only 7 degrees of freedom have to be defined for the robot to execute a task, in this case: position (e.g.: x, y and z values), orientation (e.g.: roll, pitch and yaw angles) and how close the fingers of the gripper have to be.

Figure 3-OnRobot RG2 gripper [3].

With that in mind this work intents to find a solution capable of obtaining the 3D trajectories of a hand, containing both position and orientation. Therefore, the problem here presented is one of hand pose tracking considering that both position and orientation are needed. These set of points (x, y, and z coordinates) and orientations establish an initial strategy to approach the manipulation of this objects in search for defects.

Choosing this approach pretends to simplify the interaction between a common user and the manipulator, by making it more intuitive to program the robot to execute complex movements like the ones presented in this task.

1.2 Motivation and Objectives

The main motivations for this work are intertwined with the ones that lead to the attempt of realizing automatic inspection of deformable objects with manipulators. The introduction of this type of solutions would allow for a more efficient and effective manipulation of objects, resorting on the speed, accuracy, and consistency that its solutions can offer in opposition to human manipulation.

If this work is successful, the developed solution could be used to extract the trajectories that will help in the complex motion planning task for deformable objects, which is needed to automate the inspection procedures. The main goal for this thesis is to develop a tracking system capable of extracting the hand trajectories that can be later executed by the manipulator while achieving the same type of configurations for the leather piece and highlight the existing defects.

With this main goal in mind, the dissertation must also achieve several stages such as:

- Study and development of an introductory literature review on existing technologies and methods to perform 3D hand tracking, to be used as foundation in further works on this project.
- Selection and implementation of a 3D hand tracking method/technology among the ones reviewed in the previous step.
- Development of a global tracking solution capable of outputting the desired trajectories for further implementation with a real manipulator.
- Results demonstration, comparison with ground truth values, description, and analysis from the implemented global solution.
- Manipulation of a leather piece with a real manipulator

The introduction to the 3D hand tracking problem is essential for a better understanding of the solutions one has at his disposal and which difficulties/problems are usually associated with each specific strategy available. The goal is not to develop a new method but rather select one that better fits the requirements of this project as described in the second topic. The chosen method must be implemented within a global architecture designed to adapt the base method to the requirements of our task: extract position and orientation of the hand. The developed solution doesn't need to provide state-of-the-art position and orientation results. However, it must be capable of accurately capturing the key tendencies of the trajectories executed by the hand so the manipulator can reproduce the same deformations on the leather piece. Characteristics such as being low-cost and practical implementation are also preferred.

The proposed solution will be tested with a set of basic inspection movements, that can later form more complex inspection strategies. The tracking results will be compared to the ones provided by a more powerful commercial solution to conclude about the feasibility and reliability of the solution.

It is also desired that at least some of the trajectories acquired by the solution are tested with a real manipulator.

1.3 Document structure

This dissertation is divided in six chapters, first the introduction intents to frame the reader for what comes in the remaining parts of this work, then it was necessary to introduce a literature review where different methods and solutions to help solve the problem of this work are studied. Next, some tools and theoretical principles used in the development of the hand tracking system are explained. The developed solution is described, and the corresponding results are presented in the last two chapters.

In **Chapter 1** it is presented the context for this work as well as the motivation and goals to develop it. It is also presented a detailed description of what are deformable objects as they are the base for this work.

In **Chapter 2** a literature review is made about the main existing methods to acquire data from the hands, to then extract their position and orientation. These are glove-based systems and vision-based systems. Some methods are presented for both approaches while pointing out the advantages and disadvantages of both methods, as well as their basic working principles. Lastly, the best method for this work is selected.

In **Chapter 3** are presented some basic instruments needed during the development of the hand tracking system. First, two main tools that integrate this final solution are explained starting with Openpose which is the basic tracking mechanics behind the final solution developed. Its working principles and ways of implementation are presented. Then a low-cost-depth camera is selected for this work and some key characteristics are shown. Second, two basic theoretical principles that also contributed to the design process are also explained.

In **Chapter 4** it is presented the 3D hand tracking system developed, going through the two different modules developed: the recording and processing modules. The first one, as the name suggests concerns the recording parts of our system. The second one englobes the Openpose application and all the complementary processing that occurs on the RGB-D information recorded by the first module. It is also described in this chapter the design of the "Iterative Pose Estimation Method".

In **Chapter 5** are shown the results from our system in six different videos of inspection movements compared to the ones provided by a more accurate and robust system (Vicon). It is also made an analysis on the results for both position and orientation.

In **Chapter 6** is presented an overview of all that was done in this dissertation and are discussed the main conclusions from it.

*2.*Literature review of *hand tracking methods and technologies*

This section describes the two main solutions for collecting the required data for hand tracking, the glove-based solution, and the camera- or vision-based solution, and looks at the advantages and disadvantages of each. It also discusses the difficulties related to the hand tracking task and presents different approaches/methods to get the hand pose from the data collected.

2.1 Hand tracking

The task of hand tracking is to estimate the human hand pose continuously in time whether is based on video or motion sensors [14]. These two solutions are often mentioned in the literature as vision-based or glove-based systems, respectively. The hand pose can be defined in very different ways depending on which information matters for the specific application, and therefore which hand model is considered relevant. The hand is a highly complex and articulated body part, which makes it difficult to model its kinematic and dynamic properties, especially in real time. First, it's necessary to understand its anatomical structure to arrive at a kinematic model, which is a basic representation of the hand skeleton. The kinematic model itself is the basis for the hand pose estimation problem.

The human hand consists of 27 bones belonging to one of the three parts: the wrist, the palm, and fingers, as shown in Figure 4. The bones in the skeleton form a rigid body system with joints having one or more degrees of freedom (DoF) for rotation. The joints between the bones are named as follows from the wrist to the fingertips:

- Carpometacarpal (**CMC**): joints connecting the metacarpal bones to the wrist.
- Metacarpophalangeal (**MCP**): joints between the fingers and the palm.
- Interphalangeal (**IP**): joints between finger segments. They can be further distinguished as distal interphalangeal (**DIP**) and proximal interphalangeal (**PIP**).

A kinematic hand model can be constructed according to the hand anatomy to encode the hand's kinematic properties. This model can be used to generate a feature vector to represent the hand's configuration. Considering an accurate anatomical structure of the hand will lead to a more complex kinematic model and a bigger parameter space. This high complexity might not always be necessary.

Therefore, low-resolution anatomical models can be considered, keeping the solutions only as complicated as needed.

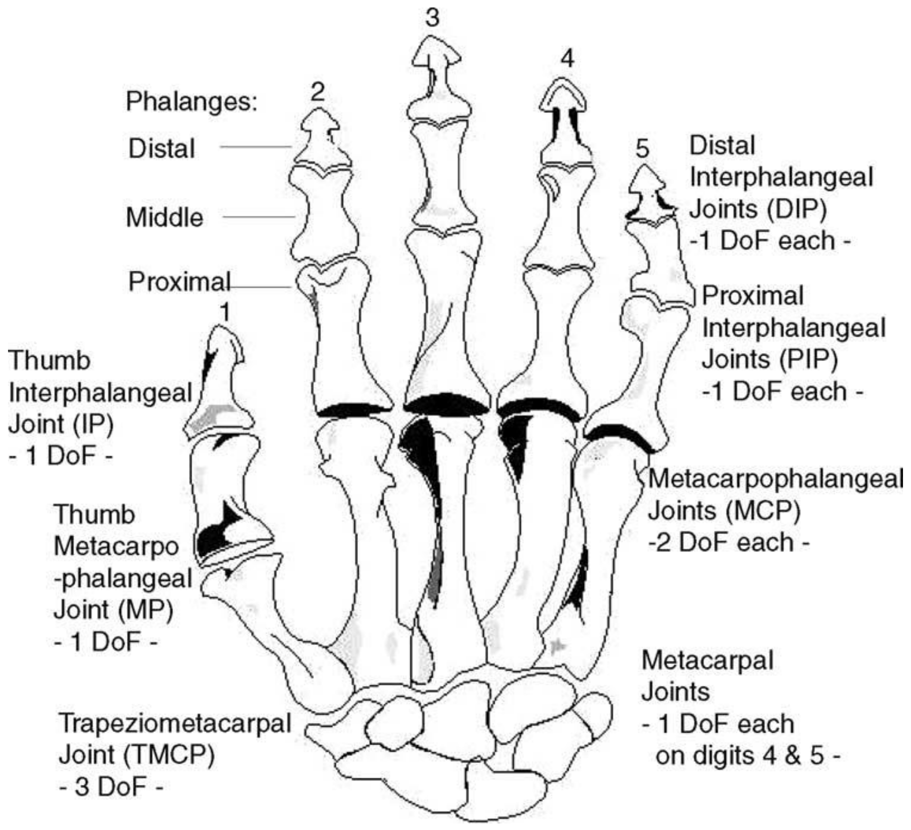

1
2
3
4
5
Phalanges:
Distal
Middle
Proximal
Thumb Interphalangeal Joint (IP) - 1 DoF -
Thumb Metacarpo -phalangeal Joint (MP) - 1 DoF -
Trapeziometacarpal Joint (TMCP) - 3 DoF -
Distal Interphalangeal Joints (DIP) -1 DoF each -
Proximal Interphalangeal Joints (PIP) -1 DoF each -
Metacarpophalangeal Joints (MCP) -2 DoF each -
Metacarpal Joints - 1 DoF each on digits 4 & 5 -

Figure 4 - Hand fingers [thumb (1), index (2), middle (3), ring (4), and little (5)], joints, and related 23 DoFs [15].

In Figure 4, for fingers 2, 3, 4, 5 the distal interphalangeal (DIP) and proximal interphalangeal (PIP) joints have 1 DoF each (flexion/extension) while the metacarpophalangeal (MCP) joints have 2 DoFs (flexion/extension and abduction/adduction). In the thumb, the interphalangeal (IP) and the metacarpophalangeal (MP) joints only have 1 DoF, but the third joint, trapeziometacarpal (TMCP) has 2 DoFs that allow flexion/extension and abduction/adduction and a third one that allows the thumb to rotate longitudinally as it is brought into opposition with the fingers. All these joints together provide 23 DoFs to the hand, which can be added to another 6 DoFs related to the overall hand motion in space (linear movements and rotations about the x, y and z axis). Other hand models can be considered, as the one presented in [14] with just 27 DoF (Figure 5).

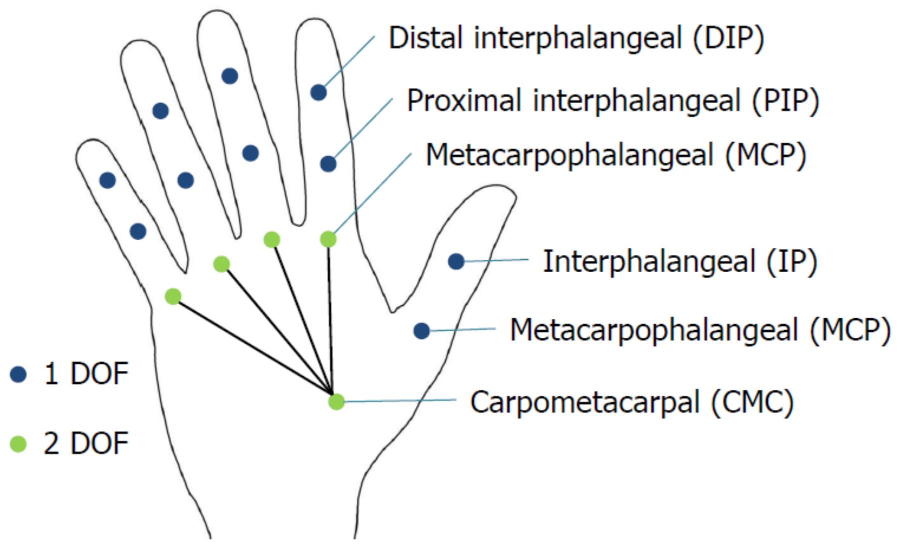

Distal interphalangeal (DIP)
Proximal interphalangeal (PIP)
Metacarpophalangeal (MCP)
Interphalangeal (IP)
Metacarpophalangeal (MCP)
Carpometacarpal (CMC)
● 1 DOF
● 2 DOF

Figure 5 - 21 DoF model of the human hand with the additional 6 DoF for translation and rotation of the entire hand. This yields a model with a total of 27 DoF [14].

After defining the hand model of interest and before using hand postures or hand trajectories one needs to gather raw data. Only after that can the data be analysed to extract meaning or context from it which can be used to carry out a specific task. As mentioned before, raw data is mainly collected in two ways. The first consists in using input devices worn by the user (glove-based), usually supported in one or two instrumented gloves. A six degree of freedom (6 DoF) tracking device placed in the glove gathers hand position and orientation data, while other sensors measure all the necessary joint angles of the hand. The second way to collect raw hand data is to use a computer-vision-based approach by which one or more cameras collect images of the user's hands. The images acquired are sent to image processing routines to perform posture recognition as well as 3D triangulation to find the hands' position in space.

There's also a third way to collect raw hand data, which combines the previous two methods in a hybrid approach with the hope of achieving more accurate results by using the two data streams to reduce each other's error.

The hand tracking problem has been being studied for many years now, and has seen most of its developments under the HCI (Human Computer Interaction) ([14],[16]) area where gesture recognition could revolutionize the way people interact with computers. This extents to many specific applications, which can range from more professional ones like motion capture, navigation in virtual environments, programming robots by demonstration; to applications more related to people's day to day life, like gesture driven game-control, interaction with smart TVs and mobile applications. Some well-known brands already use this type of approach to facilitate the interaction between the user and their products, it is the case of Nintendo, Sony or Microsoft which already implemented body or hand tracking systems in their consoles Nintendo Wii, Sony move and Microsoft Kinect respectively. In Figure 6 it is possible to see areas that can benefit from this type of systems.

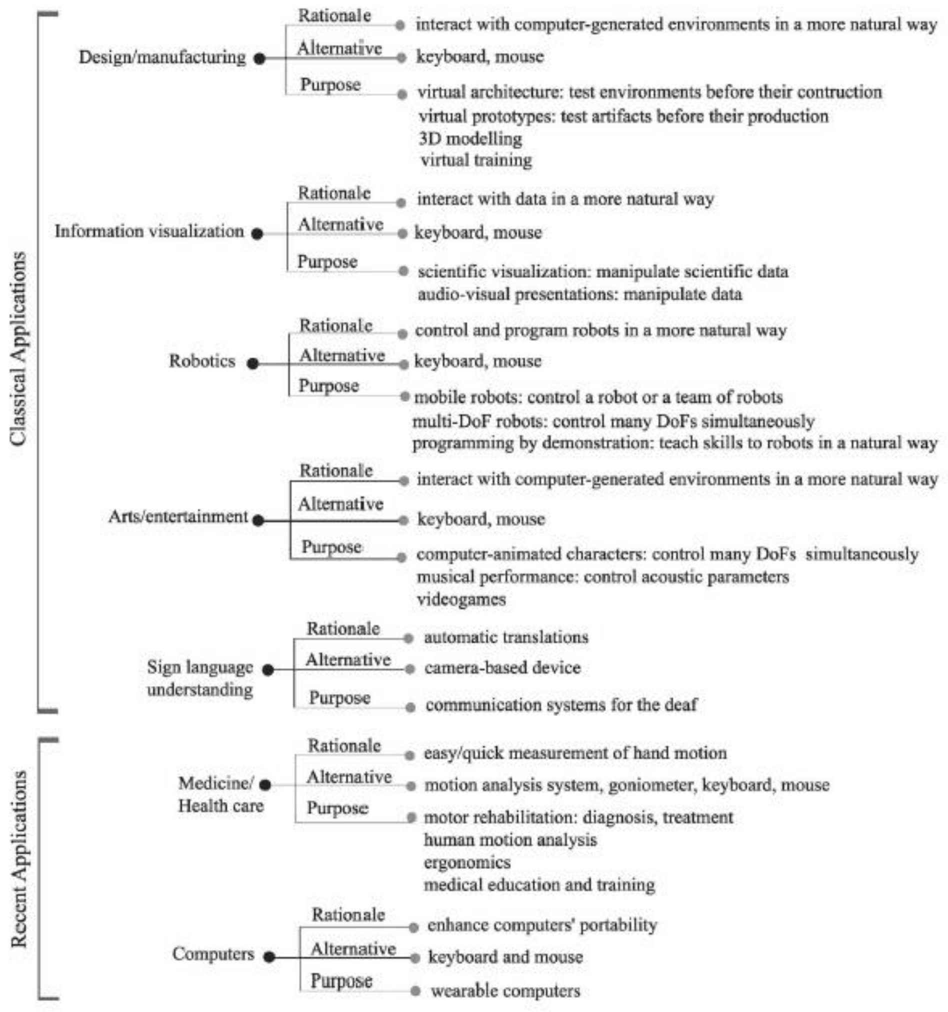

Classical Applications
Design/manufacturing
Rationale
interact with computer-generated environments in a more natural way
Alternative
keyboard, mouse
Purpose
virtual architecture: test environments before their contruction
virtual prototypes: test artifacts before their production
3D modelling
virtual training
Information visualization
Rationale
interact with data in a more natural way
Alternative
keyboard, mouse
Purpose
scientific visualization: manipulate scientific data
audio-visual presentations: manipulate data
Robotics
Rationale
control and program robots in a more natural way
Alternative
keyboard, mouse
Purpose
mobile robots: control a robot or a team of robots
multi-DoF robots: control many DoFs simultaneously
programming by demonstration: teach skills to robots in a natural way
Arts/entertainment
Rationale
interact with computer-generated environments in a more natural way
Alternative
keyboard, mouse
Purpose
computer-animated characters: control many DoFs simultaneously
musical performance: control acoustic parameters
videogames
Sign language understanding
Rationale
automatic translations
Alternative
camera-based device
Purpose
communication systems for the deaf
Recent Applications
Medicine/ Health care
Rationale
easy/quick measurement of hand motion
Alternative
motion analysis system, goniometer, keyboard, mouse
Purpose
motor rehabilitation: diagnosis, treatment
human motion analysis
ergonomics
medical education and training
Computers
Rationale
enhance computers' portability
Alternative
keyboard and mouse
Purpose
wearable computers

Figure 6- Areas of interest for hand tracking systems and its benefits[15].

2.2 Glove-based systems

Dipietro, et al. [15] defines a glove-based system as a set of sensors, electronic devices for data acquisition/processing, as well as a power supply and a support for all the necessary sensors that can be worn on the user's hand. Typically, the sensors are attached to a cloth glove, but it can also be some type of structure that maintains the sensors linked to the user's hand. This way it records data related to his/her hand configuration/motion.

A glove equipped with one sensor per DoF considered on the hand might seem the most obvious design choice. However, several gloves with different designs have been developed over the years to fit specific applications with different design requirements and goals.

When designing or selecting these glove-like devices, one of the things to consider is the type of information and respective detail needed. This decision will influence the type and number of components required for the glove to give the necessary information. Both in the market and literature, it is possible to find a variety of examples where different needs lead to diverse design choices, especially regarding the sensor technology used. When dealing with 3D tracking of the hand pose one needs a complete description of hand movement, which requires knowledge of both hand configuration (amount of joint bending or joint relative positions) and hand position in space (location and orientation of the hand, for a total of 6 DoFs—translation and rotation along the x, y, and z axis). While glove technologies record the first type of data, trackers record the latter. Usually, gloves and trackers are used in conjunction and commercial systems typically contain both. Two main types of information can be considered, the **hand's overall position and orientation** or its **joint angles**. Both might require diverse sensing technologies and post-processing algorithms.

2.2.1 Tracking hand's position and orientation

Among all the different tracking technologies available to track the hand's position and orientation (6 DoF), four sensing ones are magnetic, acoustic, inertial, and optical.

Magnetic: With magnetic tracking, a stationary transmitter emits a low-frequency magnetic field from which the moving sensor, the receiver, determines its position and orientation relative to the magnetic tracker. According to [17] this type of sensors have good range and are usually accurate. They don't require direct line of sight transmitter receiver either. However, any ferromagnetic or conductive objects present in the room with the transmitter will distort the magnetic field reducing the accuracy. In [18] an hand tracking prototype named "Finexus" uses magnetic sensors to determine the position of the fingertips in relation to a electromagnet. The prototype has a reported mean error of 1.33 mm if within a 120 mm range from the transmitter.

Acoustic: Acoustic tracking systems or ultrasonic tracking uses high-frequency sound produced by a source component that is captured by microphones to determine the position and orientation relative to each other [15]. One component can be placed on the hand and the other in the area to be tracked or vice versa. According to [17], these sensors are relatively inexpensive and lightweight. However, these devices have a short range, and their accuracy is affected if hard surfaces are present in the room and if line of sight between transmitter to receiver is compromised. Another disadvantage is that external noises can interfere with the tracking signal and thus affect accuracy. Update rate is approximately less than half that of magnetic trackers [15]. This technology is used in the Mattel Power Glove, developed by Nintendo, to track the hand's x, y, and z position and roll orientation of the wrist. The Power Glove was not very accurate and was useful only for a small set of simple hand postures and gestures [17].

Inertial: Inertial tracking systems use a variety of inertial measurement devices such as gyroscopes that measure the rate of change of an object's orientation, or a servo accelerometer that measure the rate of change of an object's translation velocity. The advantages of an inertial tracking system are speed, sourceless operation leading to no line-of-sight constraints, and very low sensor noise. The major problem with these systems is they suffer from gyroscopic drift, which means that to obtain position and orientation first the data collected from sensors is integrated from a known starting configuration, therefore errors are accumulated, and even small variations can lead to large errors over time [15].
IMUs are one of the most used devices for hand motion tracking, they can measure acceleration, rotational speed, and orientation due to its multiple sensors: a 3-axis accelerometer, a 3-axis gyroscope, and a 3-axis magnetometer. One major drawback of this solution, especially in the smart gloves' context is their rigidity and bulkiness compared to the size of human fingers. In any case, IMUs are used in numerous commercial gloves such as Cobra Glove, Exo Glove or Hi5, and can be attached to the gloves over the wrist to estimate the hand position and orientation in the space, as it is shown at column "External IMU" in Table 2.

Optical: Optical tracking uses small markers placed on the subject's body, that can either be flashing infrared LEDs or infrared-reflecting dots. A set of cameras around the subject capture the markers, then the information is used to calculate a 3D coordinate for each marker [19]. One limitation of **Marker systems** can be the processing time needed to analyse the several camera images and determine each marker's 3D position. Most of the systems operate in a batch mode, where the trajectories of the markers are captured live, followed by a period of analysis to calculate 3D coordinates. In LED systems, the LEDs are sequenced so that only one lights up at a time avoiding misperception between the markers. In the case of reflective markers, they also require a middle stage of analysis to identify markers and resolve ambiguities when they coincide in the visual field (the more cameras, the smaller this problem). These systems are widely use whenever motion capture is needed, however, their real-time limitations and an inability to resolve markers that are too close together restricts their use for tracking fingers in interactive applications. The sensitivity to reflective surfaces is another disadvantage of this methods. Like acoustic trackers, optical trackers require direct line of sight and are insensitive to metallic interference. The high accuracy of this methods and their large tracking area are other advantages to consider. Examples: Vicon systems.

2.2.2 Tracking hand configuration

There are multiple types of sensors capable of measuring the hand's configuration (joint angles), some of them can also be used for tracking the hand's position and orientation. Next, the most relevant ones are presented, namely: Flex sensors, accelerometers, stretch sensors, magnetic sensors, and rotational encoders.

Flex sensors: These are piezo resistive elements that change their resistance as they are bent or flexed, creating variations in the transmitted electrical signal. Such variations can be measured and mapped to changes in joint angles [20]. In the gloves, these sensors should be placed in the location of the joints of interest. In such a position, a one-to-one mapping between the joint blend and the sensor reading could provide an accurate measurement. It is particularly difficult to accurately measure joints involved in abduction and adduction movements. Another problem for bend sensors is their short lifespan, as the continuous bending makes them to break quite fast. Flex sensor-based technology is the most widely used method in designing wearables associated with hands [21]. Cyberglove III is a flex sensor-based glove used for gaming purposes and PC control. It has 18–22 sensors embedded on it with a reasonable accuracy of less than 1°. Connolly et al. [22] also proposed two flex sensor-based gloves for hand joints measurement, namely 5DT Data Glove and X-IST Data Glove.

Accelerometers (IMUS): Accelerometers can be used to measure position and orientation of an object. If placed in the right places of the hand these sensors can help understand its configuration. OFlynn et al. [23] developed an Inertial Measurement Unit (IMU) smart glove microsystem based on sensors, processors and wireless technology used for human computer interaction. It consists of 16 9-axes IMU's, where each one includes a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer and provides a real-time measurement of a range of hand joint movements including the measurement of flexion/extension, adduction-abduction, and complex hand movements. In this case, the relationship between sensor location and DoF involved is not straightforward. This kind of sensor measures the global movement. Therefore, a reference to a fixed point needs to be established to estimate the actual DoF performed. In case of bend or stretch sensors, the relationship between the movement and the DoF is clearer. Examples of commercial gloves using this technology are Senso Glove DK3 [24] and Cobra glove [25].

Stretch sensor: Stretch sensors are used to measure stretch, bend, pressure, and force and are widely used for tracking hand movements in applications ranging from soft robots, Virtual Reality (VR) gloves, biometric displacement reading and other physical applications [21]. These sensors are typically resistors with resistance values depending on the sensor's deformation, its stretching increases its resistance, whereas its squeezing decreases its equivalent resistance. These sensors are available in different sizes, sensitivities and elasticities based on their intended applications. The StretchSense MoCap Pro it is an example of the use of this technology to track the hand's configuration.

Magnetic sensor: The magnetic sensors working principle is based on magnetic fields to detect the position of an object. Hall-effect sensors are used for the accurate measurement of the flexion/extension and adduction-abduction motion of the proximal joint of the fingers. Figure 7 depicts the Humanglove having 20 hall-effect sensors, which is mainly used to measure the flexion/extension of the fingers and thumbs as well as their adduction-abduction motion. Based on the type of the output signal, these sensors are characterized namely as analog and digital. In the analogue sensor, the output signal is of continuous nature and is directly proportional to the strength of the applied magnetic field.

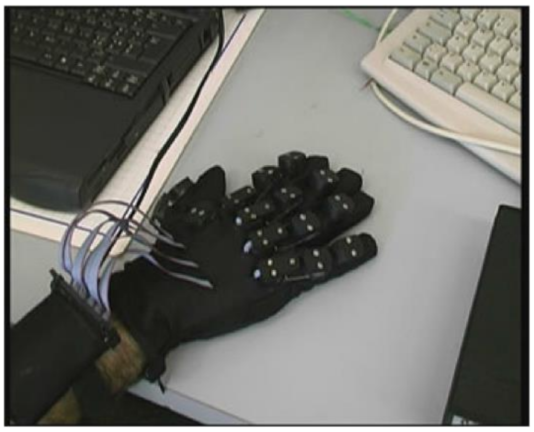


Figure 7-Humanglove [15].

Rotational sensors: There exist some gloves that include mechanical rotational sensors, such the Dexmo exoskeleton, or rotational encoders, such as the SenseGlove DK1. These sensors can convert the angular position of a shaft to a signal. They are quite bulk, but they can be embedded in these gloves because their exoskeleton structure facilitates it.

The comparison between all these alternatives is summarized in Table 1 where five important aspects are evaluated: accuracy, response time, cost, lifetime, and ease of wearing.

Table 1-Comparison of all the technologies listed to track the hand's configuration. Adapted from [16] with some changes.

Technology	**Accuracy**	**Response time**	**Cost**	**Lifetime**	**Ease of Wearing**
Flex sensor	high	medium	medium	medium	medium
Accelerometer	medium	fast	low	long	hard
Stretch sensor	medium	slow	low	short	easy
Magnetic sensor	low	fast	medium	long	hard
Rotational encoder	medium	fast	high	long	hard

However, as shown in Table 1, we can still benefit from the analysis of different types of sensors; on one hand, bend (flex) sensors and stretch (strain) sensors are very suitable for hand pose estimation as they are less disturbing for the users with deformable abilities to follow finger movements and palm deformations; on the other hand, IMUs and magnetic sensors have no burdens from mechanical deformation; thus, can have longer lifespans across usage. Thus, the optimal design of a data glove may involve multiple types of sensors to joint their advantages for better performance with lower cost. A further comparison of wearable technologies on accuracy, cost, and lifetime can be found in [11].

2.2.3 Instrumented gloves

In order to better describe the evolution of glove based systems, [15] divides the process in three different stages: early research, Data glove-like systems and beyond Data gloves. This classification takes the reader through the first advances in glove technologies, but also highlights the importance that the Data Glove had on the way to get to the models in the market today.

The turning point for the development and research in instrumented gloves for hand tracking came when the Data Glove was commercialized in the United States in 1987. The initial version developed by Zimmerman in 1982 used thin flexible plastic tubes sewn on a cloth and light sources and detectors to record joint angles. A new fibre optics version was developed and commercialized in 1987 by Visual Programming Language Research, Inc.

The Data Glove-like systems also include the commercial Space Glove, CyberGlove, Humanglove, 5DT Data Glove (Figure 8).

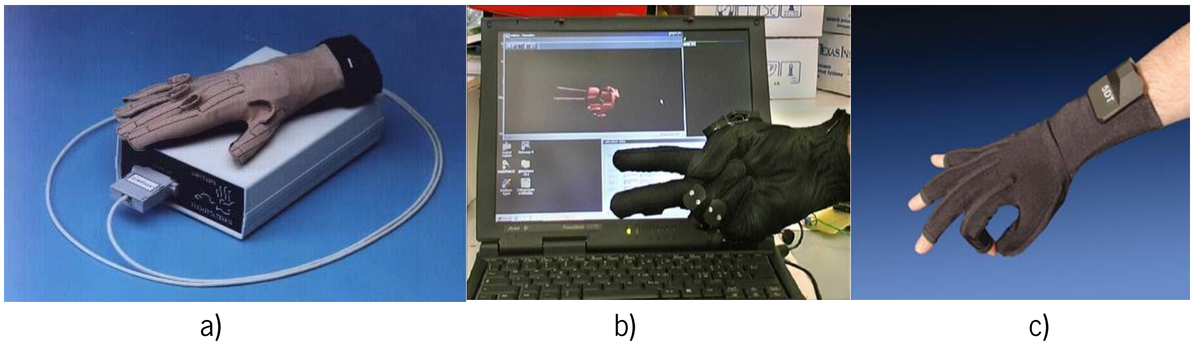


Figure 8–a) Virtual Technologies' CyberGlove from [17];c) Humanglove; b) 5DT Data Glove. Both images from [15]

Although these gloves differed from the original Data Glove in terms of sensor technologies, locations, and mounting, they all shared three basic design concepts with it: they measured finger joint bending, used a cloth for supporting sensors, and were usually meant to be general-purpose devices. Despite their widespread use, they suffered from several drawbacks. Its main limitation come from the cloth support, which limits the user's haptic sense and naturalness of movement, making the glove cumbersome. The need for a tedious user-specific calibration procedure is also a disadvantage as well as their poor robustness, poor durability, and high cost.

Nowadays, many Data-glove like models are no longer available in the market, this is the case for example for the CyberGlove and HumanGlove. With the increasing demands of recent applications, there was also the need to improve the typical design of the models inspired in the Data glove and new models started to appear in the market. In [20], a review is made on twenty four commercial "smart-gloves" currently available in the market (Figure 9) and Table 2 summarizes the most relevant information about eight of them just to provide an overview of the devices available.

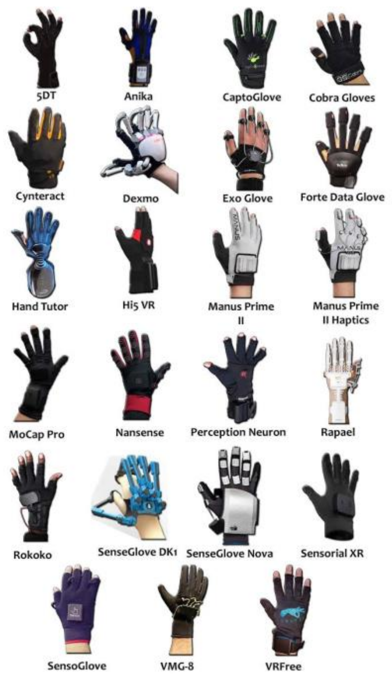


Figure 9-Images of 23 of the 24 commercial gloves reviewed in [20]. Image adapted from the same source.

The information provided for all the commercial gloves presented, proofs that in fact these are extremely complete pieces of equipment when one looks for an accurate solution since most models present an accuracy for joint angles below 1°, and the displacement resolution can go up to 0.3 mm, which is the case for the VRFRee glove. These are also capable of working in real-time (update rates up to 500 Hz) and now can even be considered as portable devices with the introduction of wireless methods of communication and batteries.

Table 2-Summary of some important information about 8 commercial gloves currently available in the market [20].

Smart Glove	**Glove type**	**Sensor type**	**External IMU**	**Battery and Autonomy**	**Connection**	**Size**	**DoF** [1]	**Price**[2]
Anika Rehap	Strips	NA	NA	Wired provided	USB	Adaptable	NA	$1300
Dexmo	Exoskeleton	Rotational	Yes	Li-Ion Polymer 5 h	Wi-Fi 2.4 USB	Unique	10	$36,000
Forte Data Glove	Strips	Bend + IMU	Yes	Li Polymer 6–8 h	BLE USB	Unique	13	$3000
Mocap pro Supersplay	Fabric	Stretch	Yes	Battery 8 h	Bluetooth, Wi-Fi, USB-C, SD card	2 (S/M, M/L)	11	$7150
Nansense R2	Fabric	IMU	Yes	Battery 6-8 h	Wi-Fi 2.4, 5 USB-A	3 (S, M, L)	7/12/15	$4798
SenseGlove DK1	Exoskeleton	IMU + rotational	Yes	Li-Ion battery 2 h	USB Bluetooth	Unique	23	€2999
Senso Glove DK3	Open tips	IMU	No	Li-Ion Polymer 1.5 h	USB RF, BLE	5 (S, M, ML, L, XL)	9	$999
VRFree[3]	Open tips	6 types	No	Replaceable rechargeable battery	Wireless USB-C	4 (S, M, L, XL)	23	CHF750

[1] - Some values provided for the number of degrees of freedom measured by each glove are estimates made by the authors considering the information given by the manufacturers. More detailed information can be found in

[2] – The prices for some gloves are estimates based on the information given by the companies, however these could vary depending on extra accessories that could be included.

[3] – The sensors used in the VRFree glove are of 6 different types according to the company, but they are not specified.

Even though this technology has evolved considerably, there's still the need to wear a structure on the hand to hold all the necessary components, which makes the experience more unnatural, especially when one intends to use these devices while manipulating real objects. Vision based methods on the contrary can track the hands freely allowing for a more natural interaction between the user and the objects being manipulated. Of course, these solutions present other problems that will be discussed later in section 2.4.The price is another problem related with glove-based solutions, in Table 2 the prices of eight examples of commercial gloves are presented and only one is below 1000$, this tendency is also shown in the remainder glove models presented in [20] where the average cost for these products is still really high when compared to vision based solutions.

2.3 Vision based systems

Vision based methods focus mainly on solving the problem of hand pose estimation, however, finding the hand pose usually requires finding the hand's position within the frame in previous steps. Therefore, in this work, there's not only interest in pose but also in position information and both are usually provided by this type of solutions, even if position it's not the main goal of some of them.
Vision based approaches can be divided into 2D and 3D, according to the type of input data used. In the case of 2D methods they only use RGB data to estimate the hand's position and pose, however these procedures must deal with the difficulties related to depth ambiguity of just 2D information. Finding datasets to train the algorithms is another issue that designers face when developing this type of solution. To solve the depth ambiguity problem, 3D methods appeared by taking advantage of the emergence of 3D sensors in the market, more specifically RGB-D cameras. These devices usually provide an RGB image and the corresponding depth map which makes it easier to understand the 3D hand configuration.
However, even though it's easier now to access 3D sensors, they are still more expensive than regular monocular RGB cameras, and have limited range and resolution, which still makes 2D methods attractive. Both for 2D and 3D approaches, there are many difficulties that designers must consider when deriving a solution. The number of difficulties to face are proportional to the robustness desired for the solution. If one wants to develop a system capable of estimating the hand pose continuously without any restrictions with just one camera it is likely that he/she will have to solve the following problems [14], [26]:

Occlusions: If only one camera is considered there is only one image plane which might lead to serious hand occlusions, caused either by the hand itself or other objects/body parts. If certain parts of the hand are hidden from the camera it's difficult to formulate a hypothesis for the whole hand configuration. Even though systems with multiple cameras also have to deal with this problem, more cameras mean more area covered being less likely that there are parts of the hand hidden.

Skin colour similarities: One of the main tasks when performing hand tracking, whether to determine the hand's position or pose, it is to find the hand within the image. One of the most used methods to find the hand is skin colour segmentation, which finds all the blobs in the image that have the colour of human skin, this technique is used for example in [27] and [28]. One of the problems of this technique has to do with chromatically similarities between the hand and other body parts, which leads to the implementation of complementary algorithms to sort things out.

High dimensionality: This problem is especially related to the hand pose estimation problem and the definition of a previous hand-model that will serve as reference for the algorithms to estimate the hand configuration. The problem stands with the fact that these models can require a large set of parameters to define the hand-model for each image which leads to a complex and time consuming process of optimizing or finding this set of parameters in real time as for the case in [28].

Chromatically uniform appearance of the hand: Chromatically uniform appearance means that the hand does not have many differences in terms of colour throughout its surface, which complicates the work of low-resolution cameras that cannot capture those subtle differences accurately. If, for example, someone is making closed fist gesture with the hand, it would be difficult for the camera to get the small details that allow to understand the overall configuration.

Complex background: Complex backgrounds is one of the most common problems that designers face when trying to develop a robust solution. This problem usually stands out more for applications where performance in the wild is required and the use of depth sensors to help define what is background is limited as in [27]. Skin colour segmentation is also tricky to implement in these situations due to all the colour noise usually present in the background

Illumination variance: When recording a video or taking a picture, lighting is one of the most important aspects to consider, to get the results closer to reality. The amount of light directed to a surface influences how accurate the colour will look to the human eye. Too much light will make the colours look brighter, until it gets white if it reaches a point of saturation. The case is the same if there's too little light on a surface, the colours will look darker until ultimately reach black. Therefore, controlling the light is one of the most important things to do if one wants to guarantee that the hands are always clearly seen in all images.

Complexity of movements (rotation/translation): To derive a solution to find the hand and its configuration one must consider the different movements that it can perform, specifically rotations and translations that might change the way the hand is presented in each frame making it challenging to detect it in an input image because neither its appearance nor position are known in advance.

Scale/size changes: When tracking the hand's position and pose during complex movements of the hand it is likely that the hand will change its size in relation to the camera as it moves away from it or in its direction. Being able to cope with these changes and still identify the hand is a hard task as referred in [29].

Besides the type of input data used, vision-based hand pose estimation methods can generally also be grouped into two categories, namely generative and discriminative as stated in [14], [16]. The purpose of the hand pose estimation based on these methods is to obtain a representation of the hand for tracking hand movement.
Generative methods are also known as model-based or model-driven methods, as they rely on the construction of a 3D hand model based on prior knowledge of the hand structure which is iteratively optimized to better fit the shape of the hand seen in the images. Discriminative methods are also called appearance-based methods or data-driven methods, and they directly predict the joint locations from images to implement hand pose estimation.

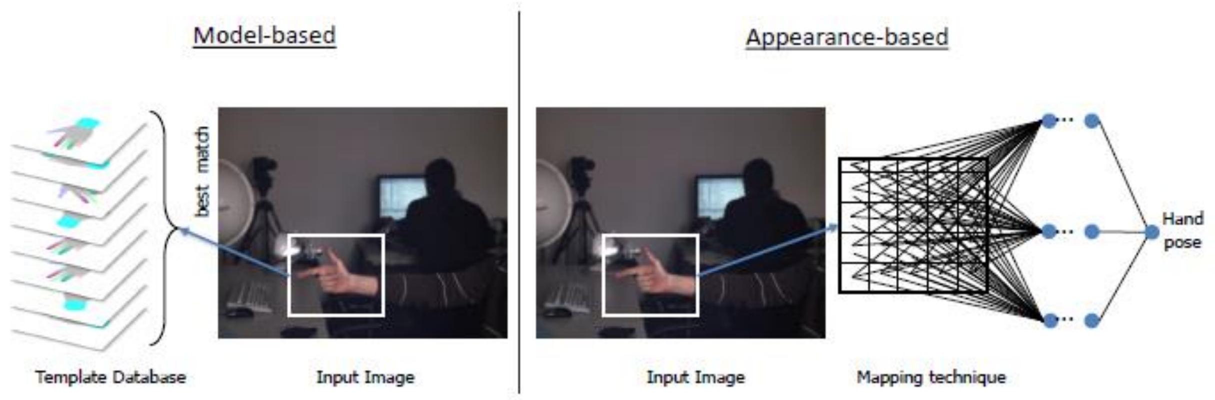


Figure 10- Model-based approaches where an object model (human hand) is used to formulate templates, each representing a hand pose and match them to the input image. In contrast, appearance-based approaches try to learn a direct mapping from the image space to the target parameter space (pose)[14].

2.3.1 Discriminative methods

Data-driven methods, as the name suggests, rely their predictions only on the data that is fed to the algorithms in the learning stage, usually with no previous assumptions. The results are as good as the amount of labelled hand data used to train the model that then can predict the coordinates of the hand joint points to achieve hand pose estimation. These methods learn how to map from visual features (e.g.: edges, shape) to the target parameter space, that can be joint labels or joint 3D locations from images.

Compared to model-based solutions, such approaches do not need to search for all possible configurations at every time step because the information about the hand poses is already encoded in the learned mapping making these solutions computationally more efficient. On the other hand, they lack accuracy and stability due to difficulties in handling noise and occlusions in the input image [30].

There are two major types of discriminative methods: random forests (RF)- and convolutional neural network (CNN)-based models [16]. The theoretical principles of these two methods can be found in the Appendix I – Theory behind Discriminative methods for 3D Hand tracking

In the literature is possible to find multiple solutions using the random forest algorithm. Keskin et al. [31], was a pioneer in this line of work using a randomized decision forest (RDF) to assign the input depth pixels to hand shape classes for hand shape classification and directs them to a multi-layer RDF framework for hand pose estimation trained specifically for that hand shape. However, the above approach uses mainly synthetic data in training since it needs large amounts of per-pixel labelled training data, which is difficult to obtain. This leads to performance discrepancies among realistic and synthetic pose data. To tackle this problem, Tang et al. [32] proposed a semi-supervised transductive regression (STR) forest to learn from both labelled and unlabelled data, these concepts are represented in Figure 11. They also designed a data-driven pseudo-kinematic technique to refine noisy or occluded joints.

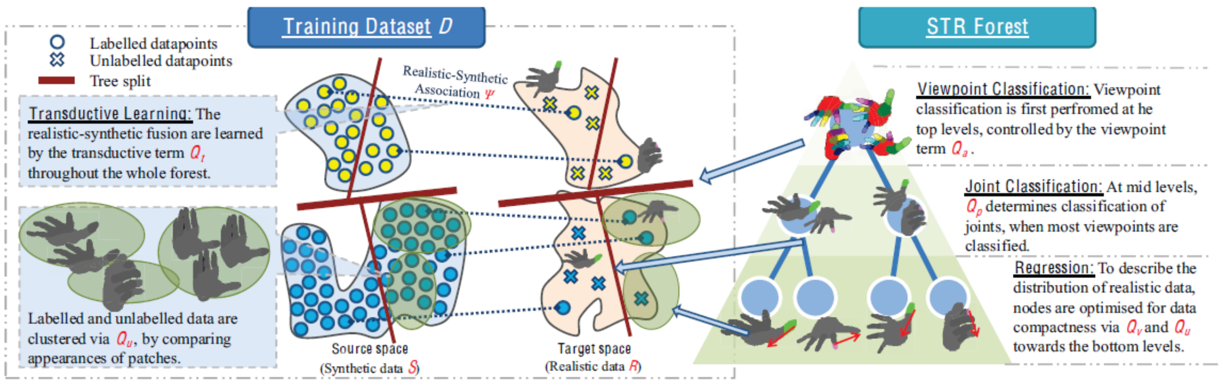

Training Dataset D
Labelled datapoints
Unlabelled datapoints
Tree split
Transductive Learning: The realistic-synthetic fusion are learned by the transductive term Q_t throughout the whole forest.
Realistic-Synthetic Association ψ
Labelled and unlabelled data are clustered via Q_u, by comparing appearances of patches.
Source space (Synthetic data S)
Target space (Realistic data R)
STR Forest
Viewpoint Classification: Viewpoint classification is first perfromed at he top levels, controlled by the viewpoint term Q_a.
Joint Classification: At mid levels, Q_p determines classification of joints, when most viewpoints are classified.
Regression: To describe the distribution of realistic data, nodes are optimised for data compactness via Q_v and Q_u towards the bottom levels.

Figure 11 -STR model proposed in [32].

In [33] a robust hand parsing scheme is developed to extract a high-level description of the hand from depth images, for that, an RDF is used to classify each pixel of the image considering a depth context feature elaborated with a novel distance-adaptive selection method, then a super-pixel-Markov random field (SMRF) is used to smooth the spatial information and guarantee smaller classification errors. They claim a much higher accuracy than other benchmark and state of the art algorithms, at that time.

To improve the accuracy and efficiency of his previous method, Tang et al. [34] proposed a new forest-based discriminative framework called latent regression forest (LRF). The method learns from a depth image the topology of the hand with unsupervised learning. The main difference of LRF from existing methods is that it employs a structured coarse-to-fine search on a point cloud instead of dense pixels, and an error regression step to avoid error accumulation. Instead of evaluating the hand as a whole, Sun et al. [23] and Wan et al. [24], use a hierarchical approach where in the first case a cascade regression is developed to estimate the hand pose. Their estimations are made for individual parts of the hands in order of their articulation complexity. In the second example, a similar framework is proposed where the joint value estimates are made only from depth maps and take advantage of the tree structured topology of the hand. Instead of considering the differences between the surface normals as features to their frame conditioned regression forest (FCRF) they use the normals directly which improves prediction results for real data compared to previous discriminative methods.

There also have been multiple approaches to solve the hand pose estimation problems using CNNs. Thompson et al. [37] start with an RDF algorithm to segment the hand in the depth map. After some filtering and downsample operations, the resulting depth maps are passed through a CNN to extract the meaningful features and reach an output set of heat-map feature images. Each feature heat-map can be viewed as a 2D Gaussian (Figure 12) whose mean is centred at one of 14 feature points of the user's hand which represent key joint locations in the 3D model (e.g., fingertips). The joint locations are then used in an inverse kinematics (IK) algorithm that can recover a full 3D pose. This solution takes advantage of the offline training of these algorithms that allow to construct robust training information for the models, however it was still difficult for the solution to deal with occlusions which lead the IK algorithm to miss the correct location of the hand joints.

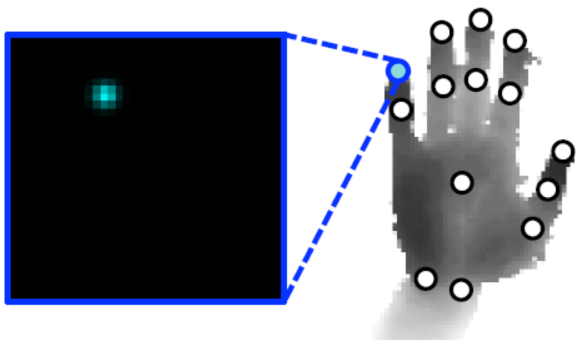


Figure 12-Depth image overlaid with 14 feature locations and the heat-map for one fingertip feature [37].

To solve this problem in [38] a solution is proposed based on local and global regression. Instead of solving the hand pose problem for the whole hand at once with just one deep neural network, the question can be tackled first with a global regression CNN to derive the parameters of the wrist and then use five local regression networks to find the parameters for each of the five fingers. This approach deals well with occlusion problems, and it can avoid the need to re-initialize all parameters when the previous frame is lost. Other examples using depth information are presented in [39] and [40]. In the first case a convolutional neural network is developed considering as input a point cloud of the hand. This network can regress a low dimensional representation of the 3D hand. The second example concerns a 3D CNN which takes as input a 3D volume of the hand encoded in the corresponding point cloud, despite its good results compared to other state of the art methods, this network has problems with its time and space complexities which grow cubically with the resolution of the input 3D volume.

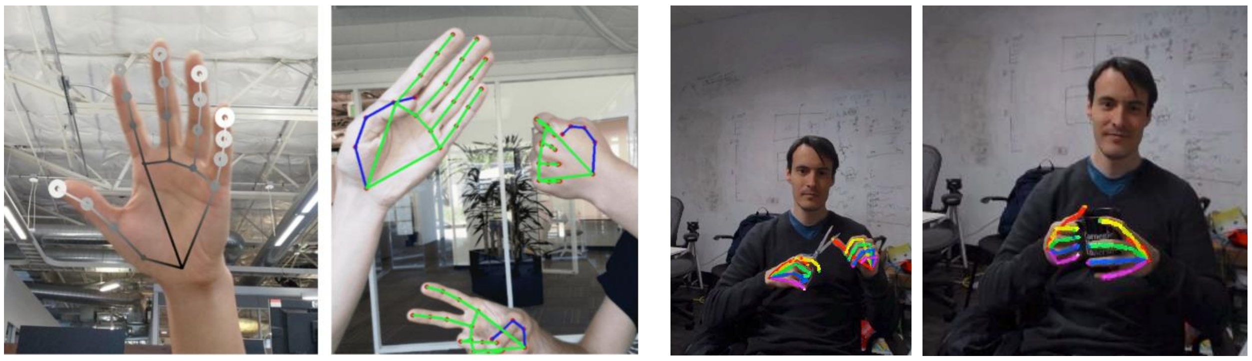

(a) (b)

Figure 13 -Results of discriminative methods for hand keypoints detection:(a) with 2.5D information (left), and multi hand tracking (right) [41];(b) while interacting with objects [42].

Depth is widely used to solve the hand pose estimation task because some authors believe that if a 3D pose is expected, then, 3D spatial information can give the networks more valuable information to reach an accurate prediction. Even though depth can be a great help, solutions using only RGB information are also developed, considering that it still is more convenient for the user since he only needs a regular RGB camera.

Zimmermann and Brox [43] were the pioneers in this line of work by developing a network capable of detecting 3D hand keypoints which are then used to predict the most likely 3D configuration of the hand based on some prior information on the hand structure. The training of the networks is made with a large-scale 3D hand pose dataset based on synthetic hand models.

Also using just RGB data as input, [43] proposes a novel 2.5D pose representation where a novel CNN architecture and leans depth map and heatmap distributions. In the same line, [41] and [42] developed hand keypoints predictors to estimate the hand configuration, designated by “Mediapipe hands” and “Openpose hand detector” respectively. Both methods follow a similar pipeline, where first the hand must be found in an RGB image and only then the predictor model can estimate a 2D skeleton matching the hand’s configuration.

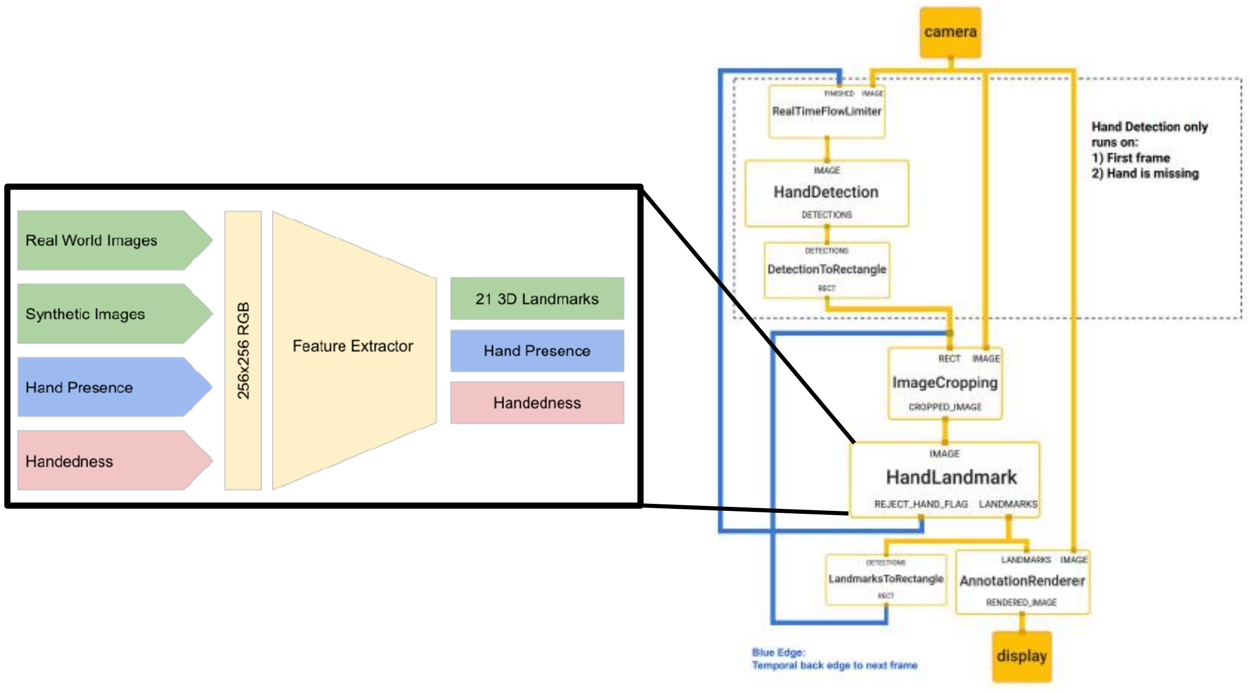


Figure 14- Pipeline used by the “Mediapipe Hands” application including the “HandLandmark” model that estimates the positions of every hand keypoint [41].

This skeleton is made of 21 hand keypoints located at the hand joints according to the hand model considered (Figure 13 (b)). In the case of Mediapipe it can generate 2.5D keypoint information, considering it can also predict relative depth within the hand (Figure 13 (a)). The openpose predictor can also be used to estimate 3D coordinates of hand joints, but it needs multiple views of the hand and a calibration procedure between all the cameras.

There are ready-to-use applications made available by the authors for both methods. These allow other users to utilize the proposed methods more intuitively without having to implement the solutions by themselves. This is a major advantage when comparing with other methods. Their intuitive output is another strength of these solutions, the 21 hand keypoints skeleton is easier to understand when compared to other hand models such as the ones presented for most generative methods.

In the case of Openpose, this is even robust when there is interaction with objects which is of great interest in this work. Mediapipe on the other hand focus on dealing with free hands and presents a better frame rate than Openpose. A summary of some methods found in the literature is presented in Table 3.

Table 3- Summary of discriminative methods for hand pose estimation built mainly on convulotion neural networks (adapted from [16]).

Input	Literature	Datasets	Method	FPS
RGB+Depth	**Thompson et al.** [37]	Self-built dataset	RDF+CNN	24.9
	Sinha et al. [38]	Dexter1, NYU	CNN: DeepHand	32
	Ge et al. [39]	NYU, ICVL, MSRA	CNN	48
	Moon et al. [44]	NYU, ICVL, MSRA, HANDS2017, ITOP	CNN: 3D CNN	35
	Ge et al. [40]	MSRA, NYU, ICVL	CNN: 3D CNN	91
RGB	**Zimmermann and Brox** [43]	Stereo hand pose, Dexter, RHD (rendered hand)	CNN: HandSegNet, PoseNet	-
	Iqbal et al. [45]	Dexter, Ego Dexter, STB, RHD, MPII+NZSL	CNN	150
	Ge et al. [46]	STB, RHD	Graph CNN	50
	Zhang et al. [41]	Self-built dataset	CNN	13[1]
	Simon et al. [42]	Self-annotated MPII and NZSL	CNN: VGG-19, CPM	4[2]

[1]-Value obtained through tests performed on laptop ASUS TUF Gaming A15

[2]-Value obtained through tests with an RTX 3080 (graphics card) and an AMD ryzen 9-12 core processor. Algorithm was at top capacity according to the parameters and equipment available.

2.3.2 Generative methods

Given a hand image, the main goal of generative methods is to find the optimal parameters to construct a hand model that is based on prior knowledge of the hand structure as explained in section 2.1. The optimal parameters are found when the constructed hand model fits the hand shown in the image.

The task of generative methods is composed of four parts, as shown in Figure 15. First should be selected a hand model followed by the initialization of the model's parameters in every frame. One common method is to use the pose from the previous frame as the initial values for the current frame. After that, a similarity or loss function measures, based on hand features/parameters, the discrepancy between the actual hand and its model. Commonly used image features are silhouettes, edges, shading, and depth value. At last, in each frame the parameters of the model are continuously updated until the smallest discrepancy is achieved. Commonly used optimization methods are iterative closest point (ICP)[47] and particle swarm optimization(PSO) [28].

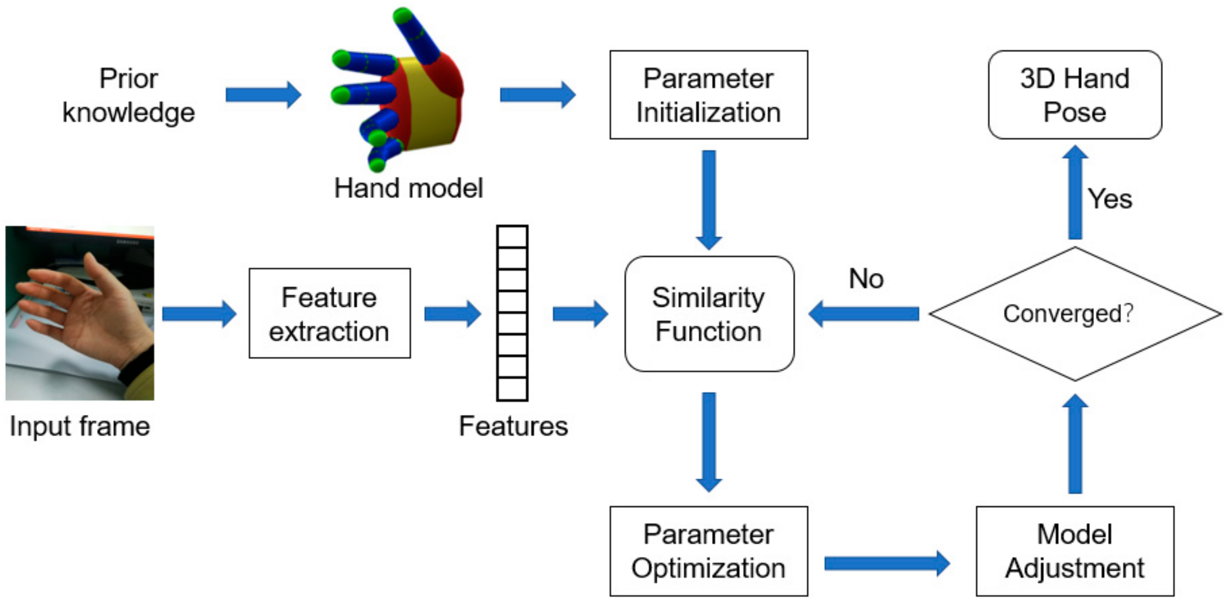


Figure 15 - Workflow of generative methods for hand pose estimation [16].

In Figure 4 and Figure 5 the kinematic hand models presented are intuitive for pose fitting tasks. However, their high dimensionality makes the optimization problem hard to solve in real-time. Currently, generative solutions often use geometric models as their 3D hand models (Figure 16). These are usually composed of simple geometric primitives such as spheres, cylinders, polygons, or their combinations. Splitting the hand model into smaller structures can largely reduce the dimension of the problem and simplify the task complexity to a certain extent. The most widely used geometric models are the **generalized cylindrical model** and the **deformable polygonal mesh model.**

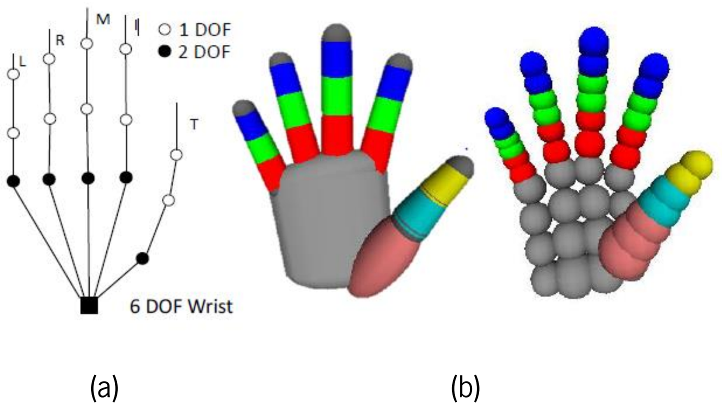


Figure 16- (a)26 DOFs kinematic hand model [48]; (b) Hand models based on different geometric primitives ([49], [48]) Oikonomidis et al.[49] uses a **generalized cylindrical model** for the hand with 26 DoF. This hand model is made of four kinds of basic geometric entities, with the hand configuration being encoded by 27 parameters (Figure 17). The method generates multiple images of the proposed hand model from different points of view and extracts edge and skin features from those images. The same feature extraction procedure is used for the RGB images of the real hand. An objective function is then used to quantify the discrepancy between the predicted and the actual, observed features. The optimal model parameters are the ones that minimize the objective function. This search process is carried out using the optimization algorithm PSO.

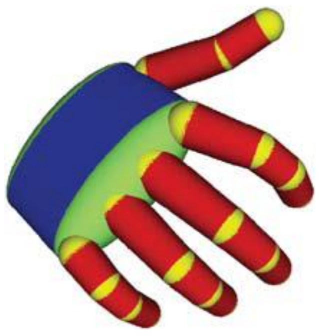
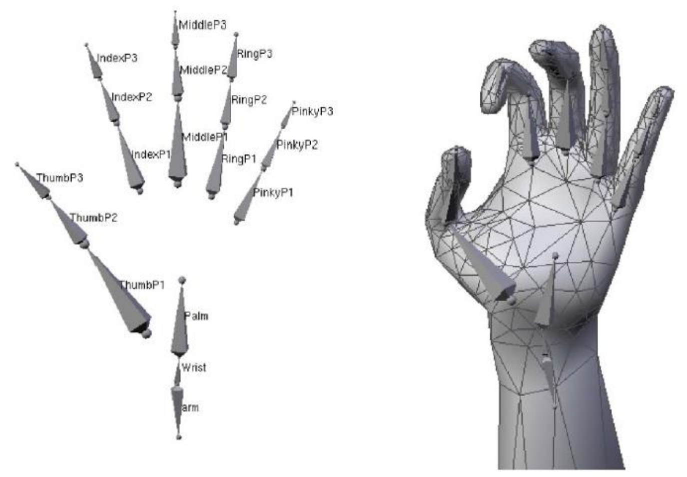


Figure 17- Hand model made of colour-encoded geometric primitives (yellow: elliptic cylinders, red: ellipsoids, green: spheres, blue: cones)[49].

In order to take advantage of depth information, [28] used a Kinect sensor to also acquire the depth map of the hand. This allows the use of depth features to compare the estimated depth maps and the real ones. The optimization algorithm used was PSO.

In an attempt to create a more robust tracking solution, [50] tried to implement generative methods to estimate the hand pose while dealing with objects. To better model this interactions, a 25 sphere model was considered, similar to the one used by Qian et al.[48] which was made of 48 spheres as depicted in Figure 16(b). The latest example also tackled the problem of parameter initialization due to fast motion of the hand, proposing a method that first detects the hand's fingers to then propose initial probable poses. A hybrid optimization (ICP-PSO) method is used to find the optimal hand model parameters leading to faster computation and better resisting local minimums in the objective function.

Deformable polygonal mesh models typically consist of a surface model covering a skeleton model. To get an accurate representation of the hand is necessary to have a method that deforms the surface model according to the configurations defined for the underlying skeleton. La Gorce et al. [51] proposed a mesh surface made of 1000 triangular facets having a total of 28 DoFs while the underlying skeleton had 18 bones and 22 DoFs (Figure 18). As for the objective function it takes into account temporal texture continuity and shading information while handling self-occlusions and time-varying illumination. The minimization of the objective function is performed with a quasi-newton method.



(a) (b)

Figure 18- Deformable polygonal mesh model. (a) Skeleton. (b) Deformed hand triangulated surface [48].

Ballan et al.[52] proposed a generative solution with a hand model similar to the one proposed in [48], but for better accuracy, it adds a salient point detector (Hough forest classifier) which helps find the hand position during an interaction between the two hands and help distinguish both of them. The method adds edges, optical flow, and collision information to the objective function, and can detect the interaction between two hands and objects. However, the method needs heavy computations leading to poor real-time performance. Tzionas et al. [53] develops a similar solution also using the linear blend skinning (LBS) as the deformation method of the surface. While this work only uses a single RGB-D camera, [48] uses a more complex multi camera setup.

As an alternative to the methods above, [54] proposed a technique based on RGB-D input and uses a linear Support Vector Machine (SVM) classifier to find the fingertip position in the depth map. The chosen hand model is the SoG (sum of Gaussian) model, with the corresponding parameters being calculated based on color information. The parameter optimization is performed with the gradient descent method.

Table 4- Summary of generative methods for hand pose estimation (adapted from [16]).

Input	Literature	Features	Hand Model	DOF	Param eters	Optimization method	FPS
RGB	**Oikonomidis et al.** [49]	Skin and edge	GCM[1]	26	27	PSO	-
	Gorce et al. [51]	Surface texture and illuminant	DPMM[2]	22	-	Quasi-Newton method	40
	Ballan et al. [52]	Skin, edges, optical flow, and collisions	DPMM	35	-	Levenberg–Marquard	50
RGB+ Depth	**Oikonomidis et al.** [28]	Skin and depth	GCM	26	27	PSO	15
	Oikonomidis et al. [50]	Skin and depth	GCM	26	27	PSO	4
	Qian et al. [48]	Depth	GCM	26	26	ICP-PSO	25
	Sridhar et al. [54]	Skin and depth	DPMM	26	-	Gradient ascent	10
	Tzionas et al. [53]	Skin and depth	DPMM	37	-	Self-build method	60

[1]GCM: Generalized Cylindrical Model. [2]DPMM: Deformable Polygonal Mesh Model.

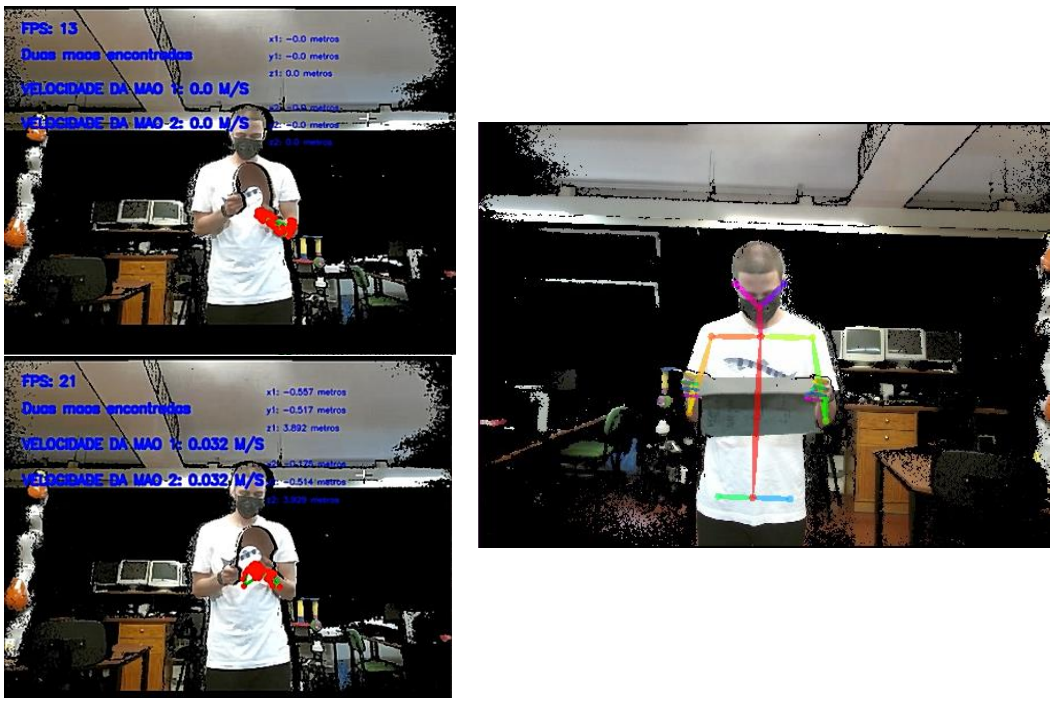
2.4 Hand pose tracking method selection

After realizing which alternatives there are to solve the hand pose tracking problem, is essential that some criterion is defined to choose which one will better suit the needs of this task. According to the approach proposed in section 1.1 most of the mentioned solutions can provide the necessary information to get the trajectories of both hands (position and configuration), however complementary requirements can help decide between them.

Glove-based alternatives are the best solution when high accuracy and precision are required, and they also avoid all the difficulties presented in section 2.3 associated with vision-based systems. However, the price is an obstacle when considering their selection, since most gloves have a price above 1000 dollars. In addition to this, gloves are the least intuitive and practical solution when comparing with vision-based ones, especially if object manipulation is to be considered. For these reasons glove-based solutions were discarded.

In the case of vision-based systems, the main criterion to select a solution was its ease of implementation. From all the methods mentioned, only two discriminative ones, namely [41] and [42], provide a free user-ready implementation of their algorithms and can work with just one camera. After analysing both solutions, two distinctive aspects were found: speed, and robustness.

Mediapipe is considerably faster, however Openpose is more robust to deal with object interactions and scale changes of the hand. Mediapipe finds the hand on the image by finding the hand palm directly which is prone to miss it when it is too small or it is occluded, however Openpose finds the hand based on the body pose estimation which is less likely to fail even when they are far away from the camera or occluded. The more robust body pose estimator is also what makes Openpose lose when it comes to speed performance.



(a) (b)

Figure 19- Results of hand pose tracking using: (a) Mediapipe. Missed hand locations and poorly built skeletons are very frequent but is capable of real time performance; (b) Openpose, which presents robust performance even while dealing with very occluded hands but does not provide results in real-time.

Considering that this work involves tracking the hands during the manipulation of an object, and that in the future it might be useful to track other body parts, Openpose becomes the obvious choice. Real time is not a critical criterion for decision since all the processing can be done offline after recording the movements. To confirm these assumptions, a quick test was made with both algorithms in a recorded inspection movement which confirmed the decision made as it is seen in Figure 19.

3. THEORETICAL FOUNDATIONS AND TOOLS

In this chapter are presented both theoretical principles and tools that were used in the development of the 3D hand tracking system. First, one can found an overview of some theoretical principles that were critical for the development of the solution to be presented. This includes the description and classification of complex objects, conversion from 2D image coordinates to 3D world coordinates, description of orientation of a body in space and also description of the digital filter to be used. In terms of tools, this includes an introduction on the main application supporting the hand tracking mechanism of the developed solution, Openpose; and an explanation about how RGB-D cameras work focusing on the Microsoft Kinect sensor, which is the camera to be used in the remaining parts of this work.

3.1 Theoretical foundations

The two theoretical principles presented next are of the up most interest for the remaining work of this dissertation. First, it is explained how one can know the location of a point in space only by knowing its location on the image caught by the camera and its depth value relative to the camera. This will be useful to get trajectories in real world coordinates for the manipulator. Second, it is described a way of defining the orientation of an object in space that will be useful to later express the hand's orientation as part of the extracted trajectories.

3.1.1 Description and classification of deformable objects for robotic manipulation

In order to comprehend the complexity of the problem tackled in this work, one needs to understand what a deformable object is. The challenges about automatic manipulation of this type of objects come essentially from their complex characteristics when comparing with rigid ones.
The main difference between rigid and deformable objects is how they react when forces are applied to them [4], while the first ones keep their shape, the latter ones change it. The number of degrees of freedom is another way to distinguish between these types of objects. Deformable objects, unlike rigid ones, have an infinite number of degrees of freedom [5], which makes it significantly harder to establish a generalized mapping between an action and a resulting object configuration.
Hence, it becomes clear that robotic manipulation of these complex objects goes beyond control of the just the manipulator itself, as happens with rigid objects. With deformable objects the manipulation strategies must also consider their complex behaviour and the deformations they are suffering. According

to [4], during automatic manipulation it is useful to continuously evaluate the deformations regarding its reversibility, extension and direction on the object. Two types of deformation can be considered concerning reversibility: elastic and plastic, whether the object returns or not to its initial configuration. Additionally, the author mentions a third category that, despite being unusual in the literature, is highly intuitive to describe objects that suffer "flexible deformations" such as cloth or ropes where there is no plastic or elastic deformation, but changes in its shape.

The goal of the manipulation of these complex objects can be to achieve certain deformations or these can be considered as a side effect [4]. In this work the goal is to control the deformations in order to facilitate the inspection of the object, namely the leather piece.

Sanchez et al. [5] also presents a classification based on the physical properties and dimensions of the object. In terms of physical properties, they consider:

Objects with no resistance to compression - They don’t show resistance when approaching two points at their ends. Examples of this type of objects are clothing items, such as sweaters or pants, and ropes.

Objects with tolerance to severe deformations – As the name suggests, even though they present resistance to compression this one occurs without leading the material to rupture. Examples of this type object are sponges and paper sheets.

Regarding the object dimensions:

Uniparametric - One of the dimensions is significantly bigger than the other two. Examples of this type of object are the cables or ropes.

Biparametric - One of the dimensions is significantly smaller than the other two. Examples of this type of object are the paper sheets or cloth.

Triparametric – None of the dimensions stands out from the others. Examples of this type of object are the sponges and food.

By combining the classifications presented, [5] divide the deformable objects into four general types:

Type I: uniparametric objects without resistance to compression (eg.: cables and ropes), or with high tolerance to severe deformations (eg.: tubes or elastic metal bars). Works with this type of objects englobe for example, insertion of ropes with tight tolerance [6], or knot tying [7]. These objects are also known as linear in the robotics field.

Type II: Biparametric objects with high tolerance to severe deformations, as it is the case for cardboards, paper sheets or sponge sheets. Tasks associated with these objects are related to their grasping [8] and folding [9]. In robotics, these are usually called planar.

Type III: Biparametric objects without resistance to compression like clothing, and for that reason are usually called as like-clothing objects. Works with these objects concern their manipulation [10] and folding [11].

Type IV: Triparametric objects with high tolerance to severe deformations, like sponges and food. Examples can be found in the literature related to these objects involving their manipulation [12] and grasping [13]. These objects are usually known as solid or volumetric.

In Figure 20 is possible to see a diagram that summarizes the classification of deformable objects.

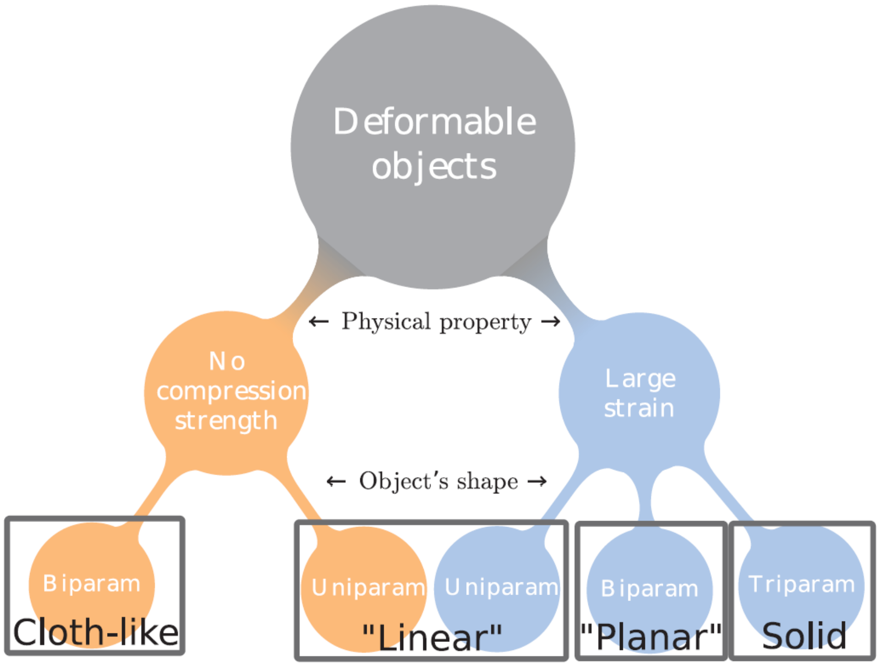


Figure 20 - Proposed classification of deformable objects [5].

In this work, the objects considered for the automatic inspection procedures are of type II (" Planar"), with two of its dimensions being significantly bigger than the other and also because it can tolerate severe deformations.

3.1.2 From image coordinates to 3D world coordinates

According to [55],

"a camera is a mapping between the 3D world (object space) and a 2D image"

, and this mapping is usually represented through the camera models that take as input the world coordinates of some point and calculate the corresponding location within the camera images. However, these models are just an approximation of the physical process that occurs when an image is captured. The same author presents several camera models that can be used according to the type of camera considered. One model derived for CCD sensor-based cameras is the basic pinhole model, which considers the camera has having its projection centre at a finite distance.

The basic pinhole model considers that all the points in the world space are projected into the image plane, whose centre is aligned with the centre of projection located at the origin of a Euclidian coordinate system as shown in Figure 21. The centre of projection is called the camera centre (point C in Figure 21) and the Z axis, which is the line from the camera centre perpendicular to the image plane is called the principal axis. Point b where the principal axis meets the image plane is called the principal point and is also the origin of the coordinate system for the image plane.

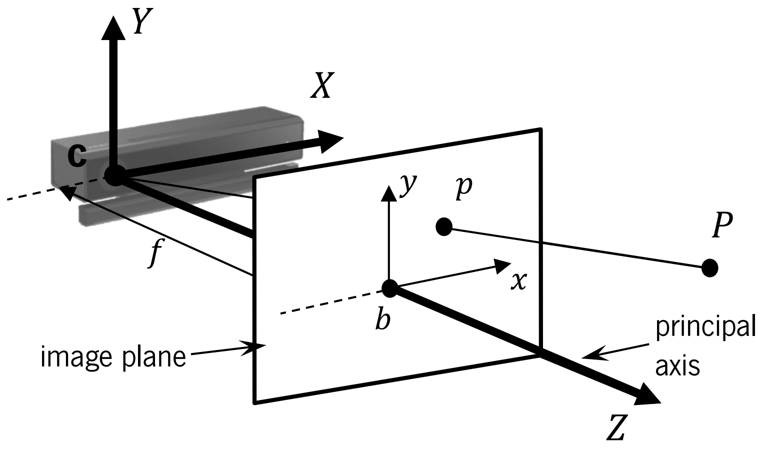


Figure 21-Layout considered in the basic pinhole moel. The centre of the camera is the centre of projection located at the origin of the coordinate system and the corresponding image plane is located in front at a distance f, also called focal distance.

Under the pinhole camera model, a point P in space with coordinates $P = \left(P_x , P_y , P_z \right)^T$ is projected into the image plane through a line joining it to the camera centre. The point where this line intercepts the image plane is the projection point p. By using similarity of triangles (Figure 22), one can calculate the coordinates of point p as

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \rightarrow \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \frac{fP_x}{P_z} \\ \frac{fP_y}{P_z} \end{bmatrix}, \tag{3.1}$$

Which describes the central projection mapping from world to image coordinates. This is a mapping from Euclidean 3D-space $\mathbb{R}^3$ to Euclidean 2D-space $\mathbb{R}^2$.

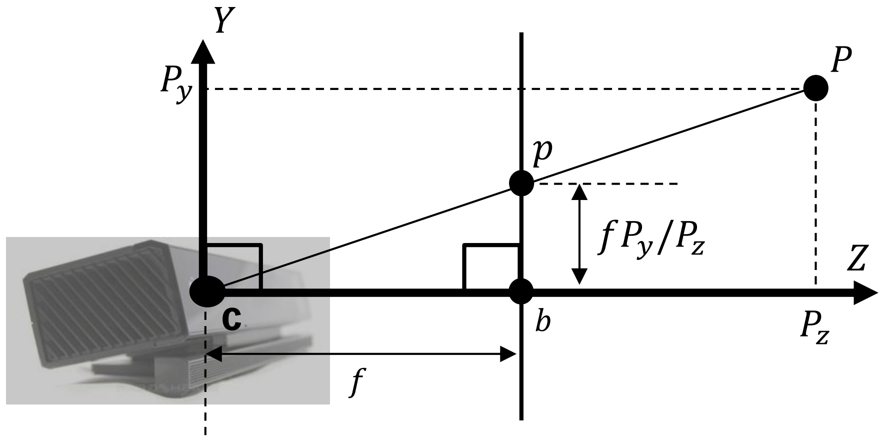


Figure 22 – Similarity of triangles used to calculate the coordinates of point p.

The model presented until now considers that the coordinate systems of both the image plane and camera plane are aligned, which might not always be the case, as shown in Figure 23. Here the image coordinate system (x, y) is in the bottom left corner and not aligned with the camera coordinate system (X, Y).

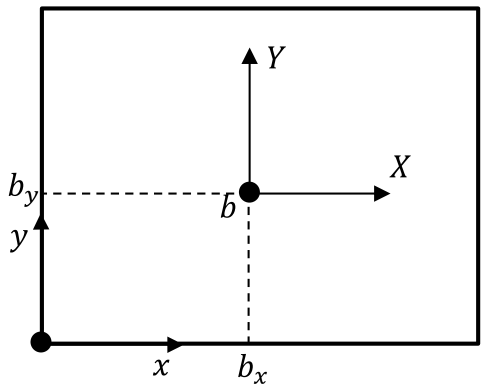


Figure 23 – Representation of the image coordinate system (x, y) and the camera coordinate system (X, Y) when they are not aligned.

For the cases where this happen, it is necessary to apply a transformation to the image coordinates to centred them with the camera, namely a translation with the vector (b_x, b_y). Thus, equation (3.1) becomes,

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \rightarrow \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \frac{fP_x}{P_z} + b_x \\ \frac{fP_y}{P_z} + b_y \end{bmatrix}. \tag{3.2}$$

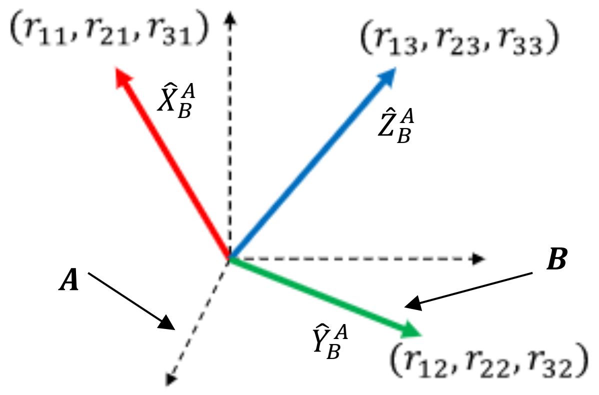
Until now, it was only established a way of mapping from 3D world coordinates to image coordinates but in this work the inverse mapping is needed. This can be obtained by solving the system equation (3.2) with respect to the coordinates P_x and P_y,

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} \frac{(p_x - b_x)P_z}{f} \\ \frac{(p_y - b_y)P_z}{f} \\ P_z \end{bmatrix}. \tag{3.3}$$

Equation (3.3) maps from image coordinates (p_x, p_y, P_z) where P_z is assumed to be known, to 3D world coordinates (P_x, P_y, P_z). For example, if an RGB-D camera is considered, P_z could be the distance measured by the camera to the point P. The pinhole model used to derive these equations is quite simple, for example, more robust alternatives could include factors to correct problems from radial and tangential distortion.

3.1.3 Description of an object's orientation in 3D space

According to [56] there are multiple conventions that can be used to describe the orientation of an object relative to a reference coordinate system, one of them is to specify a square matrix with nine elements (equation (3.4)) where each column represents the director vector of each axis of the rotated orthogonal coordinate system as depicted in Figure 24.



(r_{11}, r_{21}, r_{31})
(r_{13}, r_{23}, r_{33})
$\hat{X}_B^A$
$\hat{Z}_B^A$
A
B
$\hat{Y}_B^A$
(r_{12}, r_{22}, r_{32})

Figure 24- Representation of the orientation of the coordinate system B relative to the reference system A (dashed line).

The coordinates of these director vectors are relative to the reference coordinate system. Considering the reference system as A and rotated coordinate system as B, one can define the orientation of B relative to A with matrix,

$$R_B^A = [\hat{X}_B^A \quad \hat{Y}_B^A \quad \hat{Z}_B^A] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{3.4}$$

A simpler way of defining the orientation of an object is to represent this nine element matrix with just three values. [56] presents several representations of this type, with one being particularly common in the robotics field, the **roll-pitch-yaw** (α, β, γ) representation. This representation consists in first, making both coordinate systems, A and B, coincident and then applying three consecutive rotations to B about the three orthogonal axes of A that stays fixed (Figure 25), making it achieve the same final orientation depicted in Figure 24.

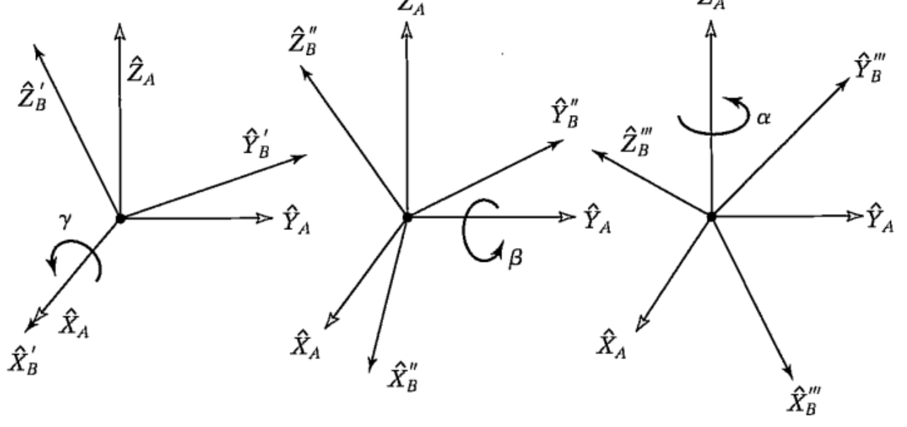


Figure 25-Roll-Pitch-Yaw angles. First a rotation of B around $\hat{X}_A$ is applied, then around $\hat{Y}_A$ and the last one around $\hat{Z}_A$ [56]

According to [56] the final orientation of B in relation to A can be represented by the matrix,

$$R_B^A(\gamma, \beta, \alpha) = R_Z(\alpha) R_Y(\beta) R_X(\alpha) = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & s_\gamma c_\beta & c_\gamma c_\beta \end{bmatrix}, \tag{3.5}$$

where c stands for the cosine function and s for the sine function. There are two main purposes for this representation: use it to find the final orientation of a body when rotated as in Figure 25 with a specified value for the roll, pitch, and yaw angles; find which roll-pitch-yaw angles are equivalent to a certain orientation. The latter is for the cases where the nine-element matrix of equation (3.4) is known beforehand but a more compact representation is desired. In these cases, the roll-pitch-yaw angles are calculated as follows,

$$\beta = Atan2(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}). \tag{3.6}$$

$$\alpha = Atan2(\frac{r_{21}}{c\beta}, \frac{r_{11}}{c\beta})$$

$$\gamma = Atan2(\frac{r_{32}}{c\beta}, \frac{r_{33}}{c\beta})$$

where Atan2(y, x) is a two-argument arc tangent function[1].

3.1.4 Low pass filter

One common filter used in motion capture is the IIR (Infinite Impulse Response) digital low pass filter, which according to [57] is a good solution considering that human motion mainly consists of low frequencies. These IIR filters are also a way of achieving long impulse responses without having to use high order FIR functions.

The design of an IIR filter is a nonlinear problem, and there are no general optimal design methods. There are however different construction methods, which can give optimal solutions for some special cases. The most known classical IIR filter methods are Butterworth, Chebychev and elliptical. With the first being chosen in this case.

Butterworth filters were developed initially for the processing of analogue signals however ended up being good solutions to help the construction of digital filters. The amplitude response of a Butterworth filter is given by,

$$|H(j\omega)| = \frac{K}{\sqrt{1 + \left(\frac{j\omega}{j\omega_c}\right)^{2n}}} \tag{3.7}$$

By analysing the graphical representation of equation (3.7) presented in Figure 26 is possible to see that this filter tries to maintain the input signal for frequencies below the cut-off frequency (ω_c) and blocks signal at higher frequencies as one can see by the gain drop. Therefore, two main components influence the performance of this filter, its order n and its cut-off frequency.

[1] Atan2(y, x) computes $\tan^{-1}(\frac{y}{x})$but uses the signs of both x and y to identify the quadrant in which the resulting angle lies.

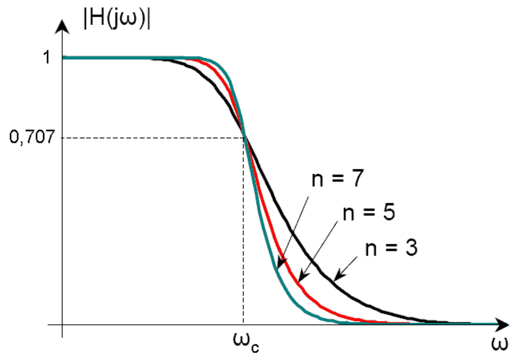


Figure 26 - Butterworth filter amplitude response[58].

According to [59], the transfer function of an nth order Butterworth filter can be represented as,

$$H(s) = \frac{K}{\left(\frac{s}{\omega_c}\right)^n + a_{n-1}\left(\frac{s}{\omega_c}\right)^{n-1} + \cdots + a_1\left(\frac{s}{\omega_c}\right) + 1} \tag{3.8}$$

but since this is an analogue filter it is necessary to use a bilinear transformation which resorts to the z-transform to convert this to a discrete domain. For this, in equation (3.8) s becomes

$$s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right), \tag{3.9}$$

and H(z) is obtained,

$$H(z) = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1 + \sum_{j=1}^{N} a_j z^{-j}}. \tag{3.10}$$

From equation (3.10) the digital IIR filter equation can be derived using the determined coefficients a and b,

$$y(n) = \sum_{i=0}^{M} b_i x(n-i) - \sum_{j=0}^{N} a_j y(n-j). \tag{3.11}$$

Equation (3.11) can be used to filter the desired data recursively.

3.2 Tools

The main tools used in this work are the Openpose application and the camera to record all the inspection movements. In the case of Openpose it is made an explanation about its origin, how it works and possible ways of using it. This application will later be part of the complete 3D hand tracking solution to be developed. In this chapter is also selected a low-cost depth camera that will be used to record the inspection movements on the objects. Some key characteristics of the selected equipment and its working principles are presented.

3.2.1 Openpose

Once the tracking algorithm was chosen, and before start using it, is necessary to understand the basics of how this method works and what type of information needs as input and gives as output. As was said in section 2.3.1, deterministic methods heavily rely on training data to present good results and most of the times it's difficult to obtain good sets of real labelled images to train these models. For this reason, many of the developed deterministic methods also focus on developing clever ways of generating this labelled data [60], or in making the training process more robust. According to [42], Openpose tries to do both by proposing a training process that not only improves the hand keypoint detector but also the training data used during this process. This procedure is called multiview bootstrapping and starts with a weak detector trained on a small dataset. This detector is then used in a multicamera setup to estimate the hand keypoints in some good image views of the hands shown to the cameras. Then, failed or missed detections are identified via robust 3D triangulation and those estimates are ruled out. By reprojecting the 3D hand joints estimated correctly, views of the hand with severe occlusions can be correctly labelled and added to the training data for the detector. Excluding missed detections and including challenging images in the training data iteratively improves the quality of the detector. The strength of this method is exactly this robust multiview training setup that can generalize the results for cases where just one camera is used.

The detector used is based on Convolutional Pose Machines (CPMs) [61]. It predicts a confidence map for each keypoint of the hand with the location being represented as a Gaussian distribution centered at the true position (Figure 27). The final position for each keypoint is obtained by finding the maximum peak in each confidence map. All models used by Openpose are already trained and ready for the user.

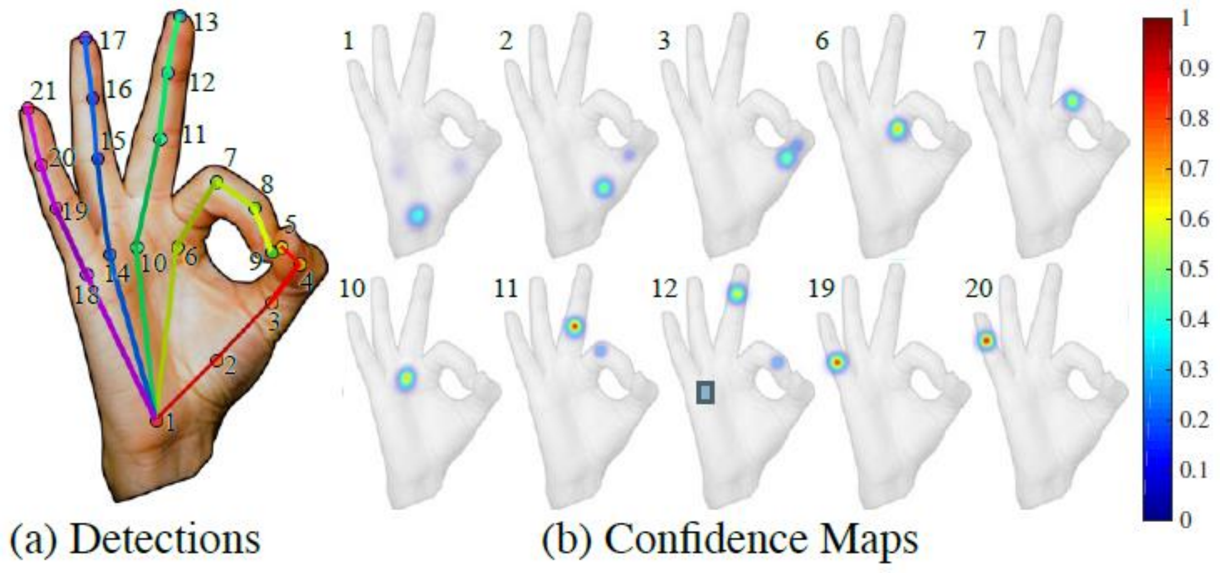


Figure 27- (a) Hand input image with the skeleton of 21 detected keypoints. (b) Selected confidence maps produced by the detector, visualized as a "jet" colormap overlaid on the input [42].

As shown in Figure 27, when an image of a hand is evaluated by Openpose it predicts the 2D coordinates within the image (pixel coordinates) of 21 hand keypoints, that together form the hand skeleton representing the hand pose. However, the hand keypoint detector assumes that the input is a cropped image of the hand which must be found first in the global image. Its location can be provided by the user or it can be found automatically using a body pose estimator that gives the position of the wrist.

According to Openpose's documentation[2], there are two ways of using the application: through its demo or by installing it from source and use it via a Python API. To facilitate the integration of Openpose in our final processing framework it was installed from source in Linux 16.04 and used via Python 3.5. Most of the installation procedure is up to the user, including the manual installation of multiple dependencies (Cuda, CUDNN, OpenGL etc.) making this a toilsome task.

Openpose provides multiple parameters (called "flags") to adjust the application performance according to any specific requirements that different applications may have. The whole list is available in their documentation, but here only are presented the ones that directly affect the 2D hand tracking performance of the application.

[2] https://github.com/CMU-Perceptual-Computing-Lab/openpose

The available flags are:

hand – Enables hand keypoint detection. Set to "**True**".

hand_detector – Defines the hand tracking method to use. 1 – To use the body detector, option recommended by the authors; 2 – To indicate that the user provides the hand location; 3 – To use in addition to the body predictor an extra method that searches for the hand in previous locations (this option was tested but didn't work). Set to "**1**".

hand_net_resolution- Changes the net resolution for convolution stages. If it is increased the accuracy potentially increases but if it is decreased the speed improves. According to the authors for maximum speed-accuracy balance the values used should keep the aspect ratio as close as possible to the images or videos to be processed. The algorithm will do this automatically for the user if the value -1 is passed in either dimension. Set to "**-1x368**".

hand_scale_range- Total range between smallest and biggest scale considered for the hand. The best results for the authors were obtained with it being set to "**0.4**".

hand_scale_number- Total number of scales to consider. Again, the best results obtained by the authors was with the flag set to "**6**".

number_people_max – Number of people to expect in the images. More people make the application slower. Set to "**1**".

maximize_positives - It reduces the thresholds to accept a person candidate during body pose estimations. It highly increases both false and true positives. Set to "**False**".

Considering that Openpose only gives 2D information (pixel coordinates) about the hand configuration but 3D information is needed, the user must find a way to measure the third dimension. One way of doing it is by using a depth camera that provides a corresponding depth map for the RGB images. By combining the 2D coordinates with the depth values is possible to generate 3D world coordinates for every point in the image plane as shown in Figure 28.

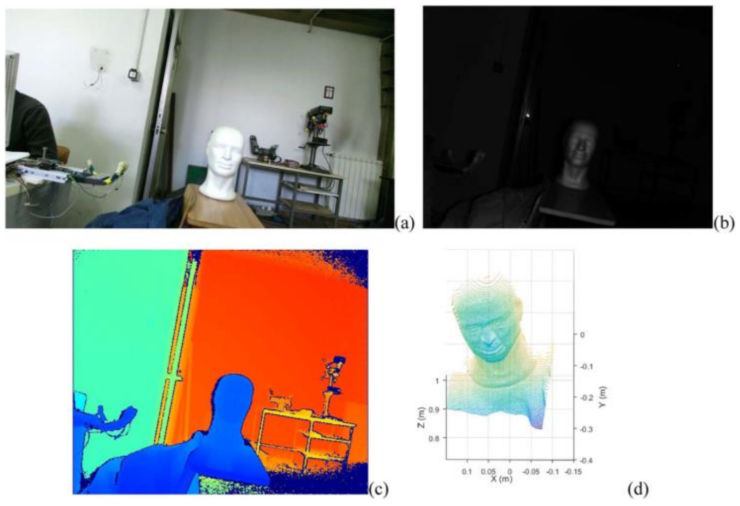


Figure 28- Kinect sensor output: a) RGB image; b) IR image; c) Depth image (different colors mean different depths); d)Plot of the 3D coordinates array [62].

The next step is to select a depth camera to start working on a solution that integrates all streams of information, both from Openpose and the depth sensor.

3.2.2 Low-cost depth cameras

Depth cameras are widely used in vision-based solutions not just because of the valuable information they can provide, namely the corresponding RGB and depth information, but also because of how easy it is to reach these devices today. The price and connectivity are also advantages.

The methods to obtain depth information with cameras are structured light, stereo triangulation, time-of-flight, and interferometry[3] [63].

Most low-cost depth cameras work roughly the same way and have three main components: a low-resolution RGB color camera, an infrared low-resolution camera, and laser projector to collect the data provided from the infrared camera. The additional integrated infrared components are the main difference to 2D color cameras, which makes it possible to get the 3rd dimension, the depth data.

In the market is possible to find many alternatives for RGB-D cameras, but the one used in this paper is the second version of the Kinect sensor since this model was already available for the project. This camera uses time-of-flight to measure the depth of each pixel in the RGB image. The basic operation principle in

[3] **Structured light:** Illuminate the scene with a pattern of light and analyse the deformation of the pattern.
Stereo triangulation: With more than one camera, find corresponding points and perform a triangulation.
Time-of-Flight: Radar-like method (e.g., LiDAR).
Interferometry: Measure the phase shift of the reflected light compared to the source light and deduce the depth.

.

a TOF device is based on measuring the time it takes for light wave to travel from emitter to object and back to sensor (Figure 29).

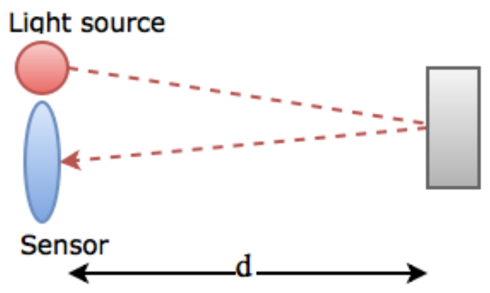


Figure 29 - Time-of- Flight operation principle [64].

Let d be distance from the sensor, then the simplest case can be expressed

$$d = \frac{t_e - t_r}{2} c \tag{3.12}$$

where t_e and t_r represent time for light pulse emitting and receiving, c is speed of light in air.

The Kinect v2 is equipped with an IR emitter and a depth sensor. The IR emitter consists of an IR laser diode to beam modulated IR light to the field of view. The reflected light will be collected by the depth sensor, a state-of-the-art 512x424 CMOS array of deferential pixels. Each pixel has two photo diodes (A, B) being controlled by the same clock signal that controls wave modulation. Photo diodes convert captured light into current which can be measured.

A timing generator is used to synchronize the actions of the IR emitter and the depth sensor. The depth of each pixel can be calculated based on the phase shift between the emitted light and the reflected light. A clever design in Kinect v2 is that the IR emitter is periodically turned on and off, and the output from the depth sensor is sent to two different ports when the light is on and off, respectively. Let the output when the light is on be A, and the output when the light is off be B.

A contains the light from both the emitted IR light and ambient light (i.e., the light already in the field of view), and B contains only the ambient light. Hence, subtracting B from A (i.e., A-B) gives only the reflected modulated light from the IR emitter, which can be used to calculate the depth accurately. Furthermore, the magnitude of (A-B) gives a high-quality IR image without ambient lighting. A narrowband-pass filer is also used to block all light except in the 860nm wavelength range that corresponds to infrared illumination system wavelength.

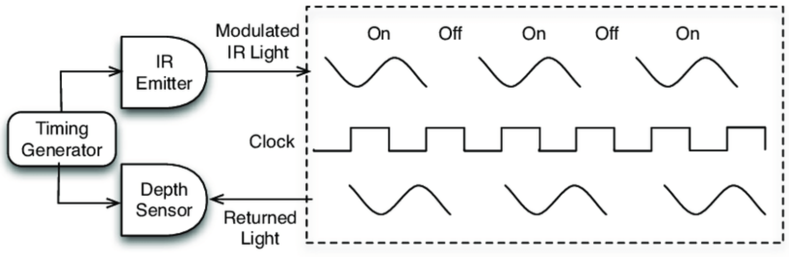


Figure 30- Time of flight technique used in Kinect V2 to measure depth values.

In Table 5 a summary of critical information about the Kinect v2 is presented.

Table 5- Summary of some important features of the Kinect v2.

Feature		**Kinect v2**
Depth sensing technology		Time of flight
Colour image	Resolution	1920x1080
	Frame rate	30 fps (12 fps with low light)
IR Image	Resolution	512x424
	Frame rate	30 fps
Depth Sensing	Resolution	512x424
	Frame rate	30 fps
Field of View		>43° vertical; 70° horizontal
Depth Sensing Range		0.5 m - 4.5 m

To access all the streams of information provided by the Kinect sensor, one must use the Microsoft Kinect SDK (Software development kit) or other third-party software toolkit. In this work the library *pylibfreenect2*[4] was used to open the streams from Kinect, this module was developed in C/C++ to access all the Kinect information similarly to the Microsoft SDK but establishes a Python interface which is useful considering the intention of implementing the overall solution in Python. The streams made available by this library are:

- RGB image frames.

[4] https://github.com/r9y9/pylibfreenect2

- Depth image frames (Depth map).
- IR image frames.
- Resized RGB image frames (RGB with correspondence to depth).

The last stream mentioned is one of particular interest for this work since it is the one that gives an exact mapping between RGB and depth information. Looking at Table 5, namely for the resolution of both RGB and Depth streams, is possible to see that there is no direct correspondence between a location in the RGB and depth images. However, Microsoft solved this problem by using the camera's physical parameters to derive a resized RGB frame that corresponds to the depth map. From this point on, every time the RGB frame from Kinect is mentioned it concerns the one that allows this mapping.

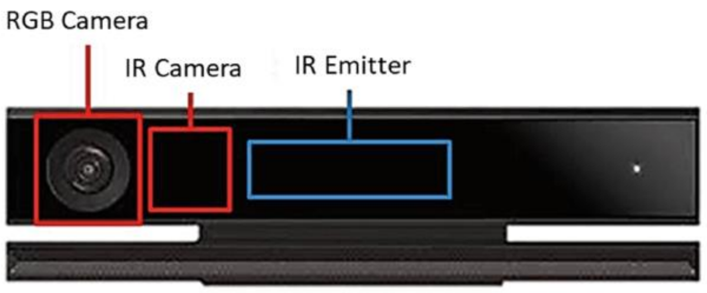


Figure 31- Version 2 of the Kinect sensor with the location of the IR hardware and the RGB camera [65].

This new Kinect version also allows an easy connection to the computer via USB, something that was not possible with the previous version. At this point is possible to access the multiple Kinect streams/frames via Python which facilitates further pre-processing and processing stages.

4. Analysis and Development of the 3D Hand tracking system

The development of the 3D Hand tracking system is described throughout this chapter. It starts with a brief description of the general idea behind the solution to be presented and its corresponding pipeline is also presented. The solution is divided in two main modules: the first is related to recording the hand movements, and the second concerns the actual extraction of trajectories from the images. A complete description is made on both modules along with a critical analysis on the results and decisions that are made during the design process.

4.1 Proposed pipeline

Considering the information provided by the Kinect sensor and the work principles of the Openpose application, a pipeline was proposed to get the hand trajectories from demonstrated inspection movements as presented in Figure 33. First the movements are recorded with the Kinect camera as in Figure 32, and every frame (RGB and corresponding depth) is stored individually.

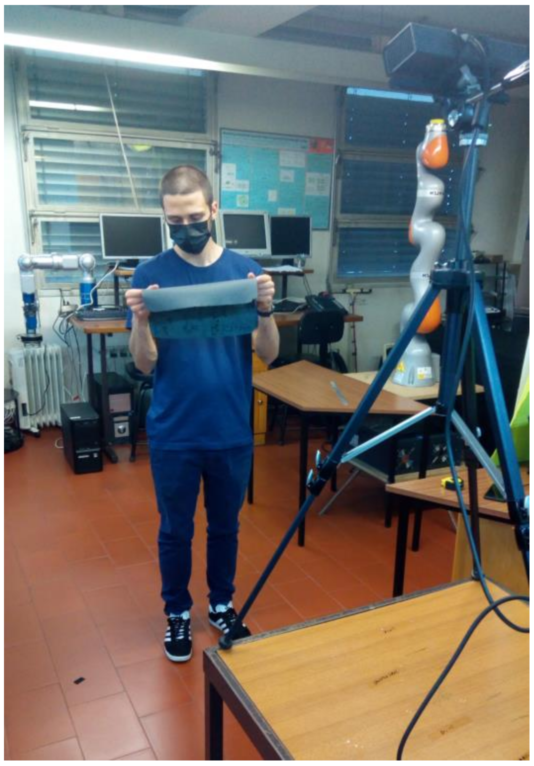


Figure 32 – Example of the inspection movements being “demonstrated” to the camera.

This procedure is included in the recording module of this solution. After every frame is stored, one by one they are fed to Openpose which predicts the location of 21 hand keypoints for each hand. These pixel coordinates are converted into real world 3D cartesian coordinates and the points of interest are selected to define position (x, y and z values) and orientation (roll, pitch and yaw angles) of the hand. At this point there is only a raw trajectory that is still very susceptible to noise and to attenuate that a low pass filter is used. To compensate for a possible lack of frames and information a linear upsample is also applied to get the final trajectories ready to be executed by the robot.

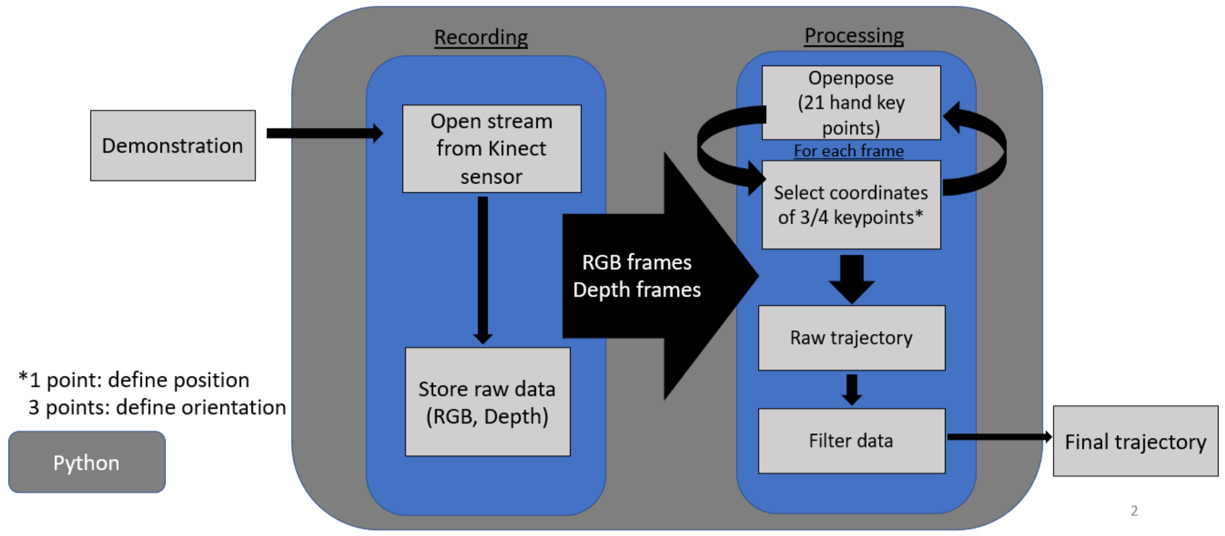

Recording
Processing
Demonstration
Open stream from Kinect sensor
Store raw data (RGB, Depth)
RGB frames Depth frames
Openpose (21 hand key points)
For each frame
Select coordinates of 3/4 keypoints*
Raw trajectory
Filter data
Final trajectory
*1 point: define position
3 points: define orientation
Python
2

Figure 33- Proposed pipeline to get hand trajectories from recorded inspection movements.

Next, every step of this pipeline will be explained in more detail pointing all the details that had to be considered when developing every stage. All the problems encountered during these steps are also discussed.

4.2 Recording module

Once the access to the Kinect streams was established and the Openpose application was ready for use, some early tests were made to check what information was possible to obtain at this point. For this matter, a pipeline such as the one described in Figure 34 was created to understand how well the Openpose could track the hands during the manipulation of a leather piece.

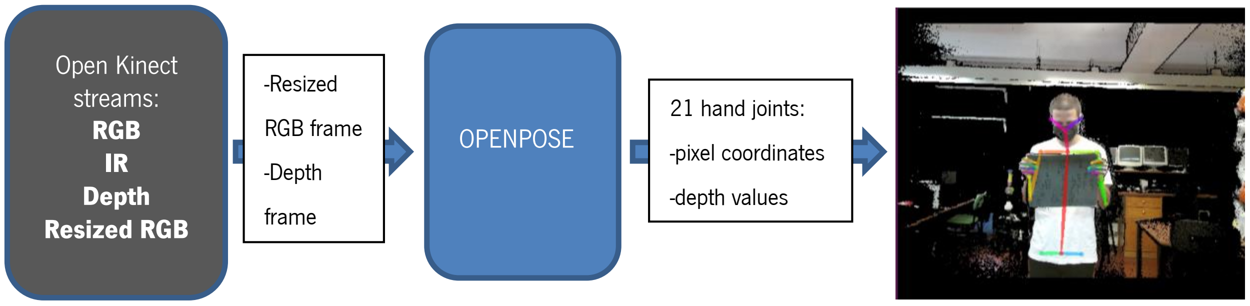


Figure 34- Initial pipeline to get the 21 hand keypoints for each hand in each frame of video stream.

The early results from this test were useful to realize that besides the good performance in finding the hands and estimating their skeleton, it was not possible to process a video stream in real time with Openpose leading to the loss of significant parts of the movements recorded.

As explained in section 2.4, this poor real-time performance is mainly due to both tracking methods used by Openpose (body and hands), since these require a lot of computational resources. Considering this, one way of improving its performance could be to replace the body predictor with a simpler and faster method to find the hands within the image. Therefore, another method was developed to find colored bracelets through color segmentation and assumes their location as the hands' location as shown in Figure 35. This approach presented significant improvements compared to the initial solution (4 to 5 times faster). However, the framerate achieved would still not be over 5/6 frames per second which is still too low to capture all the necessary information about the movements.

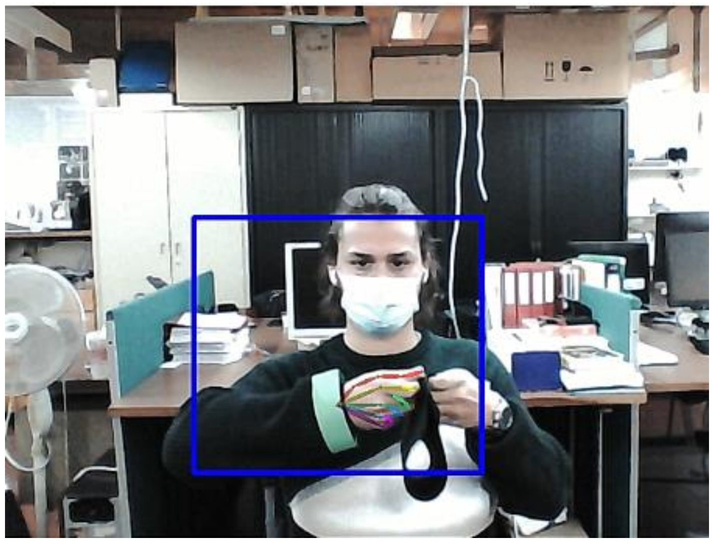


Figure 35- Right hand found by locating the green bracelet in the image. Then the hand skeleton is estimated for the hand within the cropped image(blue rectangle).

Even though removing the body predictor improves the algorithm speed, three main reasons still prevail to maintain it in the pipeline. First, the improvements are not sufficient to justify the implementation of a new solution that would have to be at least as robust as the body predictor and quicker; second, even though it would be desirable to make this a real time solution this is not a critical or needed characteristic for the solution to work; third and last reason, for other steps of this project it might be useful to track other body parts (e.g.: elbows) to further optimize the robot's motion planning and make them more human like.

Taking all of this into account, it was assumed that all the processing would be made offline after the recordings were made. Therefore, an extra step had to be introduced to guarantee that every video frame is processed, and the movements are accurately analysed. First, they are recorded with the Kinect sensor and stored frame by frame, instead of forwarding them directly to Openpose. This way, the frames can be fed one by one to the application ensuring that no useful information is lost.

The developed algorithm for the recording module is presented in Figure 36.

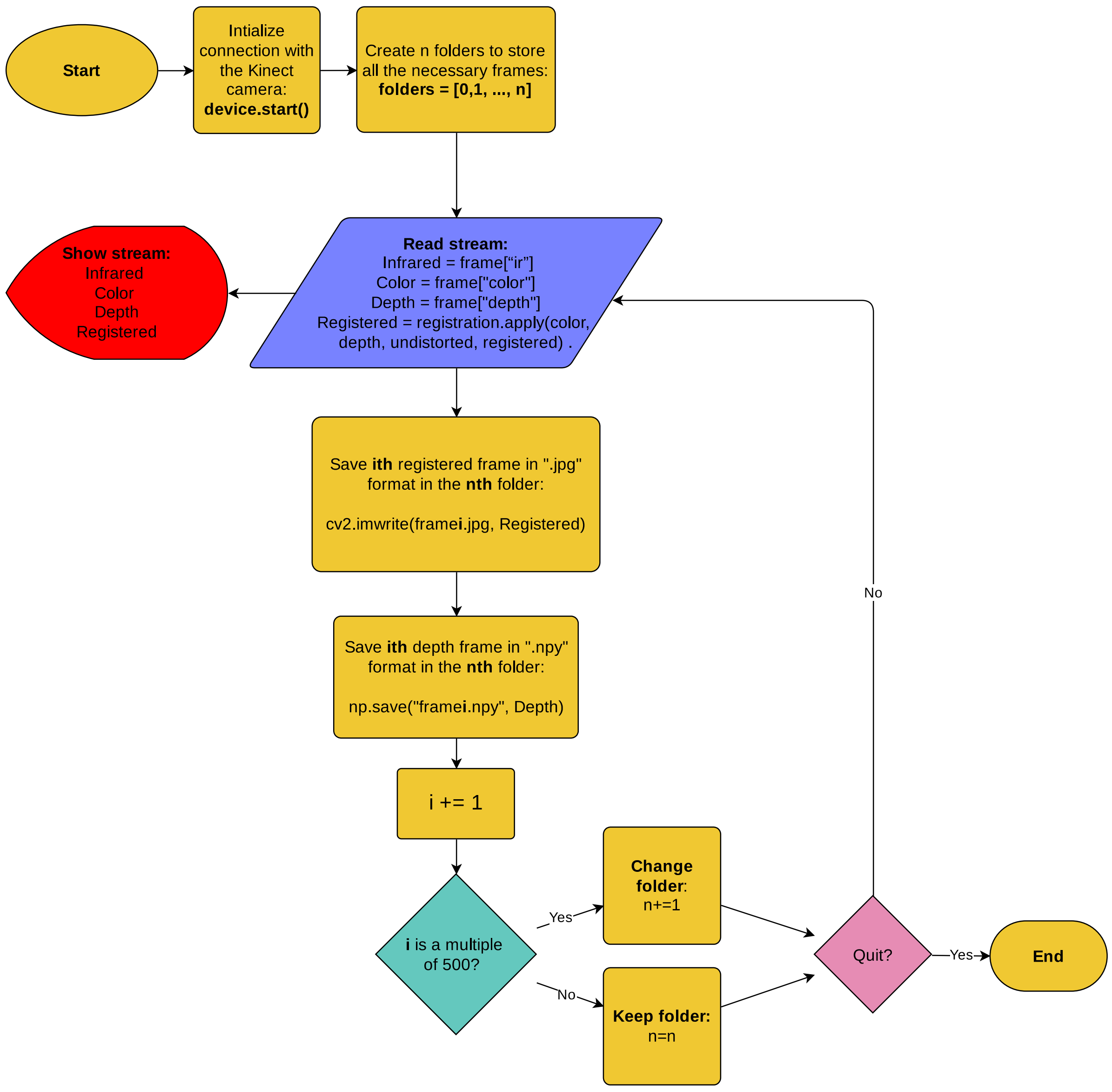

Start
Intialize connection with the Kinect camera: device.start()
Create n folders to store all the necessary frames: folders = [0,1, ..., n]
Read stream: Infrared = frame["ir"] Color = frame["color"] Depth = frame["depth"] Registered = registration.apply(color, depth, undistorted, registered) .
Show stream: Infrared Color Depth Registered
Save ith registered frame in ".jpg" format in the nth folder: cv2.imwrite(framei.jpg, Registered)
Save ith depth frame in ".npy" format in the nth folder: np.save("framei.npy", Depth)
i += 1
i is a multiple of 500?
Yes
No
Change folder: n+=1
Keep folder: n=n
Quit?
No
Yes
End

Figure 36 - General algorithm proposed for the recording module. It reads and stores all the necessary information for the following steps in the pipeline.

With the tools provided by the library *pylibfreenect2* the four available channels from the Kinect sensor are accessed via Python and displayed to the user. Once this connection is established, every RGB (also called "registered" in Figure 36) and corresponding depth frames are stored in a set of folders created in the directory provided initially by the user. These folders are needed to avoid overflooding the memory buffer while saving all the frames. If all the images were stored in only one folder, the process of writing data to it would take a lot more time once the folder starts getting "full" and the buffer memory wouldn't be capable of holding the remaining frames displayed in between. To minimize this effect, multiple folders are created, and the frames are spread amongst them. Each folder takes up to 500 frames.
Despite the efforts, all the processing steps including the ones from the library *pylibfreenect2*, from the moment an image becomes available in the Kinect sensor until it reaches one of the folders where is being stored, leads to a loss of frame rate. This means that, even though the Kinect sensor acquires the data at 30 frames per second they only become available for further analysis at about 15 frames per second. The registered frames, as are called the RGB frames, are stored in .jpg format to save memory space. The depth frames are saved in the npy[5] format, which is provided by the python library *numpy*.

4.3 Processing module

The goal of this work is to capture the essence of the inspection movements that deform the leather pieces in a way that facilitate the detection of defects. However, to do that robots need to manipulate the leather pieces similarly to the way human operators do, and that consists in trying to mimic the way they move their hands while doing the specific inspection movements. To copy these actions both hand position and configuration must be tracked over time and that is what the processing module does.
It was already established that Openpose provides the location of 21 keypoints for each hand. However, it is necessary to understand how to use those coordinates to generate the trajectories for the robot to execute and deform the leather pieces as desired. Ultimately the hand information retrieved must make sense to the robot's gripper since this is the one that will replace the hand during these movements.
To tackle this problem, first it must be considered that the human hand has a lot more DoFs than most conventional grippers such as the one to be used in this project (Figure 3). For example, Openpose considers a hand model with 21 DoFs while the gripper in Figure 3 only has 1, excluding in both cases

[5] https://numpy.org/devdocs/reference/generated/numpy.lib.format.html

the overall position and orientation in space. Therefore, it's difficult to establish a correlation between a certain pose from human to robotic hands as discussed in [66]. In the case of manipulation of deformable objects, this mapping is particularly difficult considering that the physical relation between the hand and the object as an impact on how the last one deforms. If all degrees of freedom of the hand are used during manipulation it becomes very difficult, if not impossible, to achieve similar deformations with a gripper with just one degree of freedom. Therefore, is essential to reduce the number of degrees of freedom used by the hand during manipulation, ideally to only one DoF, to facilitate the reproduction of the deformations with the robot gripper.

To simplify this mapping from the hand to the robot gripper three conditions are assumed:

- only a small set of inspection movements are considered.
- the type of grasp executed with the human hand is as presented in Figure 37.
- The movements already start with the leather piece on the user's hand, and the grasping positions don't change during the manipulation.

By using this "fixed" configuration of the hand its DoFs are reduced to 7 like the robotic gripper considered. These degrees of freedom include closing/opening (1), overall position (3) and orientation (3). If the hand is considered to be always closed during the manipulation of the objects this DoF also "disappears" and the mapping problem is reduced to just extraction of the overall position and orientation of the hand.

To extract position, the problem is relatively simple since only information about one point of the hand has to be considered. However, to get the hand's orientation the problem is more complex, since at least three points are required, and these must represent a meaningful orientation with respect to the deformations implied to the leather piece. These 3 points can be used to define a coordinate system attached to the hand, which defines its orientation as explained later in this section.

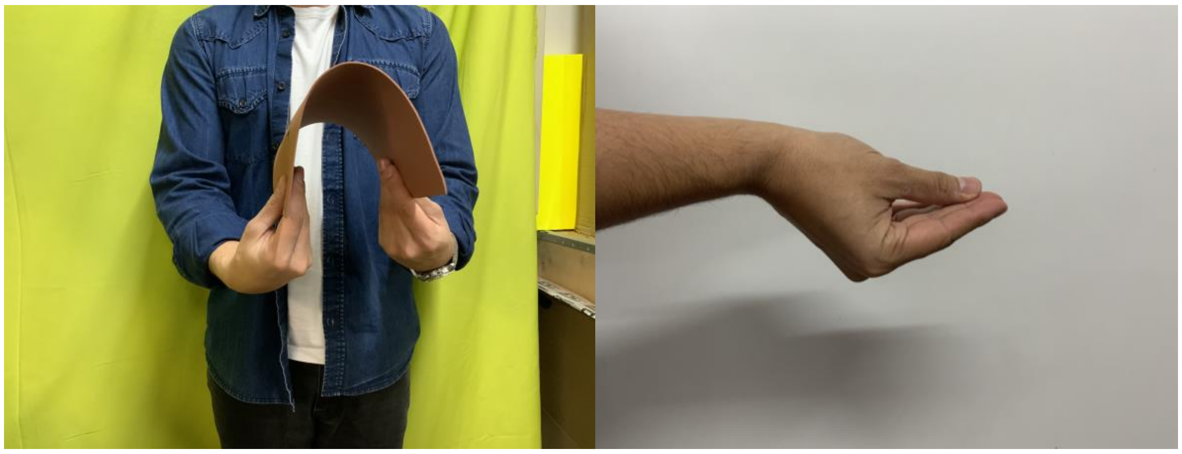


Figure 37 - Hand posture used to manipulate the leather piece.

Before choosing which points from the hands to use to define position and orientation of the hand, one must find the position of all keypoints from both hands as shown in the flowchart from Figure 38.
The set of images previously recorded are accessed frame by frame in the correct order, the RGB ones are fed to Openpose which outputs the 2D coordinates of every hand keypoint (for both hands) in matrix form. Getting the coordinates for a specific point is just a matter of accessing the matrix *datum.handKeypoints* with the right index according to the labels provided in Figure 27 a).

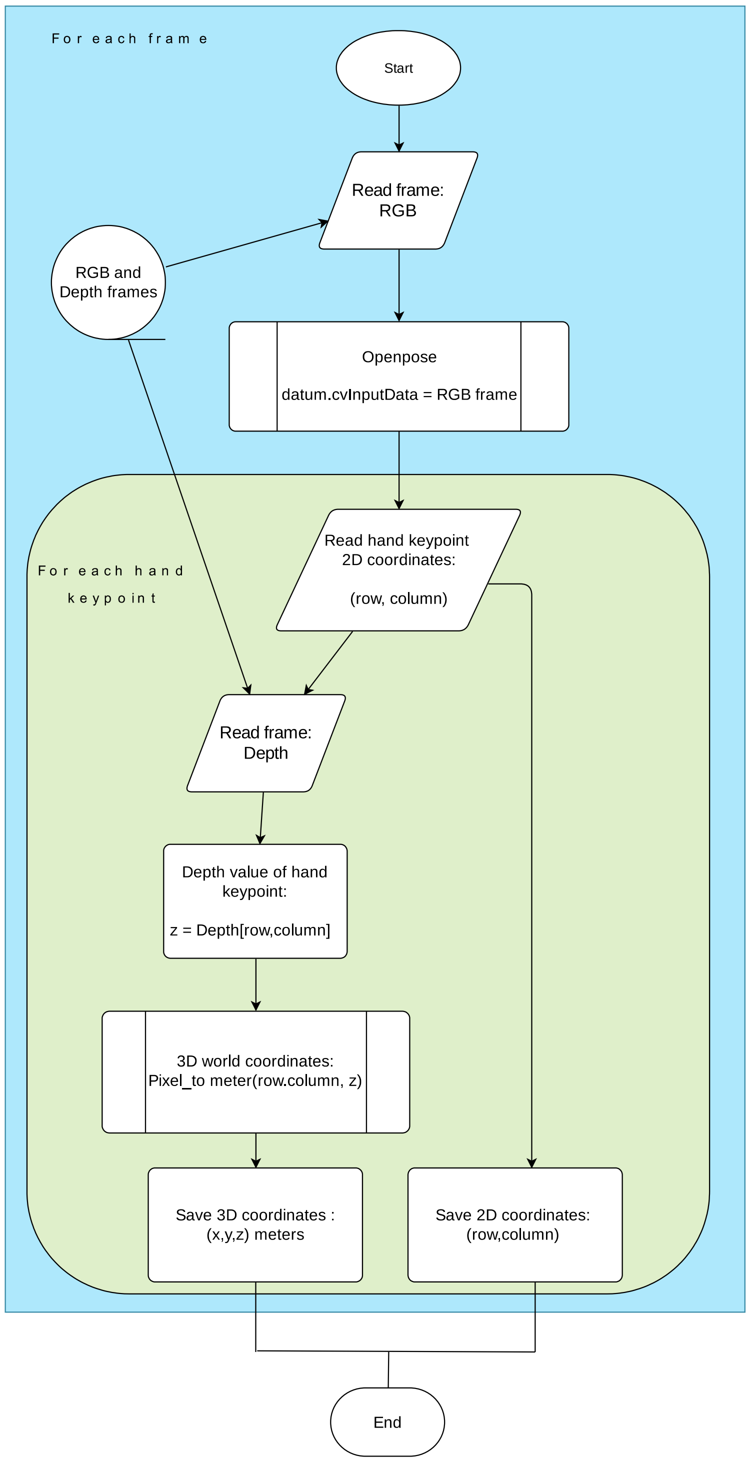

For each frame
Start
Read frame: RGB
RGB and Depth frames
Openpose
datum.cvInputData = RGB frame
For each hand keypoint
Read hand keypoint 2D coordinates:
(row, column)
Read frame: Depth
Depth value of hand keypoint:
z = Depth[row,column]
3D world coordinates:
Pixel_to meter(row.column, z)
Save 3D coordinates : (x,y,z) meters
Save 2D coordinates: (row,column)
End

Figure 38- Flowchart to get the trajectory of every point in both hands.

The 2D coordinates provided for each keypoint are relative to the image reference system (row, and column location). For the remainder of this dissertation every time 2D coordinates are mentioned, one is referring to the coordinates within the frame. On the other hand, every time 3D coordinates are mentioned one is referring to 3D world coordinates in meters.

After getting the 2D location of the hand keypoint, this position is used in the depth map to extract the corresponding depth value. Once this information is collected, developed function *pixel_to_meter()* uses it to calculate the point's 3D coordinates in meters. For every frame, the 2D and 3D coordinates of each hand keypoint are stored in different *.npy* files and folders. For example, in the folder created for point 0 (label from Figure 27-a)), the corresponding 2D and 3D position in both right and left hand are stored.

The function ***pixel_to_meter()*** implements equation (3.3) and can output for every hand keypoint the corresponding 3D world coordinates based on its location in the image and depth value. All the variables to solve Equation (3.3) are known:

$$p_x = column$$
$$p_y = (512 - row)$$
$$P_z = depth\ value$$
$$b_x = 255{,}89$$
$$b_y = 201{,}82$$
$$f = 364{,}26\ .$$

While p_x, p_y and P_z are input values provided for each keypoint, namely column, row, and depth value, b_x, b_y and f are physical parameters of the camera provided by the manufacturer. To note that p_y is not directly the row value because the image coordinate coordinate system of reference is in the top left corner of the image with the y axis pointing along the image height, which is different from what is considered in Figure 23.

Both depth information and location of the keypoints within the frame are extremely important to accurately estimate their true position in the world. In the case of 2D location, its accuracy depends on the capabilities of Openpose to estimate the hand pose from the RGB images provided. Regarding the depth information, the mechanism used by the Kinect v2 to measure those depth values, as explained in section 3.2.2, is only capable of providing measurements to points that the camera can "see" from its point of view. This means that it is impossible to get depth values in parts of the hand that get occluded whether by itself or other objects during manipulation, which can lead to poor information when trying to extract the 3D world coordinates for the hand.

With Openpose this depth ambiguity problem is even more relevant since the estimates for the hand pose are given regardless to what parts of the hand are occluded, so some coordinates given in the RGB frame for hand keypoints might not correspond to points seen by the camera. When trying to read depth values for those estimated points what one will get is the depth measured to the parts occluding the hand and not the point of interest. For example, in Figure 39 is possible to see the case where, even if Openpose accurately detects all the hand keypoints, the depth value measured at the 2D location of point 8 will not be right since the leather piece is partially occluding the left hand's fingers.

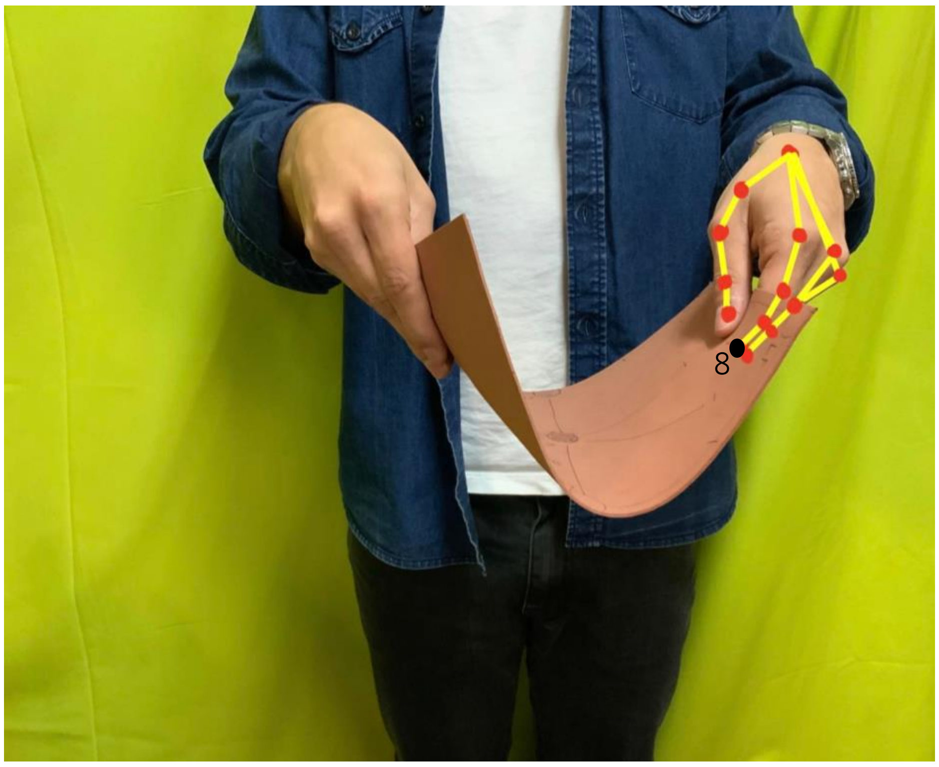


Figure 39 - Even with the detection of the hand skeleton in 2D, obtaining the depth value for point 8 would be impossible considering the occlusions. The depth value for that point would be measured to the surface of the leather piece.

In the case of tracking position if the point is chosen correctly this problem might be attenuated. Point 5 (Figure 40) was selected to represent the hand's overall position over time. This point is almost always visible to the camera during manipulation of the leather pieces under the conditions already mentioned, besides that it is also a point that usually stays relatively close to the leather piece which is useful to later mimic the process with the robot gripper.

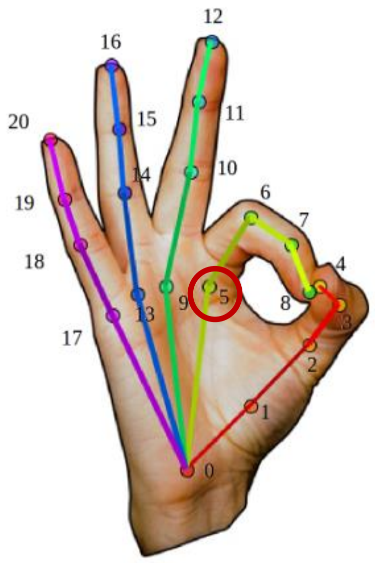


Figure 40 - Hand keypoints location and corresponding label.

In the case of orientation, the selection of three points is more complex since it is more difficult to walk around the depth ambiguity problem. To better understand this, is important to define how to describe the orientation of the hand. For that, one needs to find the orientation matrix in equation (3.4) derived in section 36 based on the location of the points selected to define the orientation. It is with the coordinates of these points that one can build the coordinate system attached to the hand. To do this let's consider three hand keypoints(P_5, P_8 and P_{17}) whose coordinates are

$$\begin{aligned} P_5 &= (x_5, y_5, z_5)^T \\ P_8 &= (x_8, y_8, z_8)^T \\ P_{17} &= (x_{17}, y_{17}, z_{17})^T. \end{aligned} \tag{4.1}$$

These points together form a plane that will be the starting point to derive the coordinate system representing the hand's orientation. As it can be seen in Figure 41, two vectors connecting these points ($\overrightarrow{v_{5,17}}$ and $\overrightarrow{v_{5,8}}$) can be found,

$$\begin{aligned} \overrightarrow{v_{5,17}} &= P_{17} - P_5 = (x_{17}, y_{17}, z_{17})^T - (x_5, y_5, z_5)^T \\ \overrightarrow{v_{5,8}} &= P_8 - P_5 = (x_8, y_8, z_8)^T - (x_5, y_5, z_5)^T \end{aligned} \tag{4.2}$$

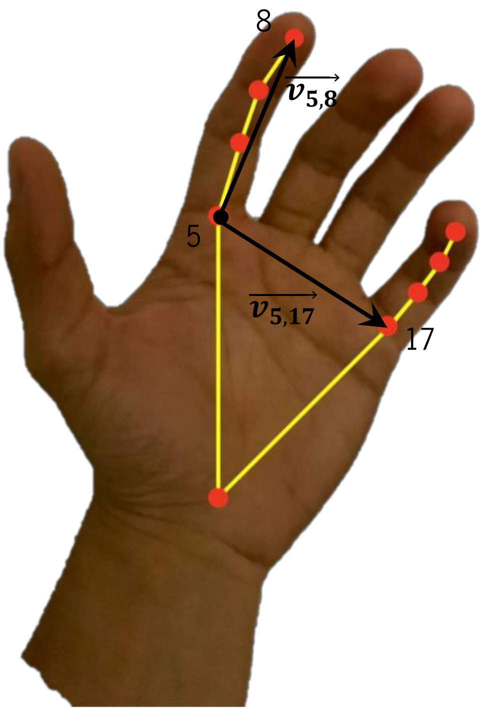


Figure 41- Representation of 3D vectors, $\overrightarrow{v_{5,17}}$ and $\overrightarrow{v_{5,8}}$, that can be used to define the hand's orienation.

Once these vectors are found, a vector normal to this plane can be calculated with their corresponding cross product,

$$\overrightarrow{v_{5,17}} \times \overrightarrow{v_{5,8}} = \vec{x}. \tag{4.3}$$

The calculated vector can be considered as the director vector for the x axis as shown in Figure 42.

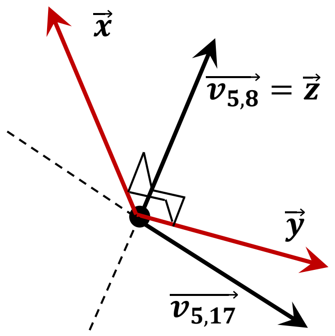

$\vec{x}$
$\overrightarrow{v_{5,8}} = \vec{z}$
$\vec{y}$
$\overrightarrow{v_{5,17}}$

Figure 42-Representation of the coordinate system ($\vec{x}$, $\vec{y}$ and $\vec{z}$) derived for the hand's orientation.

To get the y axis, the cross product can be used once again but now one of the two vectors initially used must be selected as the director vector for the z axis and only then y can be calculated,

$$\overrightarrow{v_{5,8}} = \vec{z}. \tag{4.4}$$

$$\vec{z} \times \vec{x} = \vec{y}.$$

Finally, all the director vectors calculated must be converted to unit vectors before building the orientation matrix,

$$\vec{z} = \frac{\vec{z}}{||\vec{z}||}$$
$$\vec{y} = \frac{\vec{y}}{||\vec{y}||} \qquad (4.5)$$
$$\vec{x} = \frac{\vec{x}}{||\vec{x}||}$$

The procedure is the same for every set of three hand keypoints considered.

This approach to measure the hand's orientation is heavily dependent on the quality of the data acquired for the hand keypoints (equation (4.1)), so if poor information is used in this stage the corresponding orientation results will also be unsatisfactory. This effect is aggravated because even small variations in the position of the points can represent big variations in the orientation calculated since we are mapping from positions to rotation angles.

Considering the high sensibility of this calculations to noise and poor estimates of the points location (2D coordinates and depth), the depth ambiguity problem assumes great relevance since this can be a big source of variations in the data. Therefore, it becomes very difficult to trust the measured depth values for every point in the hand when calculating its orientation. For this reason, and to eliminate the dependency of the previous method on the measurements of depth for each hand keypoint, a new method called "ITP - iterative pose estimation" is proposed.

4.3.1 ITP - iterative pose estimation

In section 2.3 are presented multiple solutions where an iterative approach to solve the hand pose task is used. For example, model-based solutions ([28], [34], [52], [67]) are inherently iterative considering the fact they usually start by generating a set of hypothesis they evaluate against real data until convergence. These approaches also take advantage of previous knowledge about the hand structure and motion restrictions to generate the hypothesis in the first place. This way it eliminates unrealistic options for the hand pose. However, these are not the only methods to consider an iterative approach along with previous knowledge about the hand, to estimate its pose. Some appearance-based solutions also take advantage of this to improve the accuracy of their CNNs in the hand pose estimation task which is especially useful in monocular setups heavily prone to occlusions on the hand. For example, [68] and

[47] add to their base networks, IK algorithms that use the features extracted from the hands to either refine or estimate the hand keypoints position iteratively. [37] also presents an architecture where kinematic restrictions are introduced at the end of a CNN to recover the hand pose.

Considering the problems related to occlusion of the hands derived by the use of only one camera in our system, it was developed an iterative method that estimates the 3D pose of the hand only based on the 3D world coordinates of the wrist and the 2D locations of the remaining hand keypoints.

The approaches presented above introduce their IK algorithms directly in their model-base or appearance-base methods. However, in the case of this dissertation that level of complexity was not necessary. Therefore, a simpler solution is proposed, which we refer to as iterative pose estimation method.

This solution is based on the following observation: for a particular 2D pose estimated by Openpose there is only a set of depth values for the hand keypoints that accurately define the hand's 3D pose considering the anatomical restrictions imposed and the initial position of the hand's wrist. Thus, instead of needing to know accurately the depth values for every keypoint one can just use the wrist information to derive the data for the remaining points through pseudo-inverse kinematics techniques. The use of inverse Kinematics techniques in hand tracking methods is not new as shown in the previous examples. However, according to the research presented, this is the first approach that tries to reconstruct a 3D pose of the hand from a 2D hand pose prediction from a discriminative method with a simple pseudo-inverse kinematics approach.

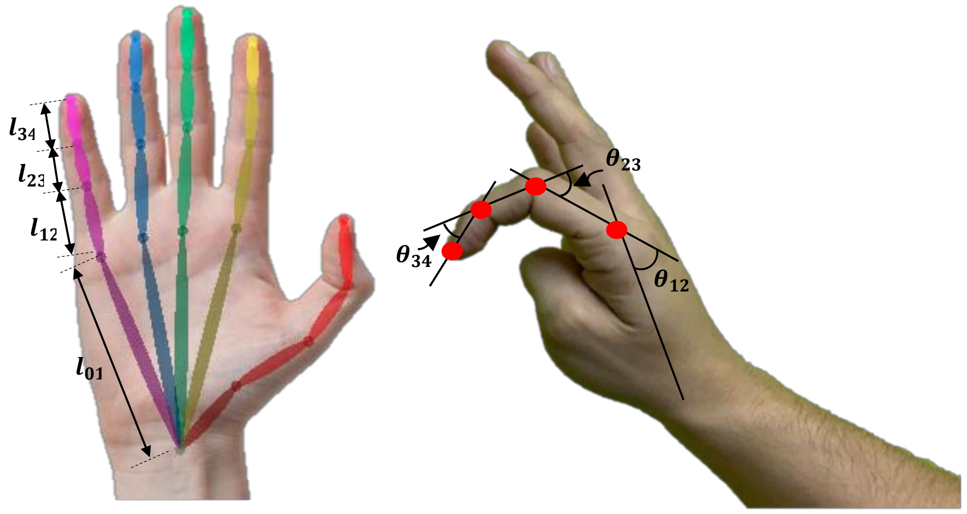


Figure 43 - Example of anatomical restrictions considered for the hand model: length restrictions for the hand links of the little finger (left); angle restrictions for the joint angles of the Index finger(right).

The main goal of this method is to iteratively derive the set of depth values from the wrist to the fingertips that can build the 3D configuration that best matches the 2D configuration of the hand estimated by

Openpose in each frame. For that it is considered the simplified hand model used by Openpose (Figure 40) and the corresponding anatomical restrictions of the hand such as link lengths and joint angles (Figure 43).

From this point on, when coordinates are mentioned with the x, y or z they refer to 3D world coordinates. For each frame recorded, both hands are processed with this method. Starting from the corresponding wrist position (Point 0) whose 3D world coordinates are assumed to be correct, each finger is reconstructed with the same procedure. Two consecutive finger keypoints are considered (Point 1 = $[x_1, y_1, z_1]^T$ and Point 2 = $[row, column, z_2]^T$), the first one is assumed to have known 3D coordinates (starts at the wrist) while for the second one only the location within the image is known (row and column). The next step is to find the depth value for Point 2 that locates the hand joint in the correct position according to the hand model's restrictions. With this in mind, a set of possible depth values for Point 2 must be evaluated. The maximum possible distance between the two points in any circumstance is their corresponding link length defined initially, thus it is reasonable to consider the possible depth values ranging from $(z_1 - l_{12})$ to $(z_1 + l_{12})$ with l_{12} being the link length. With all possible depth values defined, one can iterate over them in order to generate a set of possible 3D world locations for Point 2 with the help of the *pixel_to_meter()* function. These estimated 3D world coordinates for Point 2 are used to calculate the vector $\overrightarrow{v_{12}}$ connecting it to Point 1,

$$\overrightarrow{v_{12}} = link_{12} = Estimated\ Point2 - \ Point1 = [x_2, y_2, z_2]^T - [x_1, y_1, z_1]^T. \tag{4.6}$$

In each iteration the length of the calculated vector $\overrightarrow{v_{12}}$ is evaluated against the predefined link length of the hand model and only the estimated position that yields the lowest error is taken as the correct one.

To make the process more robust, the angle between two consecutive finger links (equivalente to joint angle) is also computed and evaluated against the predefined motion range for the corresponding joint. Figure 44 represents the estimation procedure mentioned above, but for the case of thumb's Point 1 and 2. The Iterative pose estimation method used in this finger would start by estimating Point 1 from the wrist (Point 0) as explained before, however these steps were omitted to simplify the explanation and because the estimation procedure is the same for every 2 consecutive points. Once the 3D world coordinates of Point 1 are estimated, is possible to estimate those for the following point (Point 2). The only thing that is known about Point 2 is its 2D image location, however one needs to find the optimum depth value that locates Point 2 in its right 3D world position. For that, the iterative pose estimation method creates a set of possible depth values with an increment of 5 mm for Point 2 ranging from $(z_1 - l_{12})$ to $(z_1 + l_{12})$ with l_{12} being the corresponding link length (Figure 43) and z_1 the depth value estimated for Point 1. Then, it runs the function *pixel_to_meter()* for every depth value considered which will yield the set of all possible 3D locations of Point 2 (set of red dots in Figure 44). In order to select the right location for Point 2, vector $\overrightarrow{v_{12}}$ is computed and evaluated against length and angle restrictions imposed. Consider in Figure 44 that all possible locations have the index n, only two of them (n=5 and n=10) respect the length restriction for that specific finger link, however only the location where n =10 respects the angle restriction imposed leading to the selection of this location as the right 3D world position for Point 2.

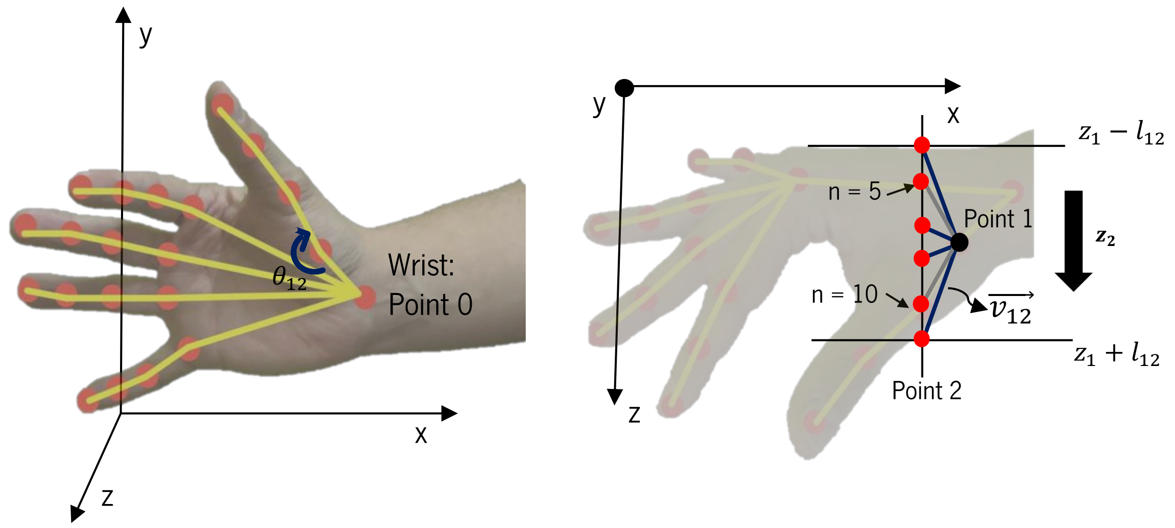


Figure 44 – Iterative pose estimator used to estimate the 3D hand pose for the 2D hand on the left. On the right, representation of the estimation process for Point 2 of the thumb from previous Point 1 = $(x_1, y_1, z_1)^T$. Two solutions (n =5 and n=10) respect the requirement for the link (l_{12}) length but only one solution (n=10) respects both angle (θ_{12}) and length requirements.

To calculate the angle between two consecutive finger links ($link_{ant}$ and $link_{12}$) one can use the dot and cross product as follows,

$$\theta_{12} = \operatorname{atan2}\left(\left\|\frac{link_{12}}{\|link_{12}\|} \times \frac{link_{ant}}{\|link_{ant}\|}\right\|, \frac{link_{12}}{\|link_{12}\|} \cdot \frac{link_{ant}}{\|link_{ant}\|}\right). \tag{4.7}$$

With,

$$\sin\theta_{12} = \left\|\frac{link_{12}}{\|link_{12}\|} \times \frac{link_{ant}}{\|link_{ant}\|}\right\|, \text{ and } \cos\theta_{12} = \frac{link_{12}}{\|link_{12}\|} \cdot \frac{link_{ant}}{\|link_{ant}\|} \tag{4.8}$$

Since the angle calculations depend on the previous link estimated, for the case of the link between the wrist and first joint of each finger this evaluation is not made and only the link length is considered. To also restrict this joint one could consider the forearm position and orientation (also given by Openpose). The range of motion of the joints (equation (4.9)) was defined taking into consideration the grasp pose considered in Figure 37 and it was equal for all fingers. Considering the joint angles in Figure 43,

$$\begin{gathered} 0° < \theta_{12} < 90° \\ -10° < \theta_{23} < 10° \\ -10° < \theta_{34} < 10° \end{gathered} \tag{4.9}$$

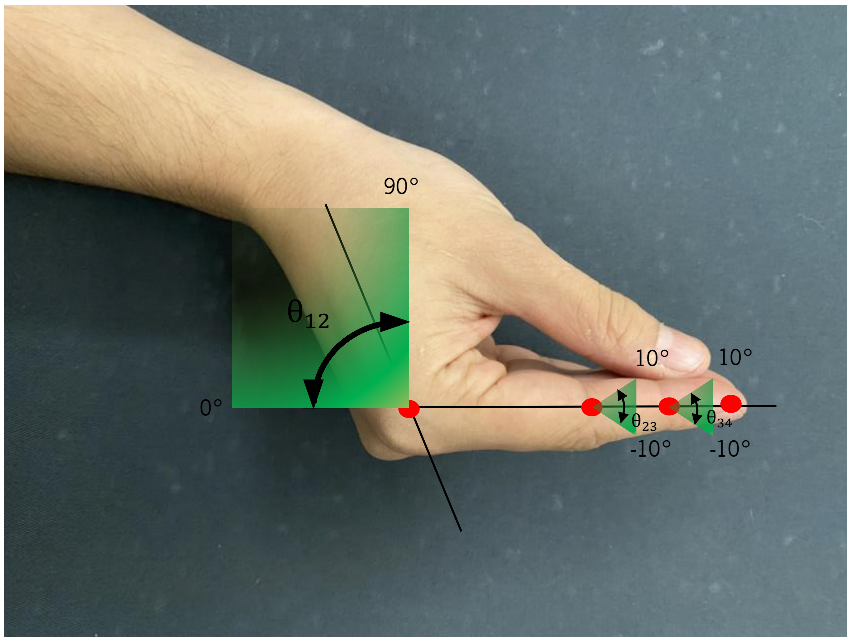


Figure 45 – Ilustration of the motion restrictions for every finger joint considered in the model. In this case, the representation is made in the index finger. (note: angle representations are not accurate. Just to help the reader understand the idea.)

In all cases a margin of error was considered to deal with possible uncertainties from the data.

The pseudo-code for this method is also presented in Figure 46.

```
For each frame:
    $Point\_0 = Wrist = [x_0, y_0, z_0]$
    For each finger:
        For two consecutive points (1 and 2) starting
        from the wrist up to the fingertip:
            $Point\_1 = [x_1, y_1, z_1]^T$
            For $(z_1 - l_{12} < z_2 < z_1 + l_{12})$:
                $Point\_2 = [row_2, column_2, z_2]^T$
                $Estimated\_Point\_2 = [x_2, y_2, z_2]^T = pixel\_to\_meter(Point\_2)$
                $link_{12} = Estimated\_Point\_2 - Point\_1$
                $\theta_{12} = atan2\left(\left\|\frac{link_{12}}{\|link_{12}\|} \times \frac{link_{ant}}{\|link_{ant}\|}\right\|, \frac{link_{12}}{\|link_{12}\|} \cdot \frac{link_{ant}}{\|link_{ant}\|}\right)$
                If $\theta_{12}$ lies inside the allowed range and $\|link_{12}\| = l_{12}$:
                    $Point\_2 = Estimated\_Point\_2$
```

Note: For the estimation of the first point of each finger, angle restrictions are not evaluated since it is not considered information about the forearm.

Figure 46- Pseudo code of the Iterative Pose Estimator

Even though this method heavily reduces the depth ambiguity problem for most of the points on the hand still requires that the wrist is never occluded during the manipulation movements. Besides, it is also assumed that the 2D configuration given by Openpose is correct.
The iterative pose estimation method only needs to compute the position of the fingers where the selected points for the orientation belong. This means that, if for example point 5, 8 and 17 (Figure 40) are chosen to define the hand's orientation according to the method already described, only the index and little finger are estimated and not the whole hand. In Figure 47 is possible to see an example of the output of this method when applied to the right hand during the manipulation of the leather piece. One can see that as expected a 3D skeleton of the hand is built.

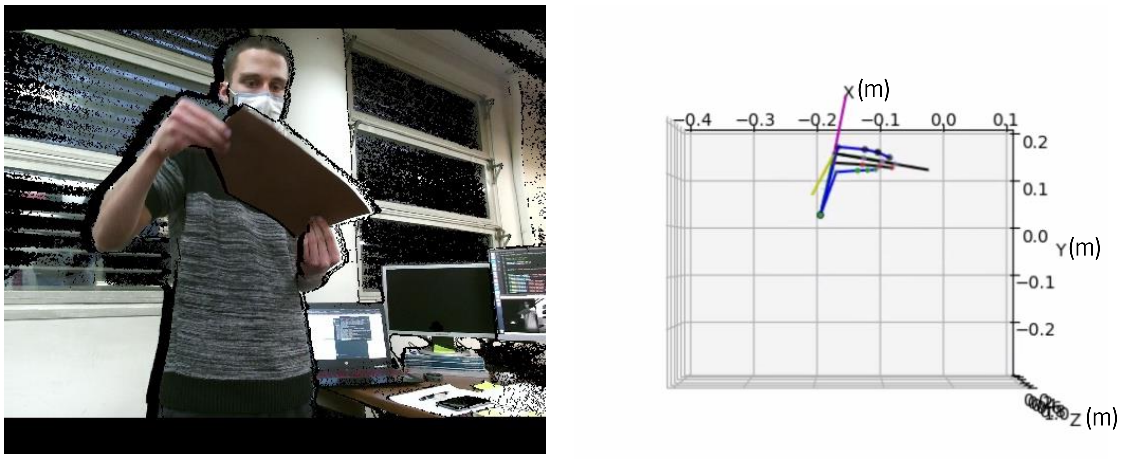


Figure 47 - Output of the iterative pose estimation method: estimating the 3D pose of 4 fingers of the hand using only 3D location of the wrist and the 2D configuration of the hand given by Openpose and corresponding orientation represented by a coordinate system attached to the hand.

The final step to get the position and orientation for the manipulator to inspect the leather pieces is to filter the noise from the data collected. Like any type of measurement, the position of the joints extracted by our system comes with noise that must be removed in order to smooth out the trajectories and the variations in orientation. There are two main types of digital filters: finite impulse response (FIR) and infinite impulse response (IIR). IIR filters, as the name suggests, have an infinite impulse response that is the result of their recursive nature. While a FIR filter only bases its output on the input signal, an IIR filter bases its output on former output values as well.
Nowadays, many available libraries allow an easy implementation of the IIR digital low pass filter. As it is the case of Scipy, which provides two functions, *signal.butter()* [6]and *signal.lfilter()*[7], that compute the coefficients of equation (3.10) and filter the input signal with equation (3.11), respectively. The input for

[6] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html
[7] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.lfilter.html#scipy.signal.lfilter

these functions is the sampling frequency of the input signal -15Hz - which is the available framerate, filter order – n= 6, and the cut-off frequency - ω_c =1Hz. These values were tuned by comparison with ground truth values acquired with a motion capture commercial software (Vicon system[8]). Figure 48 shows the x coordinates of an inspection trajectory acquired simultaneously by our method and the Vicon system, and it is noticeable the presence of noise in our data when compared to the one provided by the commercial software that already implements filtering algorithms to their measurements. The Vicon data can be used as reference to identify which parts of the trajectory are meaningful to represent the inspection movement. By identifying those features one can better design the filter to smooth trajectory without losing those main characteristics.

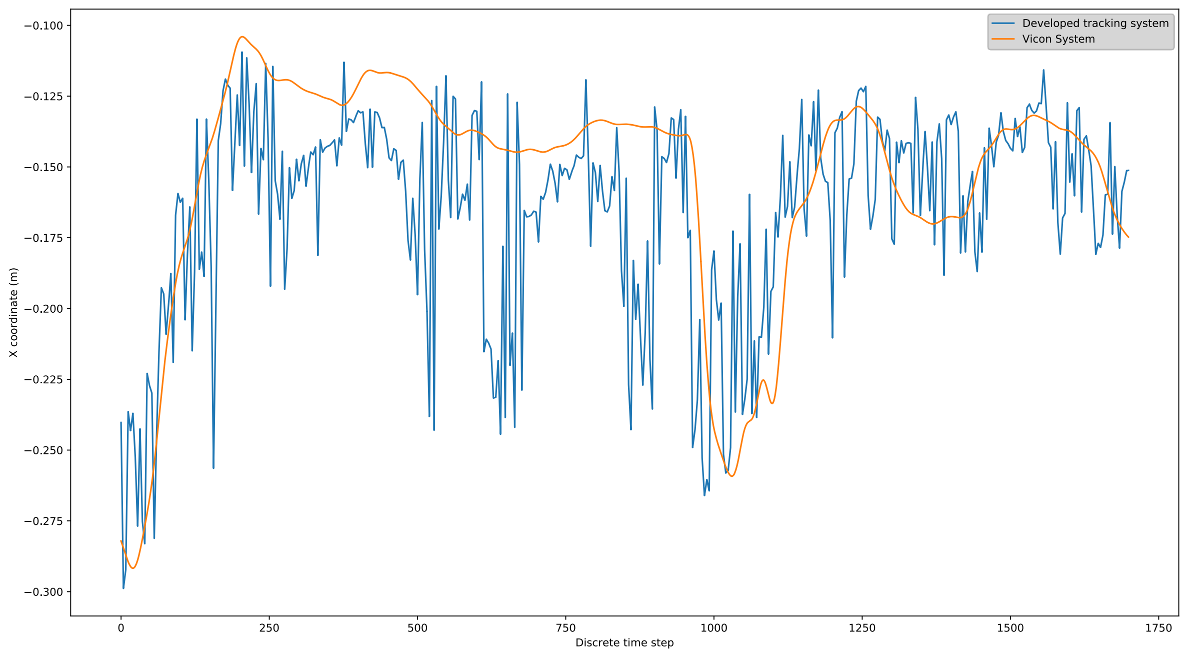


Figure 48 -Example of the x values of an inspection trajectory acquired by the developed system (blue line) and acquired by the commercial software Vicon system provided by CCG (Centro de Computação Gráfica).

To design the best fitting filter, the two main filter parameters were evaluated to see which values would present the best trade-off between smoothing the trajectory as much as possible but without losing the meaningful movements of the trajectory. In Figure 49 is possible to see the impact of the low pass filter's order in the output filtered signal.

[8] https://www.vicon.com/

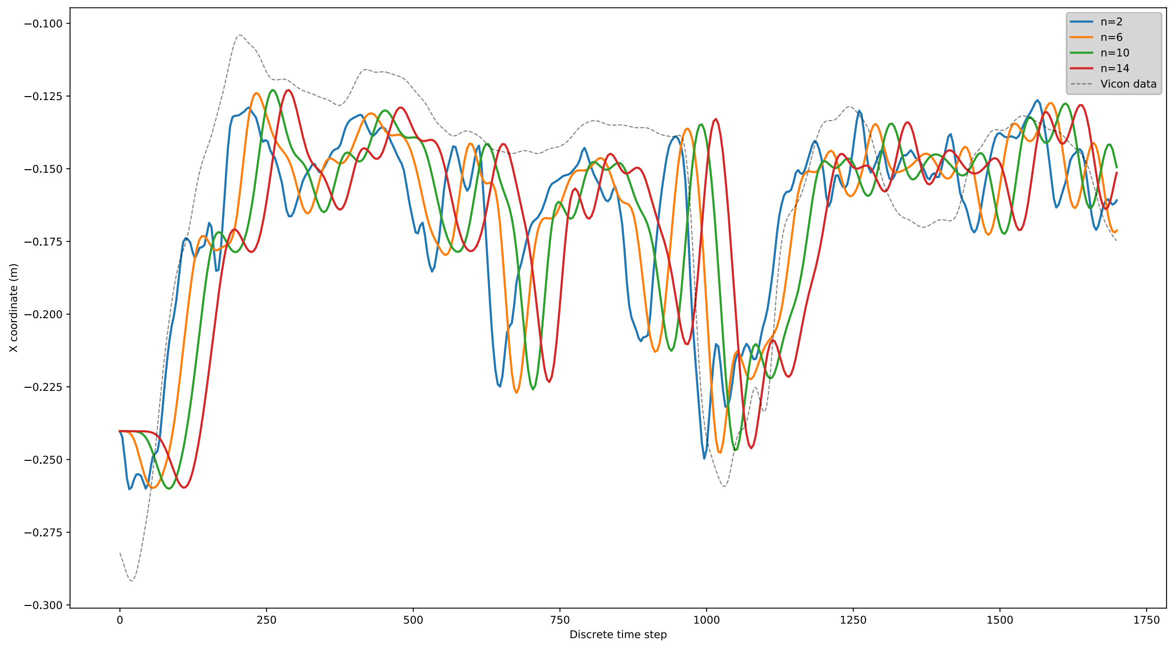


Figure 49- Effect of the low pass filter's order in the filtered signal with ω_c = 1.

As expected, as the order increases the better the filter can eliminate the “movements” at higher frequencies corresponding to meaningless noise information. However, the phase shift between the original and filtered signal also increases. At low orders, the filter is less discriminative, and can’t distinguish as precisely between what’s noise and what’s not. Considering this the value of n = 6 was selected since it’s low enough to guarantee low phase shift but without being too permissive to high frequency noise. After selecting the filter’s order, the cut-off frequency was also evaluated for different values to understand which could be the best solution for this problem. In Figure 50 it is possible to see the filtered trajectories for four different cut-off frequencies which range slightly below typical values used in motion analysis (around 10 Hz) according to [57]. However, this is the range that better suited our experiments.

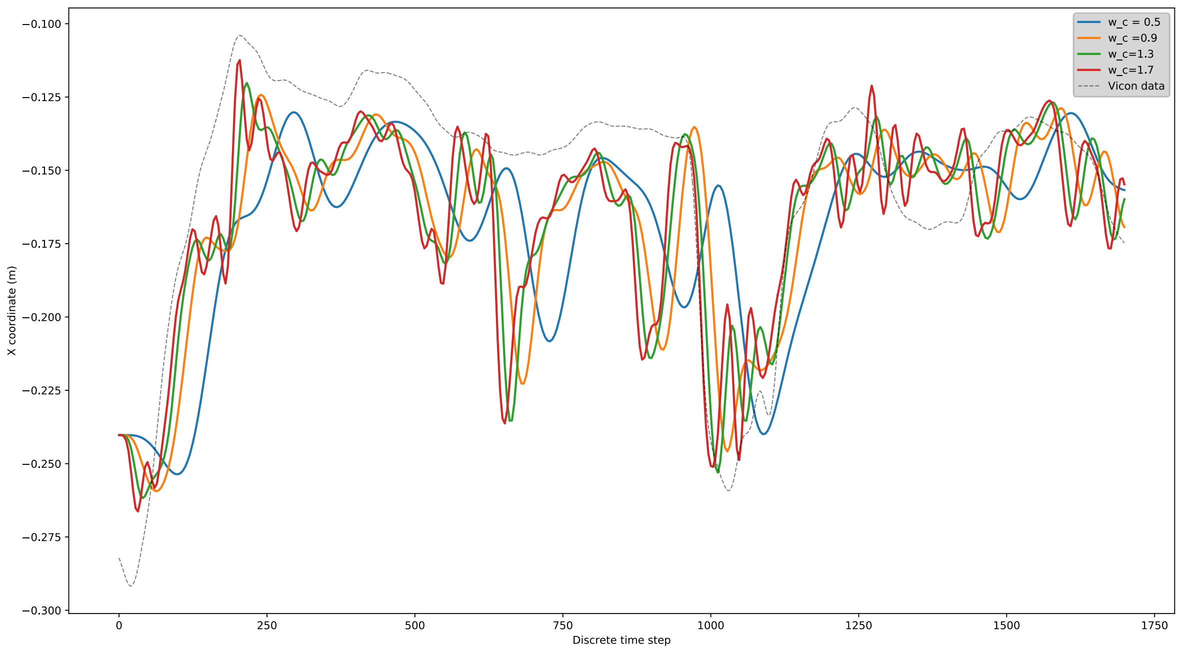


Figure 50 -Effect of the cut-off frequency in the filtered signal with n = 6.

The main effect of the cut-off frequency in the filtered signal is the amount of information one wants to eliminate. For extremely small cut-off frequencies ($\omega_c = 0.5\ Hz$) the filtered signal loses track of the meaningful trajectory. However big values ($\omega_c > 1.7\ Hz$) let noise pass through, for this reason a cut off frequency of 1 Hz was selected. These values were the ones that provided a better commitment between smoothing enough the trajectories and not losing important information about the movement.

5. Results

After developing the complete pipeline, one needs to understand if it is capable of achieving the proposed goals. As explained before, the pipeline must be capable of recording the inspection movements and extract the trajectories (position and orientation) of the hands to then replicate them with a manipulator. However, is important to mention that even though we are looking for trajectories, the real goal of this work is to facilitate the inspection of deformable objects, namely leather pieces, using automatic inspection (manipulators). Therefore, more than accurate absolute positions and orientations, this solution must be capable of capturing the crucial parts of the movements that are responsible for deforming the leather pieces as needed in order to detect the defects. With this in mind, it is necessary to get ground truth trajectories to then compare them with the ones from our solution and check if they are similar enough. The following sections present the system used to acquire ground truth trajectories and the results obtained for both position and orientation of the hand. Tests with a real manipulator are also presented.

5.1 Vicon system

To evaluate the output of the developed solution the Vicon optical motion tracking system was used. This is a high-performance motion tracking solution that uses IR cameras to track the 3D position (x, y, z) of a set of markers through triangulation procedures with an uncertainty for the measurements below 1mm. Since it uses IR signals it is recommended that there aren't any specular or shining surfaces in the volume to be covered by the cameras. Each marker can be tracked individually but for better accuracy and robustness of the results it is useful if they are integrated in a predefined kinematic model.
Ideally the system would track the same points of the hand estimated by our system. However, due to the size of the hand it would be impossible to place that many markers on it. Besides, the software for the vicon system used doesn't have a predefined kinematic model just for the hands. Therefore, it was only possible to use their upper body kinematic model that assumes the corresponding position for the markers as depicted in Figure 51.

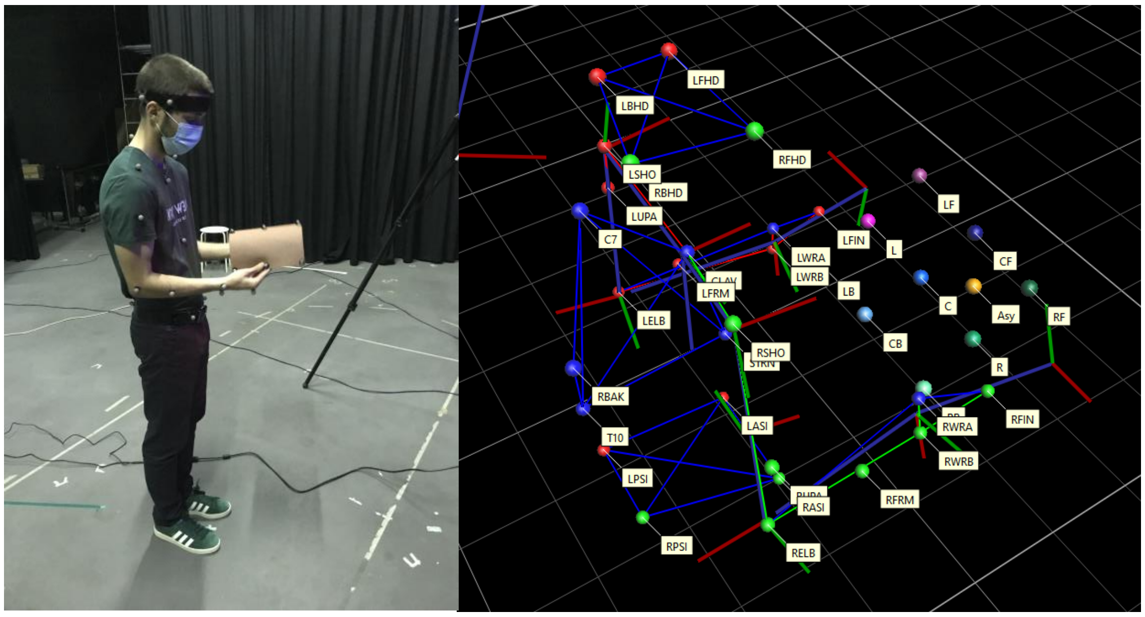


Figure 51 -Position of the markers on the user's upper body and leather piece (left). Corresponding kinematic upper body model constructed with information about the location of each marker (right).

The predefined position for each marker allows the software to build the corresponding kinematic model of Figure 51. The restrictions associated with this model help the software deal with occlusions and missed detections that can occur during the recordings.

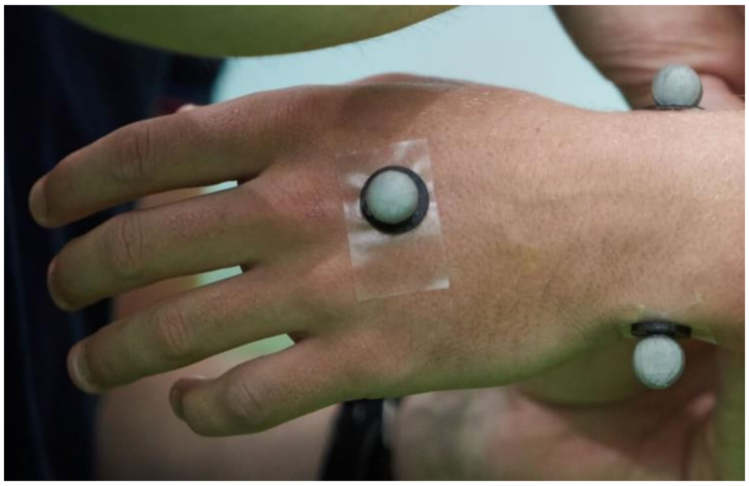


Figure 52 – Position of the markers placed on the hands

Even though there are multiple markers placed on the body, there is only interest on the ones placed on each hand (Figure 52). Since we are only considering three markers on the back of the hand further evaluations must consider the hand keypoints estimated by our system closest to these locations.

In order to get more information out of these tests, data from the leather piece surface was also acquired by placing a set of markers as in Figure 53. The results for these markers were less robust since the markers were not considered within a specific kinematic model for the leather piece. However, everything was corrected with a simple post-processing routine where the miss-detected markers were reorganized manually.

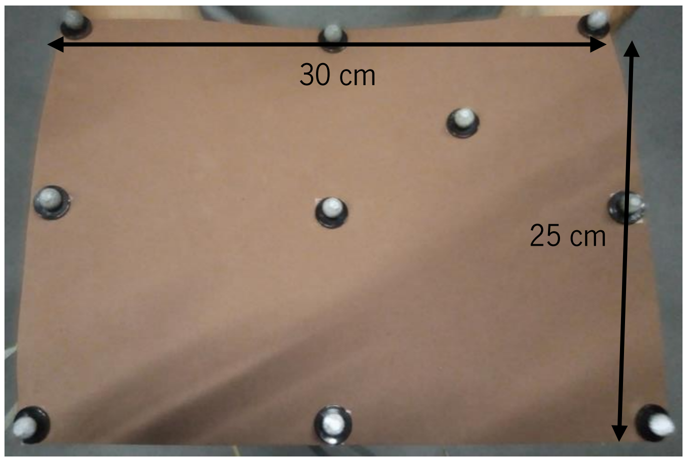


Figure 53 – Markers placed on the leather piece surface and its respective width and length

Information about the leather piece deformation could be useful to help complement and correct the information that comes from our system generating reliable information for the robot. For example, the orientation of the leather piece's surface in the point of contact with the hand might help correcting the orientation estimated by our solution.

The results from the Vicon system are the ground truth for further comparisons with results from the developed solution, and for that reason it is also necessary to guarantee that the inspection movements recorded by the motion tracking software are also recorded with Kinect.

The complete setup is presented in Figure 54. The Kinect sensor had to be placed away from the volunteer to avoid interfering with the cameras from the Vicon system. This was not ideal since it makes the task for our system more difficult.

Finally, all the data provided by the Vicon system was recorded at 60Hz which required a posterior matching procedure to align it with the data recorded by our system for comparison.

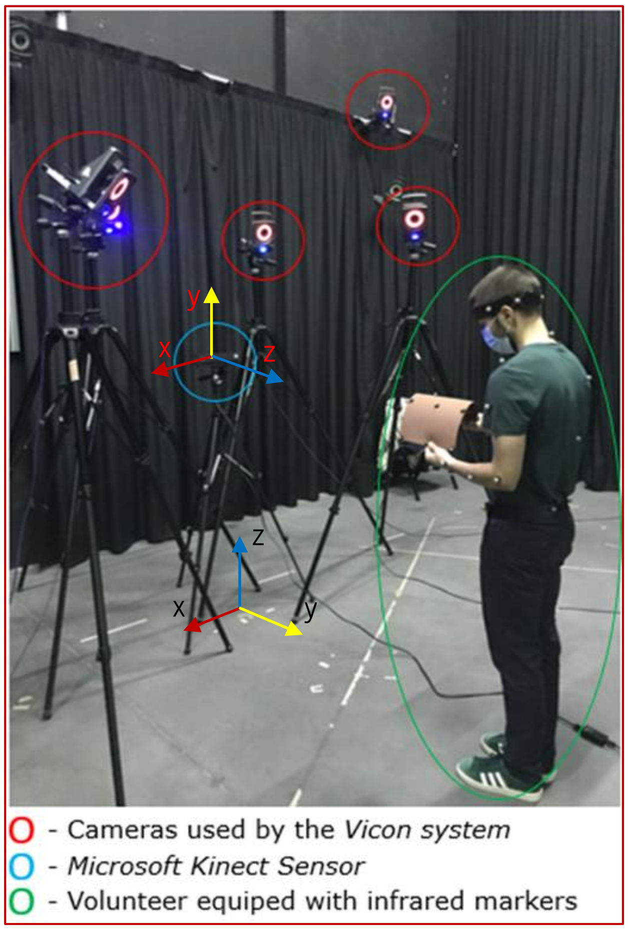

y
x
z
z
x
y
O - Cameras used by the *Vicon system*
O - *Microsoft Kinect Sensor*
O - Volunteer equiped with infrared markers

Figure 54- Complete test setup with Vicon cameras and Microsoft Kinect sensor. With the location of the reference coordinate systems of the Kinect camera and the Vicon system.

Before starting the test, it is necessary to calibrate the Vicon system by sweeping the area with the stick presented in Figure 55. Then, the stick must be placed in the location desired for the origin of the reference system, as shown in both Figure 54 and Figure 55-left.

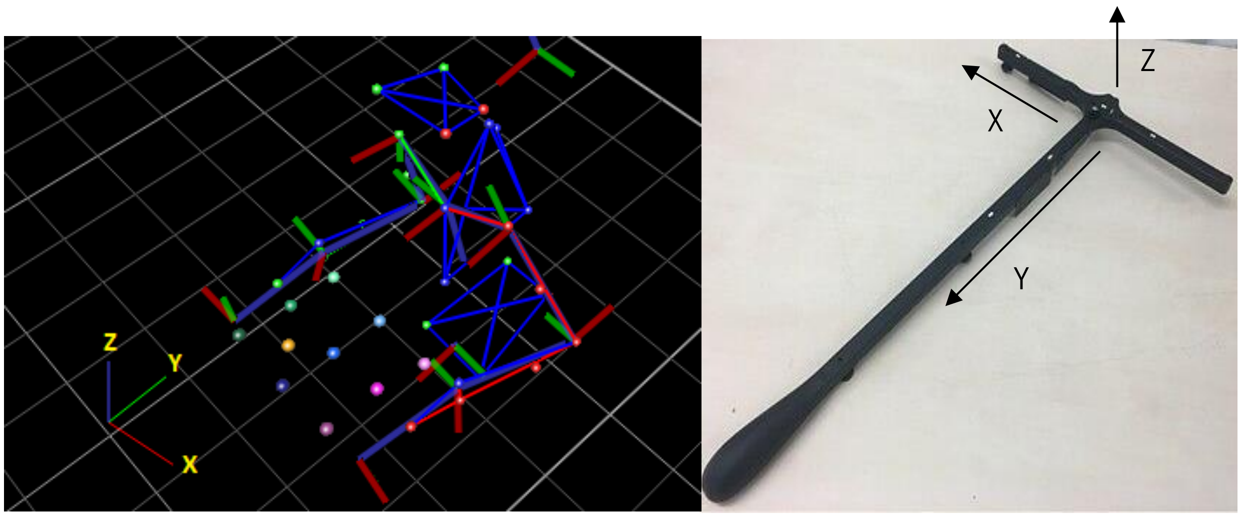
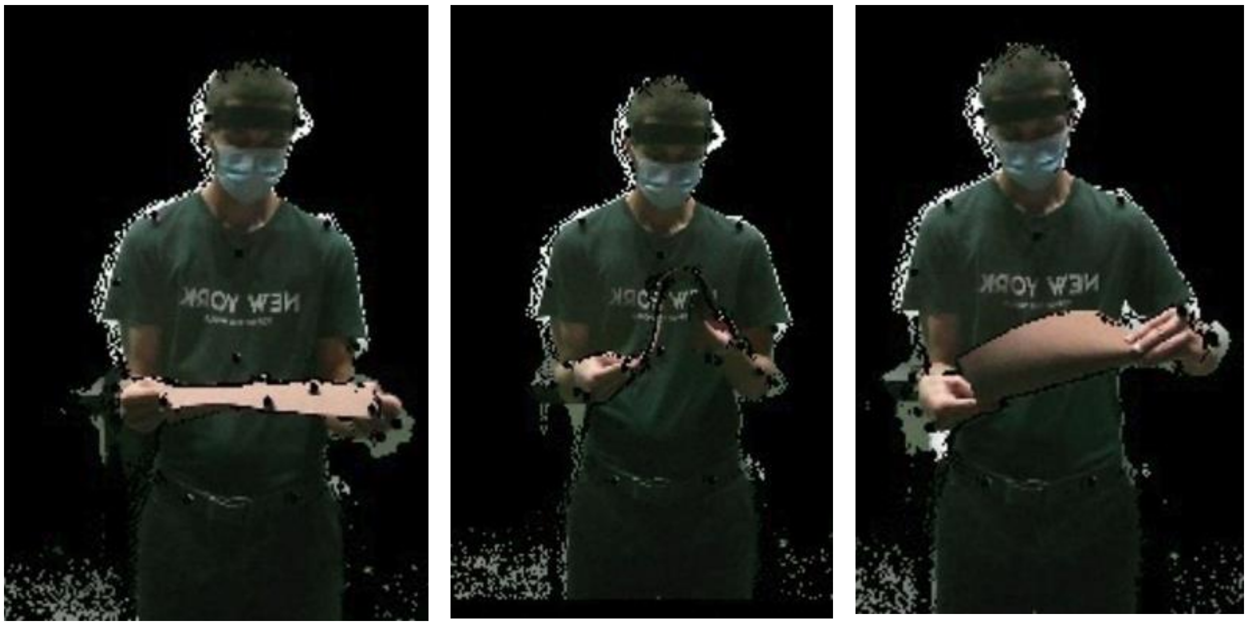


Figure 55- On the left, positon and orientation of the reference coordinate system relative to the volunteer. On the right, marker device used to calibrate the system and define location and orientation of the reference coordinate system.

Once the system was prepared, a set of inspection movements were recorded to later evaluate the corresponding data for position and orientation of the hands. The movements used in this test were selected based on what were considered primary inspection movements (Figure 56). The idea is to understand if our solution works on these primary movements. If it does, it becomes the starting point to later evaluate even more complex trajectories that result from the combination of these basic movements.



a) b) c)

Figure 56 – Three types of inspection movements evaluated: a) Traction; b) Periodic movements; c) Torsion.

These inspection movements are performed with complementary trajectories from both hands, which means that most of the times, in the case of leather pieces, what really matters to reach the desired deformations is the position of one hand relative to the other. This means that one hand can be completely fixed while the other does all the necessary movements. If this is the case, it would allow for a more robust solution since the camera used could focus mainly on the moving hand while the other becomes a fixed support. To verify if in fact the manipulation of the leather piece can be performed with just on moving hand each primary movement was recorded with two moving hands and with just one. The summary of all the movements considered is presented in Table 6.

Table 6- Summary of all the inspection movements considered in the Vicon system tests.

Traction		Periodic Movements (PM)		Torsion	
One hand (1H)	Two hands (2H)	One hand (1H)	Two hands (2H)	One hand (1H)	Two hands (2H)

The information to be evaluated is only about the moving hands, so in the movements where only one hand is moving (the right hand) only information about that hand is considered.

5.2 Position

In these tests the position information for the hand was extracted with point 5 in the case of our hand tracking system (location as in Figure 40), and markers 'LFIN' and 'RFIN' (Figure 57) for the Vicon system depending if it is the left or right hand respectively (Figure 51).

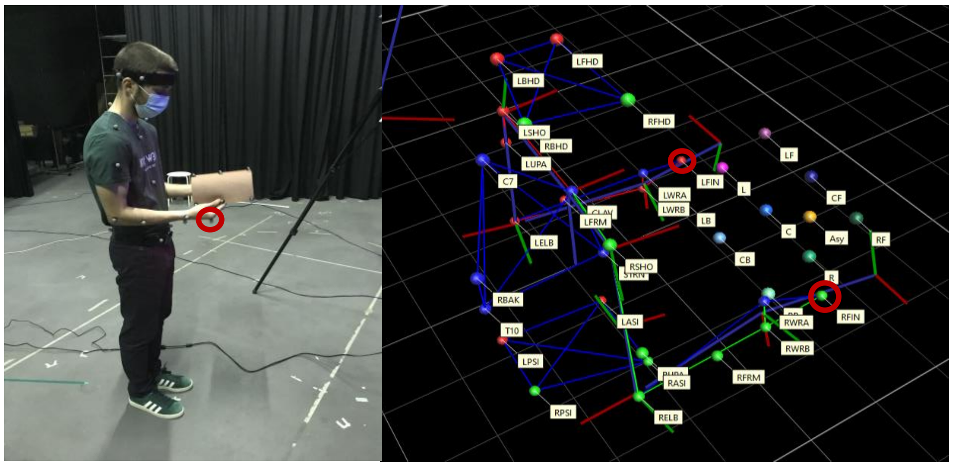


Figure 57 – Location of the marker "RFIN" on the right hand of the user, highlighted in red. Location of the markers "LFIN" and "RFIN" in the kinematic model, highlighted in red.

The main goal of our system is to get the trajectories from the hands that will make the manipulator deform the leather piece at least similarly to the way it was deformed during demonstrations. Therefore, one is more interested in the overall shape of the trajectory rather than the individual accuracy of each measured position.

Two methods were chosen to compare the trajectories from our system to the ones from the Vicon system. The first consists in doing it qualitatively and the second in evaluating their corresponding gradient. The second one consists in computing the mean squared error (MSE) between the gradient of both trajectories in each direction (x, y, z) as depicted in Figure 58. The gradient was calculated with the function *numpy.gradient()*[9] from the Python library Numpy.

[9] https://numpy.org/doc/stable/reference/generated/numpy.gradient.html

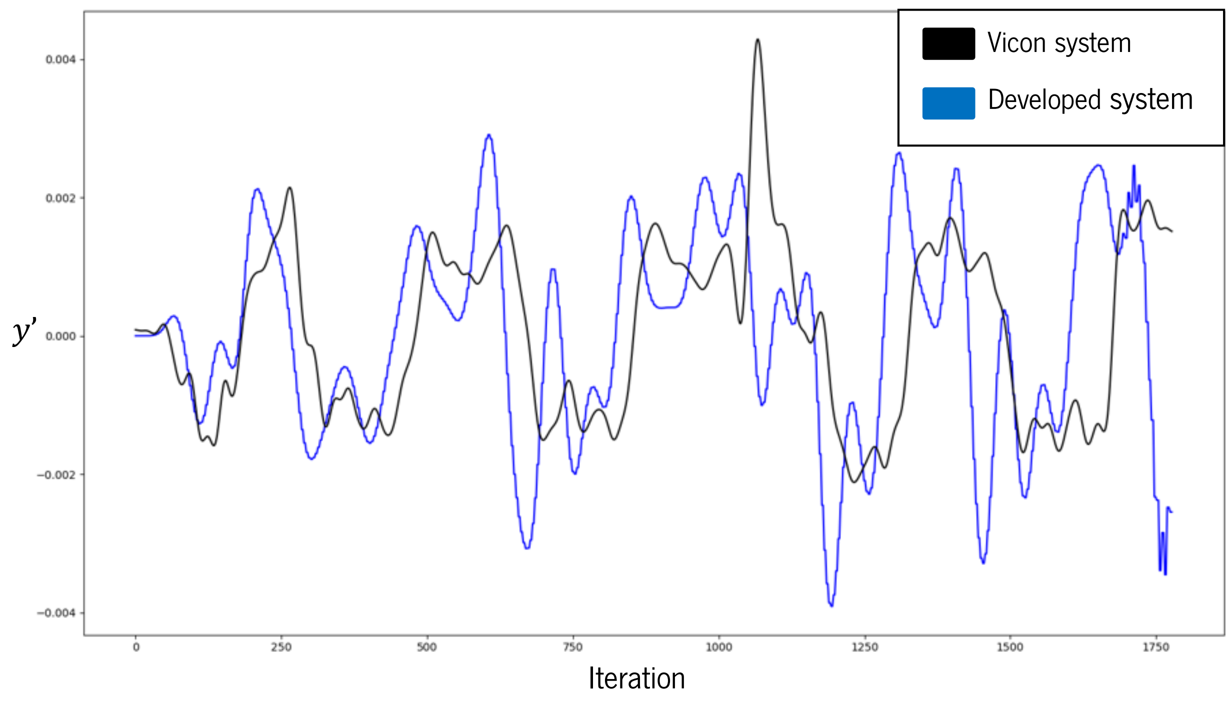


Figure 58- Gradient of "PM 1H" trajectory in the y direction, from the developed system and Vicon system.

As it was said above, the metric selected to evaluate quantitively the results from our solution is the mean squared error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \tag{5.1}$$

with n being the number of data points, Y_i the ground truth values from the Vicon system and $\hat{Y}_i$ the estimated value with the developed system. If equation (5.1) is used for the MSE in each direction of the trajectory, to get the single error value for the whole trajectory one must compute the norm of the error vector as,

$$MSE_{trajectory} = \sqrt{MSE_x^2 + MSE_y^2 + MSE_z^2} \tag{5.2}$$

The single MSE value for every trajectory is presented in Table 7. By analysing these values is difficult to get to any conclusion since they are so small. This is not necessarily due to the gradients being similar, but because the position information used in its calculation is in meters (very small values) which leads to even smaller gradient values. A way to extract meaningful data is to normalize these values relative to the maximum gradient value of every corresponding trajectory. This way is possible to understand the weight of this error in the overall context of the trajectory.

Table 7- MSE values for trajectories considered in the tests. The MSE values are also normalized relative to the maximum derivative registered in each trajectory.

Trajectory	**Traction 1H**	**Traction 2H**		**PM 1H**	**PM 2H**		**Torsion 1H**	**Torsion 2H**	
		lhand	rhand		lhand	rhand		lhand	rhand
MSE $\left(\frac{m}{\delta t}\right)$ $(\times 10^{-6})$	3.79	5.00	2.28	4.27	2.30	4.60	1.75	4.66	2.43
Maximum gradient $\left(\frac{m}{\delta t}\right)$ $(\times 10^{-3})$	6.00	1.80	3.50	2.70	4.40	4.40	2.50	4.10	5.10
Normalized MSE $(\times 10^{-3})$	0.63	2.78	0.65	1.58	0.52	1.04	0.70	1.14	0.48

From the information obtained, namely the normalized MSE (equation (5.3)), one can conclude that, in general our system can capture the main features of the trajectories of the hands.

$$Normalized\ MSE = \frac{MSE}{Maximum\ gradient} \tag{5.3}$$

The registered differences are either extremely small (close to zero) or at a scale that has a very small impact on the overall purpose of the movement. This can also be evaluated by analysing qualitatively the trajectories obtained from both applications. In Figure 59 is possible to see that the main features of one of the most complex trajectories, "PM 1H", were captured by our solution since this has a similar shape to the one provided by the Vicon system. Obviously, the trajectory from our proposed solution has more noise and it isn't as accurate in terms of locations as one can see from the different scales, but the overall intention of the trajectory was acquired. It is also important to mention that all the trajectories (x, y, z) shown next were positioned in space to facilitate their visual comparison. Therefore, one should not care about the specific

position in space of each individual trajectory but rather be interested in the relative information between them, such as the shape of one relative to the other.

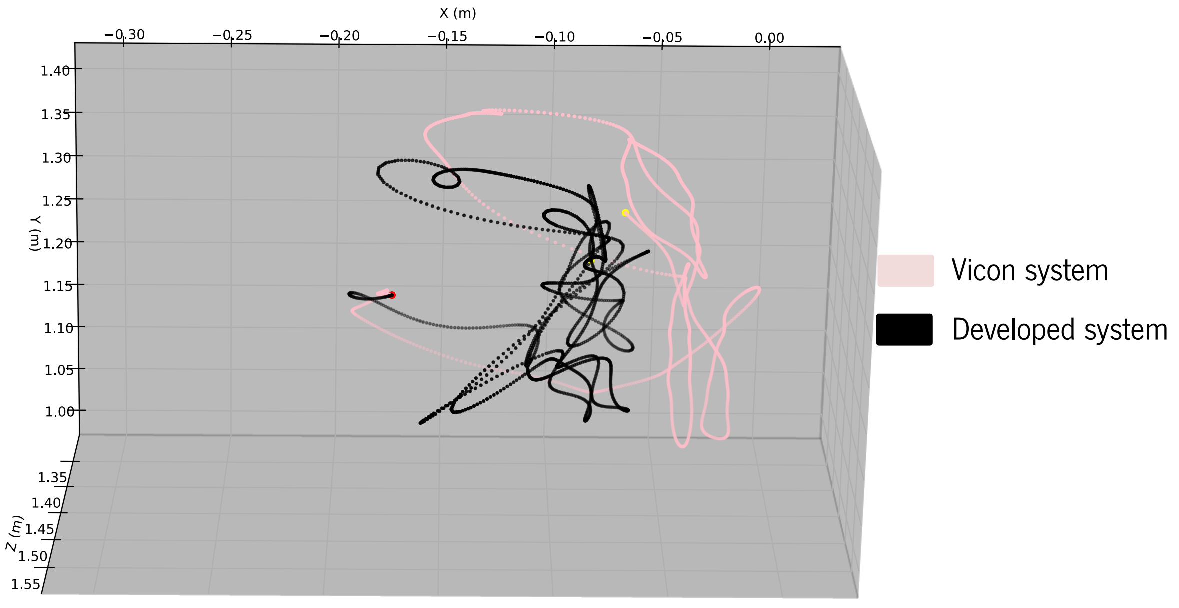


Figure 59- “PM 1H” trajectory from the developed system and the Vicon system.

It is also possible to see in Figure 58 and Figure 60 that the corresponding gradients in each direction also present some noise compared to the gradients from the Vicon system which in part reflects all the additional noise visible in Figure 59. One thing that came to our attention is that the gradient in the y direction is far more stable and less noisy than the one computed for the x and z direction. This tendency is probably caused by some poor calibration of the camera and by extra depth noise caused by occlusions of the hand during the movements.

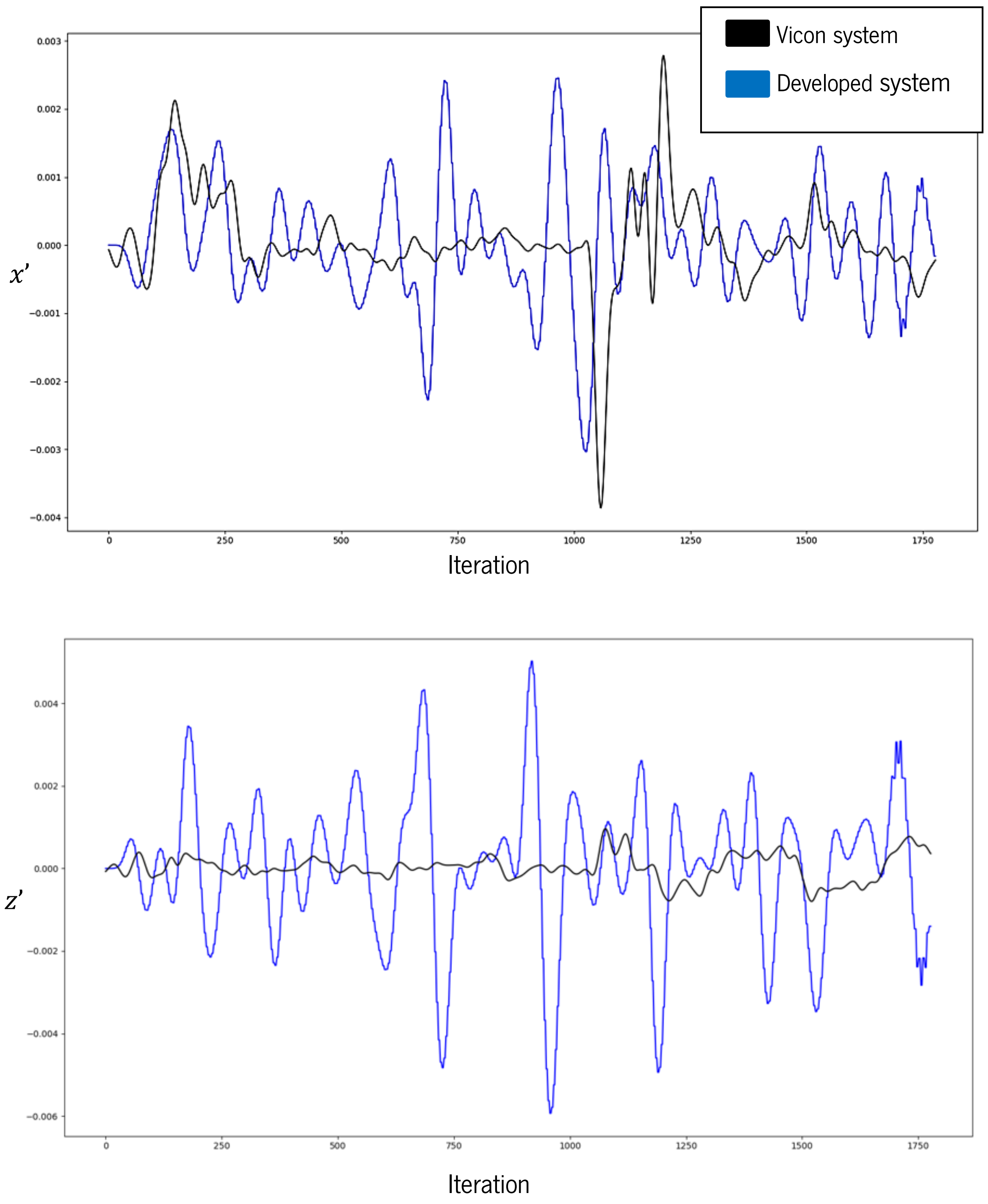


Figure 60 – Gradient in the x and z direction of trajectory "PM 1H".

Other trajectories are presented in Figure 61 and Figure 62. In the first one, once again is possible to indentify some key similarities between both trajectories as the overall intention of the movement is captured by Openpose. In terms of the overall shape of the trajectories it is clear that more or less the core part of the movements is captured by Openpose. However, in terms of relative position within the trajectories (scale) this is not the case, since most trajectories acquired present scale differences to the ones obtained with Vicon. In some cases, these differences are quite severe as shown in Figure 62.

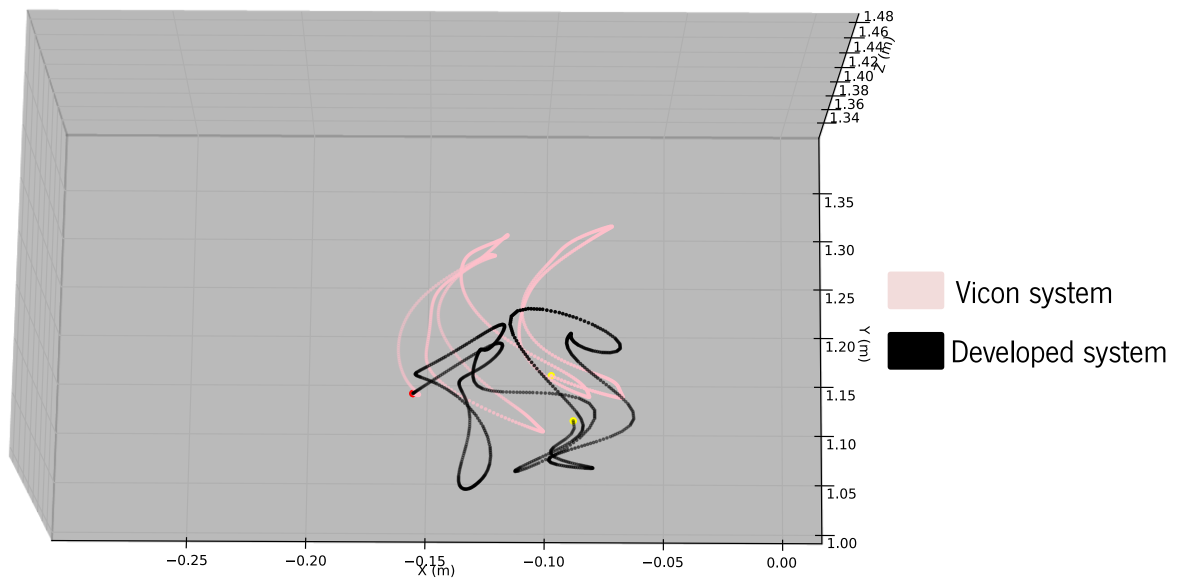


Figure 61- "Torsion 1H" trajectory acquired by the developed system and the Vicon system

In some types of movements these position errors within the trajectory may not be relevant, for example if the position of the hand is poorly estimated by 5 cm during a periodic movement, is very likely that it won't make a difference on the outcome since the leather piece is folded during the movement. However, in movements like torsion where the leather piece is near its geometrical limit, missing the location of the hand by 5 cm may lead to the robot potentially damaging the object.

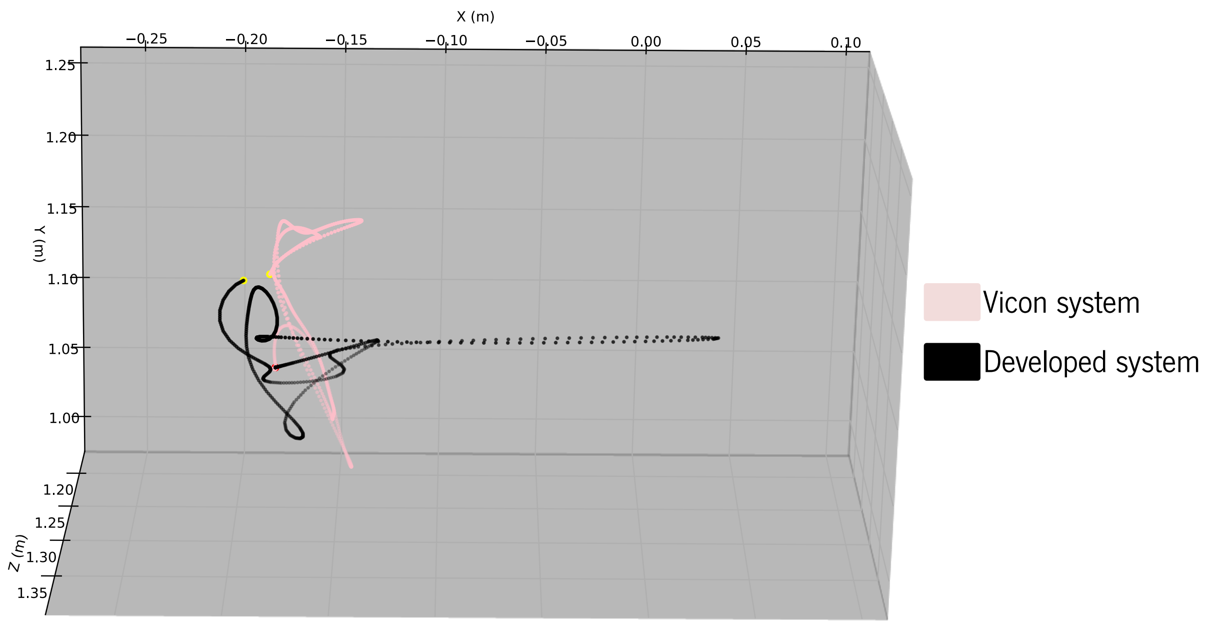
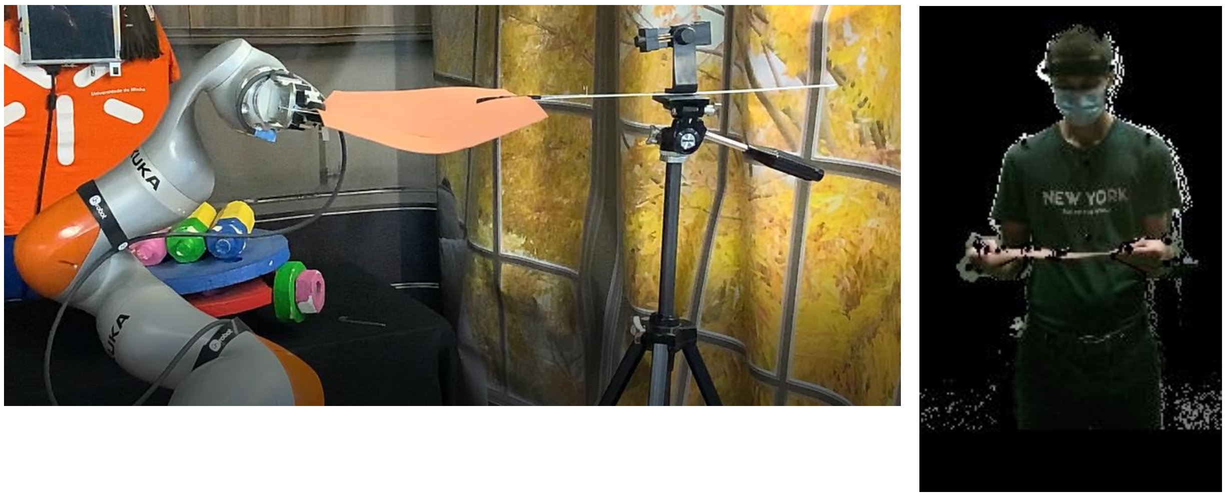
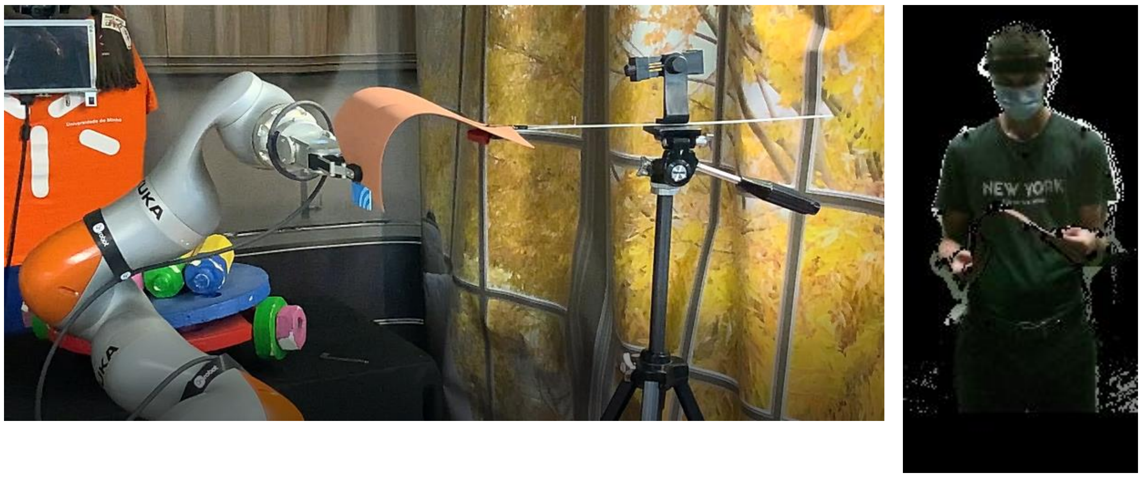


Figure 62 - "Traction 1H" trajectory acquired by the developed system and the Vicon system.

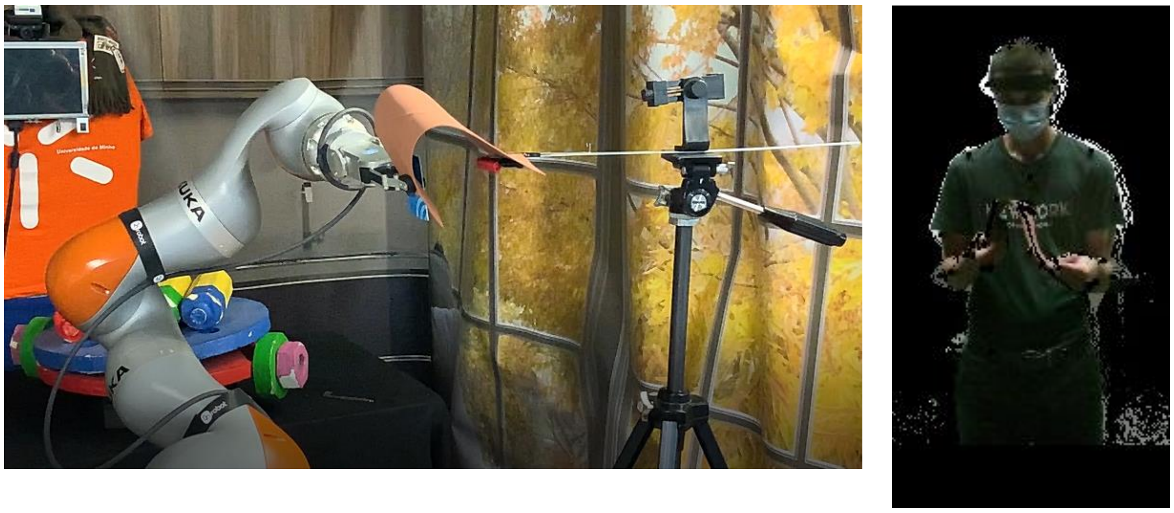
To prove that in some cases these errors might not be crucial, the trajectory “PM 1H” was used to manipulate a leather piece using a robot, namely the Kuka LBR iiwa 14. This trajectory was used because of its complexity and because only requires one moving support/hand.



1)

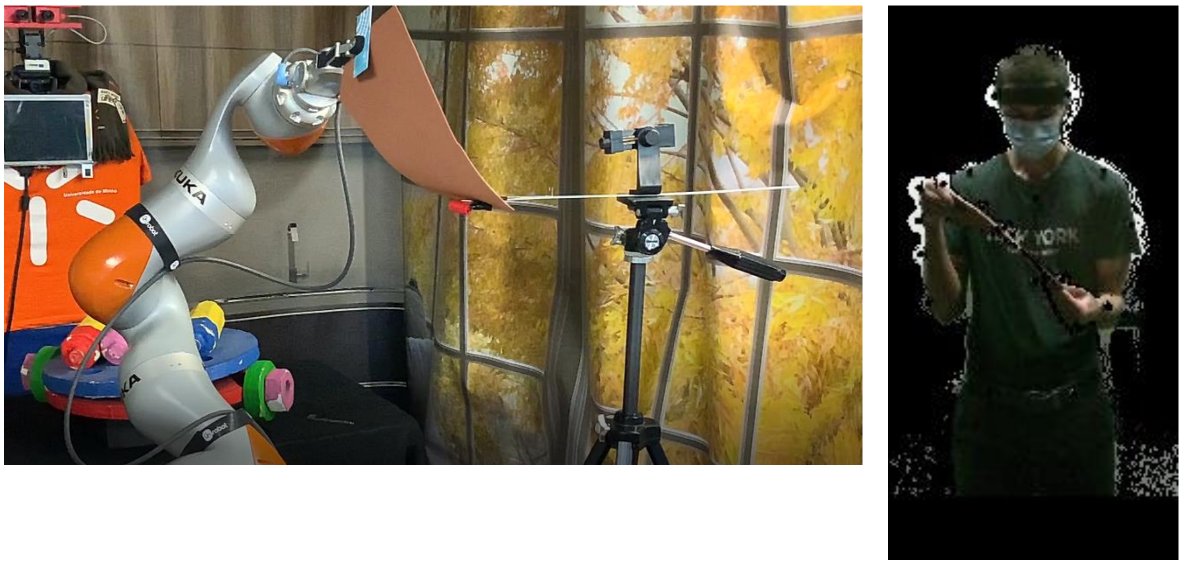


2)

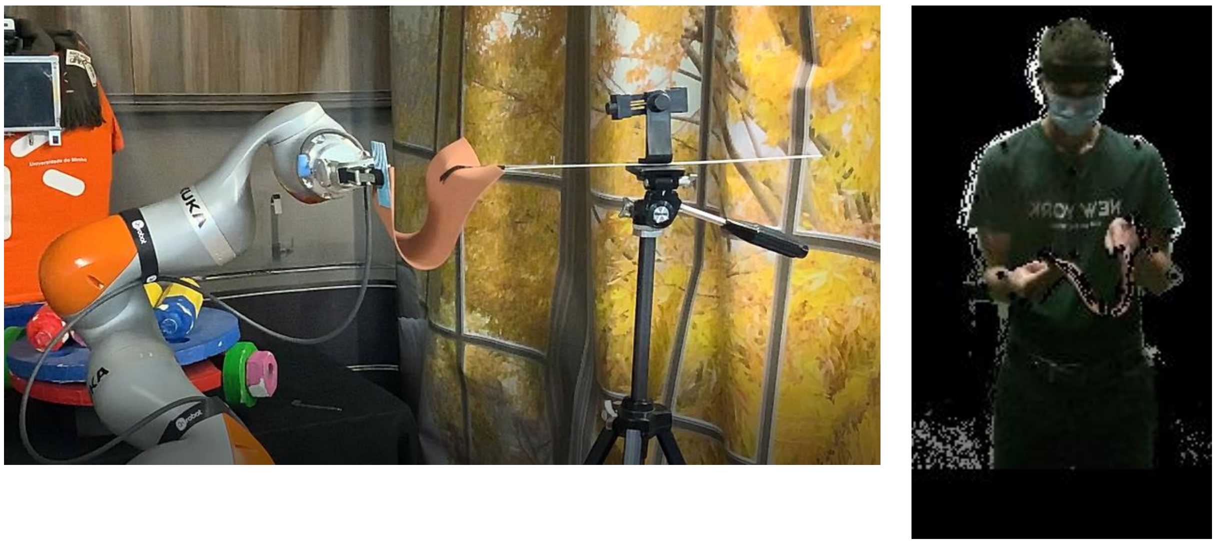


3)

Figure 63–Reproduction of inspection movement “PM 1H” with manipulator KuKa LBR iiwa 14.



4)



5)

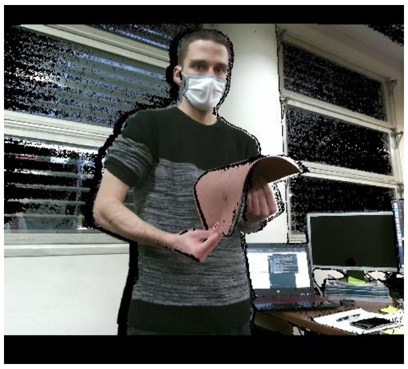
Figure 64- Reproduction of inspection movement “PM 1H” with manipulator KuKa LBR iiwa 14 (cont).

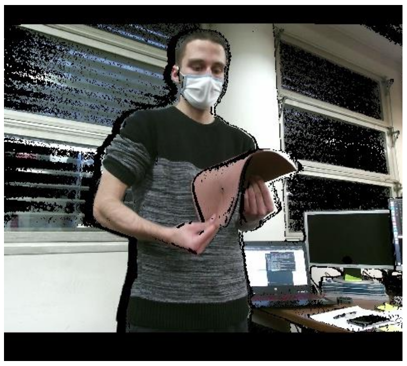
Even if our solution couldn’t estimate the hand position as accurately as the Vicon System (Figure 59), one can see from the images in Figure 64 that it was capable of reliably extracting the key parts of the movements that allow a similar deformation of the leather piece by the robot.

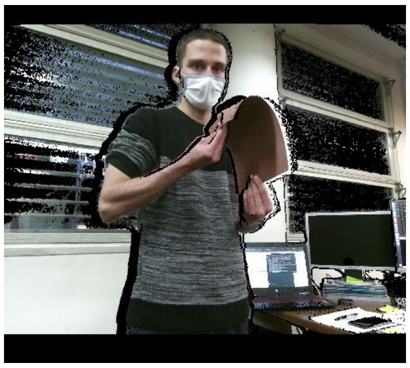
Despite the good results for the trajectory “PM 1H”, where the inconsistent position results did not harm the final goal, there are other types of trajectories where these errors may prevent their use. The poor results in terms of noise and accurate location of the hand are likely explained by the not ideal conditions in which the tests were carried out and not by the capabilities of our solution.

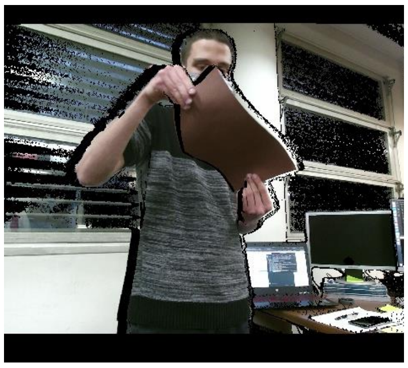
As explained in the description of our tracking solution two sources of information are needed to derive the 3D position of the hand: the 2D configuration estimated by Openpose and the depth value measured by the Kinect sensor. Due to restrictions of the Vicon system, the volunteer was too far away from the camera which spoiled the results from Openpose and made the hand more susceptible to occlusions. Since the hands were far from the camera, they appeared very small in the images processed by Openpose making the 2D pose estimation task harder. In addition, the presence of markers on the hand deteriorated the hand images even more. With the hands more susceptible to occlusions due to their distance to the camera, the depth measurements were also unstable.

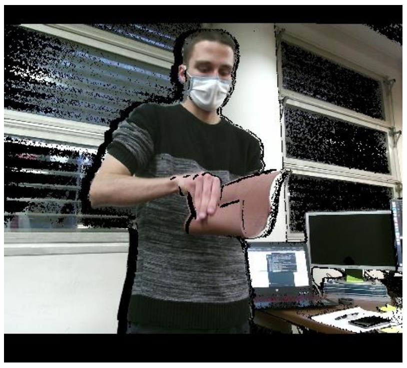
In order to check if in fact these were the causes for the limited performance of our solution in some trajectories, new ones similar to "PM 1H" and "Torsion 1H" were recorded without supervision of the Vicon system but this time with the person closer to the camera (Figure 65). The goal was to test with a clearer view on the hand if the system would provide smoother and more accurate results for position.


1)


2)


3)


4)


5)

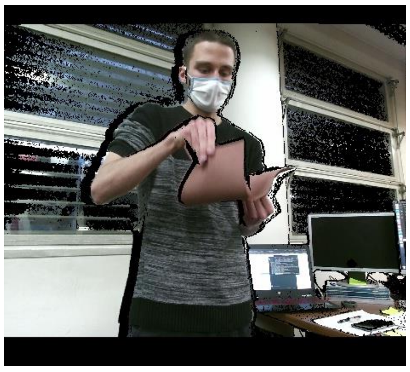

6)

Figure 65 – Inspection movement similar to "PM 1H" recorded closer to the camera with a clearer view on the moving hand.

Even though it is only possible to evaluate them qualitatively and they are not exactly the same in terms of orientation and position in space, the results for both new trajectories show clear improvements relative to the previous ones (Figure 66).

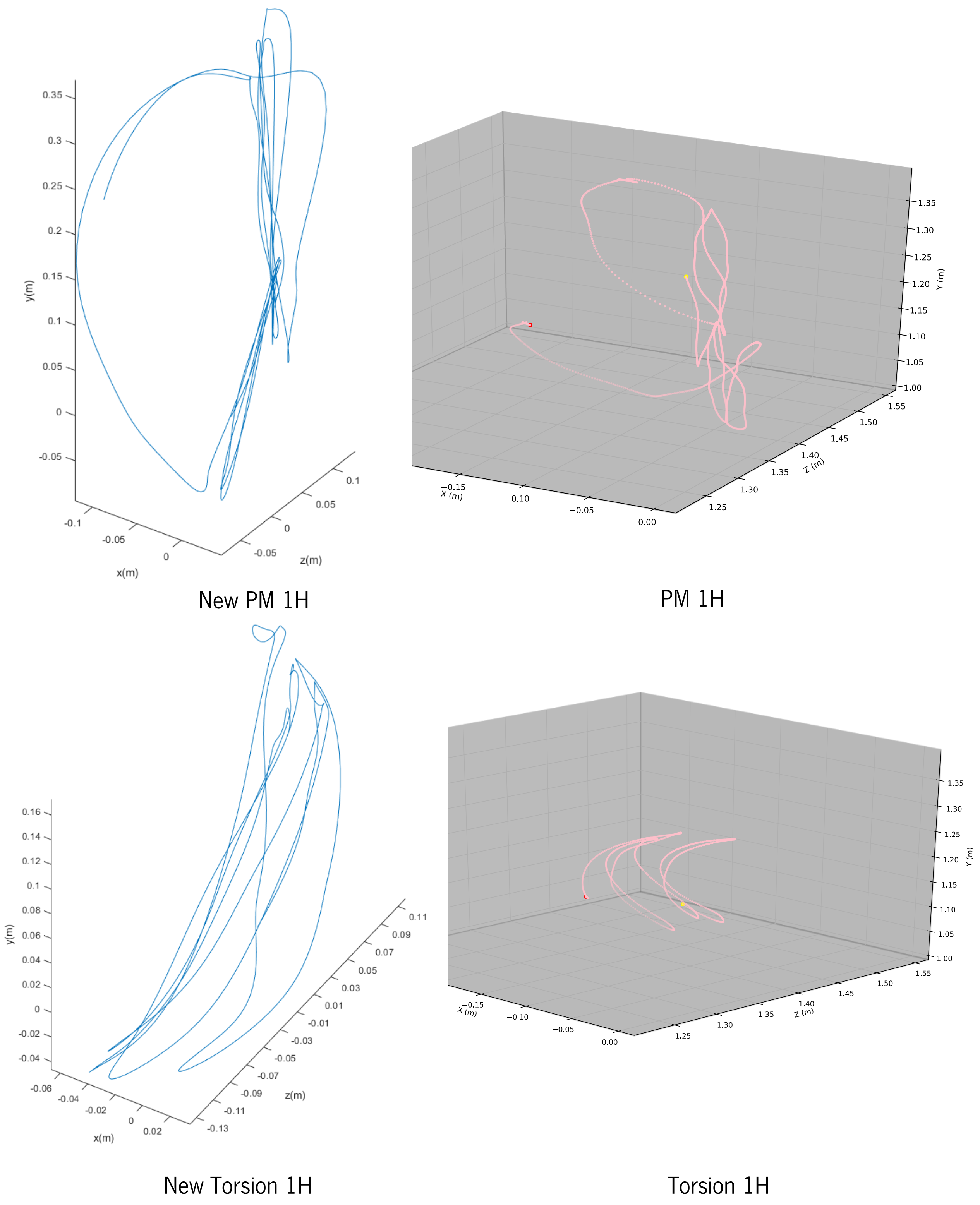


Figure 66 – Trajectories provided by our system for the new “PM 1H” and “Torsion 1H” trajectories and the first ones recorded with the Vicon system

They are significantly smoother and are actually more similar in terms of their overall shape to the trajectories provided by the Vicon system in the initial tests.

5.3 Orientation

The problems faced in the hand position tracking during the tests with the Vicon system had an even bigger impact on the estimation of the hand orientation. As explained in section 4.3, the implemented procedure for calculating the hand's orientation is highly sensitive to the information about the 3D hand configuration, which depends on the Openpose and Kinect sensor data. If the 2D location of the hand keypoints is estimated poorly or the wrist gets occluded, the corresponding 3D hand pose estimated by our method will also degenerate. Therefore, the orientation will be wrongly defined.

The conditions in which the tests were made (explained for the case of position) lead once again to highly incoherent and noisy results for the orientation. In Figure 67 is possible to see the pose of three hand fingers poorly estimated by ITP in the "PM 1H" trajectory which led to a bad computation of the hand orientation. The high disparity between the orientations obtained and the actual configuration of the hand made it impossible to establish any kind of comparison with the data collected by the Vicon and the one provided by our solution.

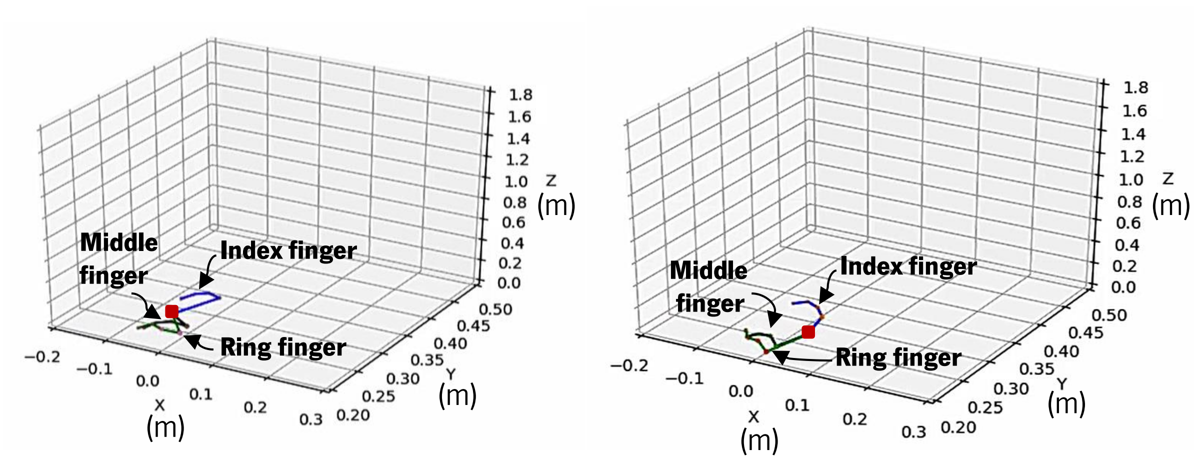
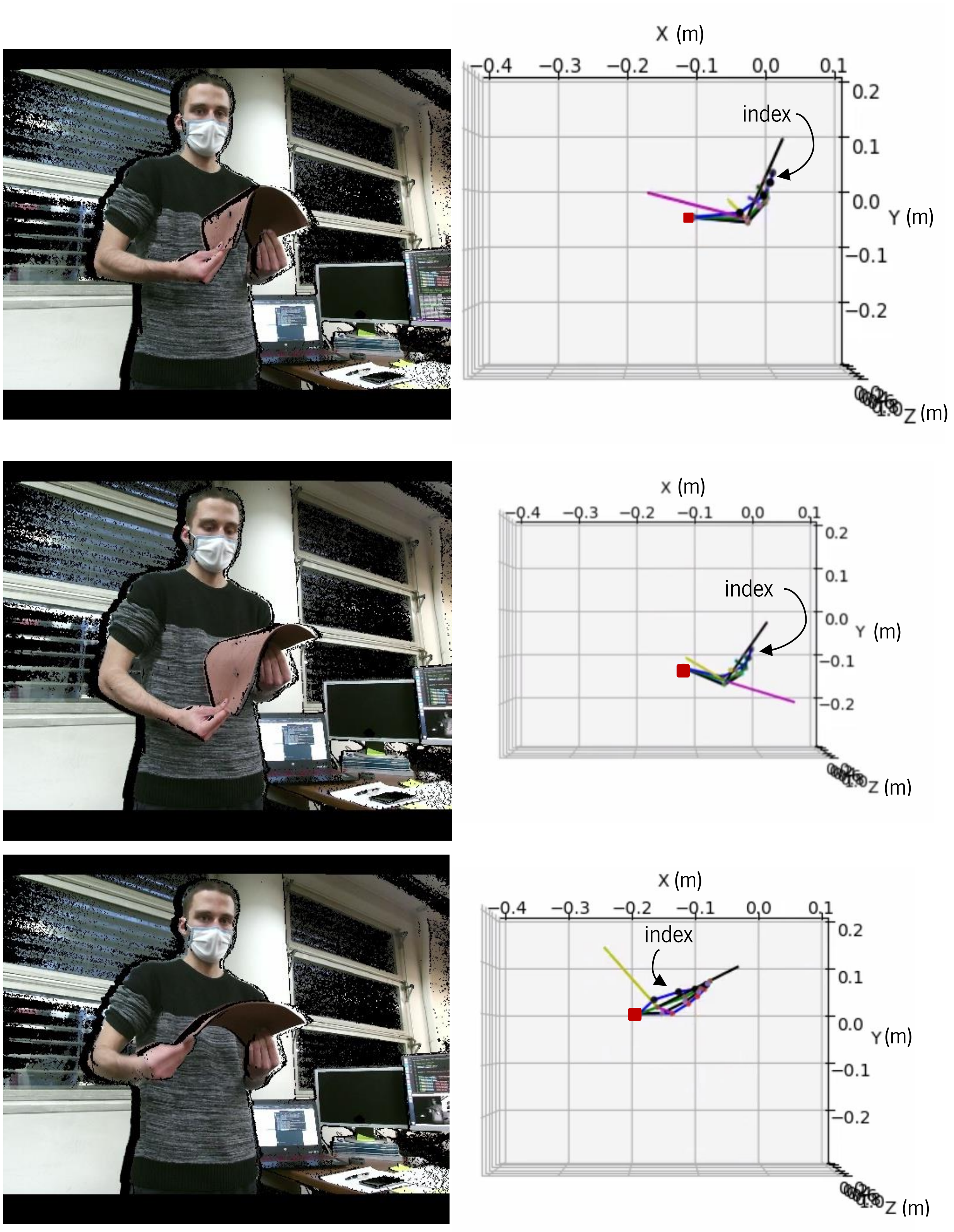


Figure 67 – Pose of three hand fingers (red: index finger; black: middle finger; blue: ring finger) poorly estimated by ITP in two different moments of the trajectory "PM 1H". (Note: red rectangle represents the hand wrist).

Hence, in order to accurately test the capabilities of our solution a better source of information had to be considered. For this matter, the two latest videos mentioned in section 5.2 were used. In these videos the wrist of the right hand was never occluded during the movements and the hand scale was much bigger for Openpose to estimate its pose. With this approach is not possible to establish any type of metric to evaluate the orientation results considering that there was no ground truth. However, a qualitive assessment can be made on how well our solution performed under these new conditions.

In Figure 68 one can see the 3D pose of the hand estimated in multiple stages of the trajectory new "PM 1H" trajectory recorded and it is clear that the results are robust and reliable. In most of the frames our iterative pose estimation method was able to reconstruct the hand's 3D pose very similarly to what is seen in the recorded videos, this proofs that if good images are provided to our application it is capable of providing solid results. This outcome is very important because it ensures that at this point the 3D information estimated for each hand keypoint is accurately determined within the hand context



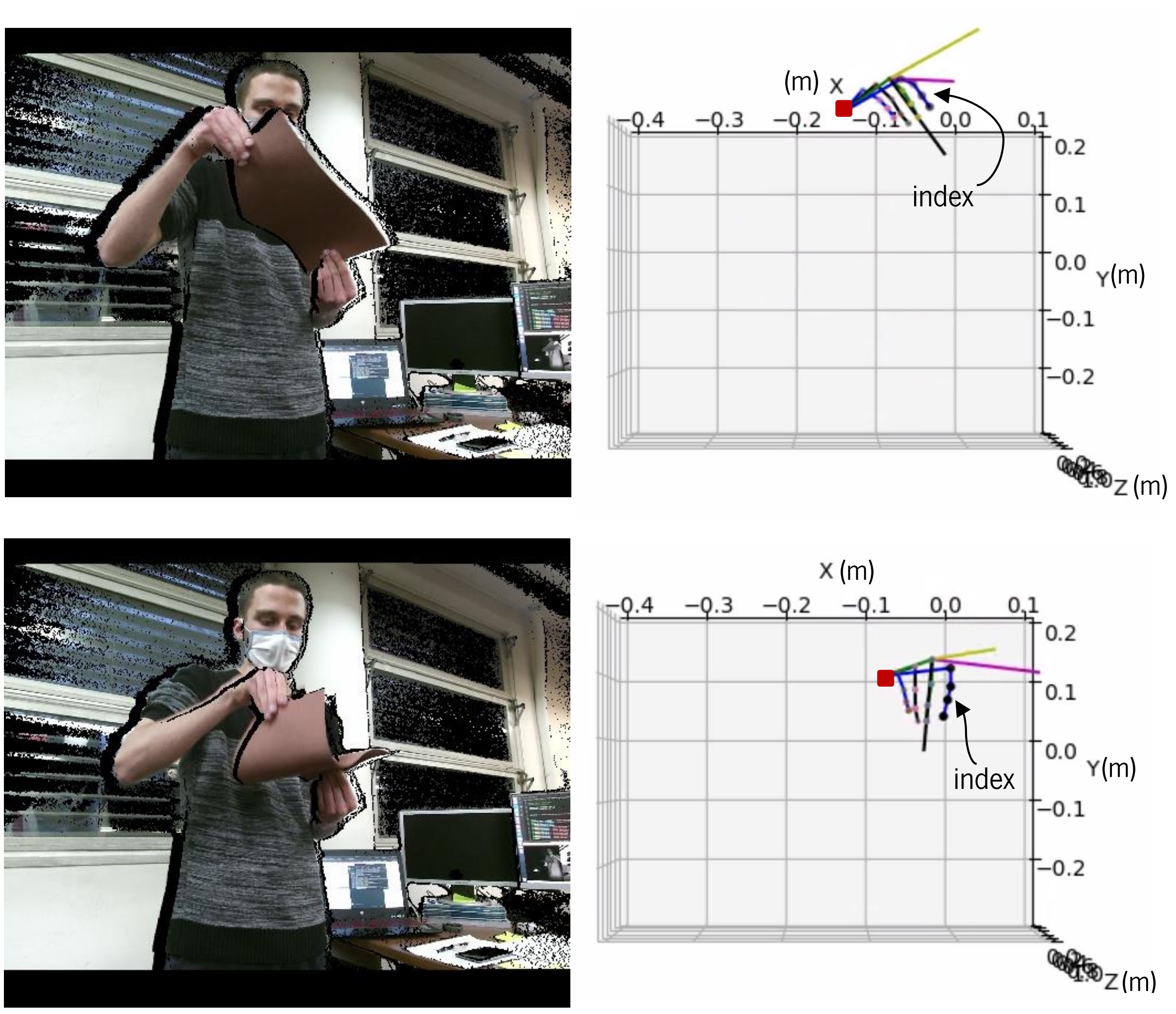


Figure 68 - – Inspection movement similar to "PM 1H" and the corresponding 3D pose estimation from ITP for the right hand. A reference frame representing the hand's orientation is also attached to the hand. (Note: red rectangle represents the hand wrist).

In the case of the torsion movement in Figure 69 the results show again the reliability of our proposed 3D pose estimator as it predicts very similar 3D hand configurations to the ones registered in the video. This occurs even in the presence some parts of the hand are occluded, where without this method it would be impossible to get the right 3D location of the hand keypoints.

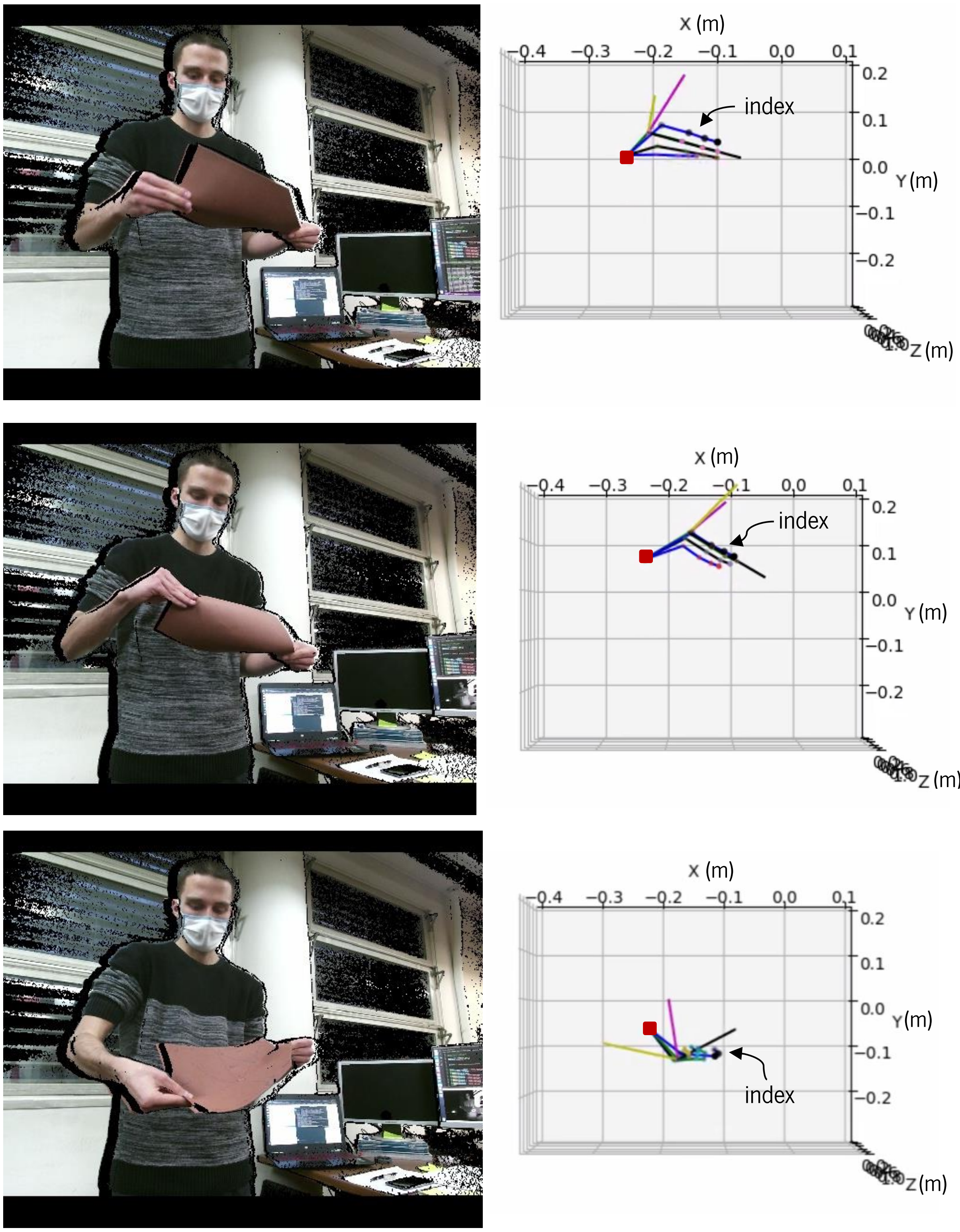


Figure 69- Inspection movement similar to "Torsion 1H" and the corresponding 3D pose estimation from ITP. A reference frame representing the hand's orientation is also attached to the hand.

Once the 3D poses are generated, three points must be selected to calculate the orientation for the robot's gripper. The decision on which points to take should consider certain aspects that might help interpreting the results. First it is helpful to choose three points that define a plane with a simple relative orientation to the leather piece (e.g.: parallel or perpendicular), this will make it easier way to evaluate the resulting orientations against the actual configuration changes of the hand. Second, it is important to decide if points from different fingers will be considered or not. After some tests it was clear that using more than one finger to estimate the hand orientation was not the best choice because there was a higher chance of getting inaccurate estimations for the orientation. This is due to the fact that the hand model considered in the Iterative Pose Estimation method does not account for restrictions between fingers (Figure 43) which allows non-realistic phenomenon such as finger overlapping. Even if this doesn´t occur in the majority of the frames, it only takes a couple of these cases to make the trajectory impossible for to follow by the manipulator. Hence, only points of the same finger were considered, namely within the middle finger (keypoints: 0,9 and 12).

After calculating the orientation for each point in the trajectory, these were tested in the robotics simulator CoppeliaSim[10]. Here it was possible to simulate the movements with an LBR iiwa 14 and check if the manipulator could follow the entire trajectory with the calculated orientations and more importantly if the orientation of the robot gripper during the movements was similar to the hand's during the demonstration.

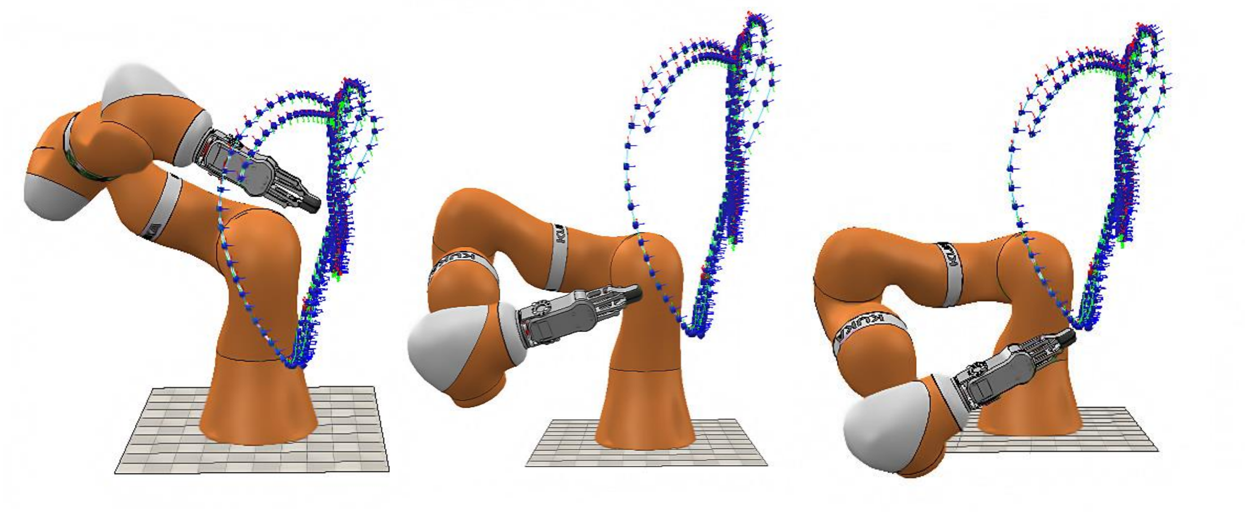


Figure 70-Simulation of the trajectory similar to "PM 1H" in CoppeliaSim with the KuKa LBR iiwa 14.

The results for this trajectory were very promising as the robot was able to follow the entire trajectory, which in complex trajectories like this is not an easy task. Besides this, also the orientation of the gripper was similar to what one would expect from the demonstrated movement during most part of the trajectory. However, there was one small portion where an unexpected rotation of the gripper occurred. This wasn't expected because the hand during the inspection movement never did anything similar. Considering the smoothness of the rotation and the way it fits the remaining orientations makes it less likely to be a problem of our calculations. Which leads us to the conclusion that it was probably a

[10] Robotics simulator available at: https://www.coppeliarobotics.com/

problem of how the mapping from the hand to the gripper was considered. This might be one topic to explore in future work but due to time restrictions it wasn't possible to explore more about this problem. In the case of the torsion trajectory depicted in Figure 71, the simulations shown different problems from the ones seen in the trajectory of Figure 70.

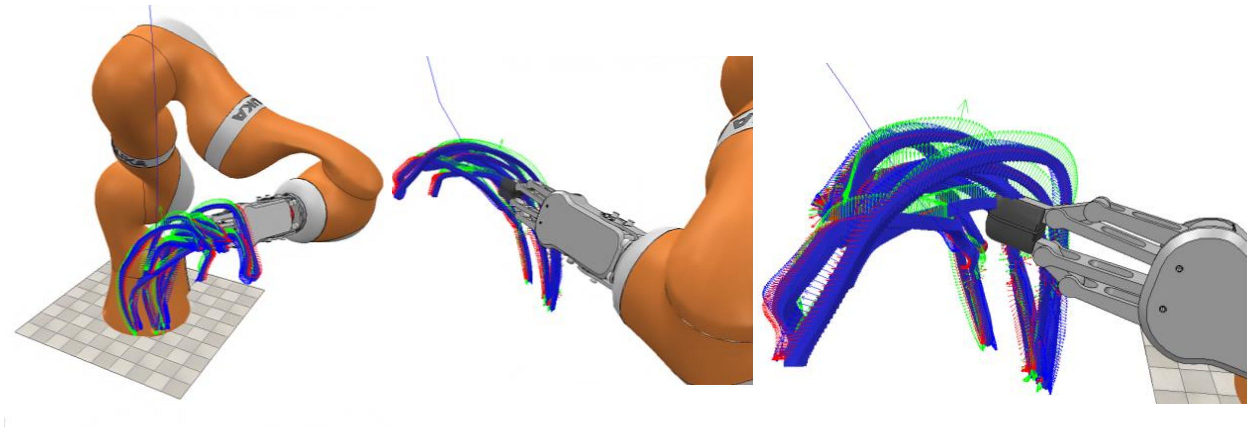


Figure 71 - -Simulation of the trajectory similar to "Torsion 1H" in CoppeliaSim with the KuKa LBR iiwa 14.

In this simulation the robot was not able to execute the whole trajectory due to the highly demanding changes in orientation that ended up sending the manipulator to its joint limits or a singularity[11]. These changes were most of the times not supposed to happen according to the hand example. In Figure 72 one can notice that in the top or bottom of the trajectory the orientation (coloured coordinate system attached to a blue cube) varies quite smoothly most of the times. However, in the bottom part are visible various segments where these abrupt changes in the orientation occur, which difficult the robot's work in following the trajectory and resulting in a wrong configuration of the leather piece.

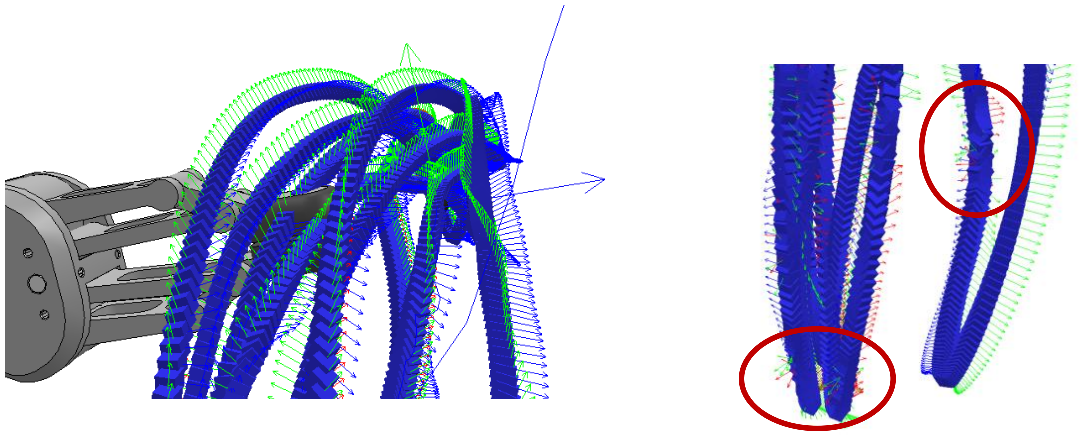


Figure 72- Left: Detailed view ot the top of the trajectory similar to "Torsion 1H"; Right: Detailed view of the bottom of the trajectory similar to "Torsion 1H" with highlights on abrupt changes in orientation.

[11] Singularity of a manipulator is a configuration in which the robot end-effector becomes blocked in certain directions.

It is clear that the current method used to compute the orientation from the 3D hand pose estimated is not robust enough. For that reason, a second stream of information could help correcting some of these weird phenomena that occur in the orientation. Even though this solution searches for the hand's orientation, information about the orientation of the leather piece's surface could help correct some of the mistakes from the proposed method. In order to check if this information can in fact be useful, the data collected from the leather piece's surface during the Vicon systems tests was used to extract its orientation at the grasping points.

As shown in Figure 53, 10 markers were placed in the surface of the leather piece, and our approach consisted in getting the 3D position of all the markers and connecting them through 4th order polynomials as exemplified in Figure 73.

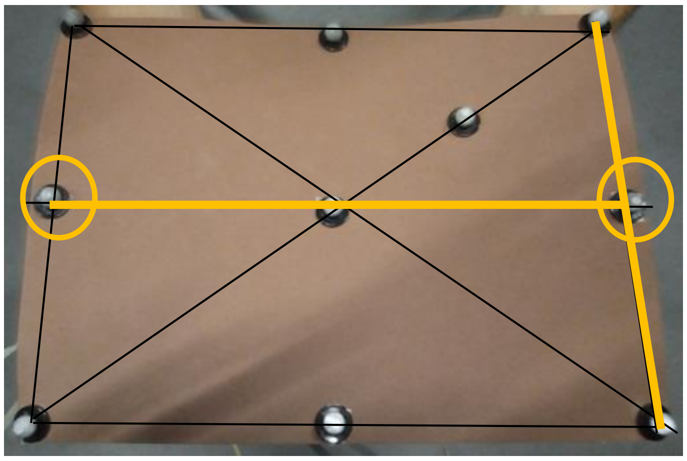


Figure 73-Connections considered among the makers placed in the leather piece surface and markers and links of interest to define the surface orientation are highlighted.

The polynomials were fitted through least squared error using the function *numpy.polyfit()* from the Python library Numpy. After finding the polynomials linking all the markers, it was a matter of computing the derivative of the polynomials at the place of interest as shown in Figure 73.

The horizontal polynomial was used to compute director vector for the z direction and the vertical one was used for the y direction. To obtain the director vector for the x direction one must compute the cross product of the other two. The results from this procedure are presented in Figure 74.

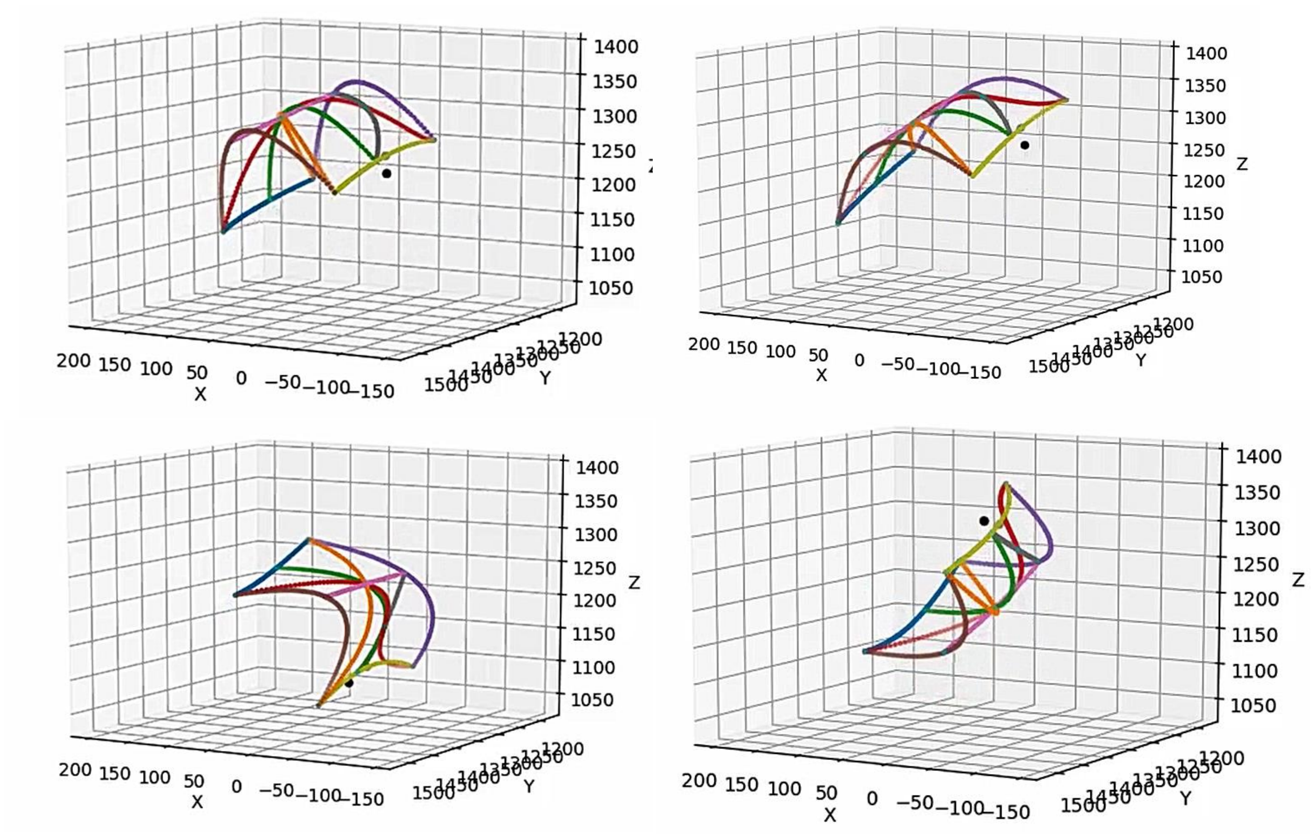


Figure 74- Representation of the leather piece surface usig polynomial fitting for the markers. Black and grey dots on the right side of the leather piece represent the direction (from the marker near the grasping point) of the gradient of the surface at the grasping point

The director vectors can be used to compute the matrix from equation (3.4) and obtain the corresponding orientation for the robot end-effector. This method to extract orientation was tested with the surface data acquired during the trajectory “MP 1H” recovered by the Vicon system and it was implemented on the robot together with the position information recovered by our solution for the same trajectory (Figure 59). The results from this combination were already presented in Figure 64. The orientation extracted from the leather piece surface revealed itself as an extremely efficient way of getting accurate orientations for the robot. To achieve a more robust solution both approaches can be used to correct each other. Future implementation should be considered.

6. Conclusions and Future Work

6.1 Conclusions

The work presented in this dissertation concerns the development and implementation of a 3D hand tracking solution that must be capable of outputting the hand trajectories (set of positions and orientations) executed during the manipulation of deformable objects, namely leather pieces. It starts by studying the current state-of-the-art solutions to tackle this problem. Glove and vision-based systems are the two main approaches to acquire the necessary data to describe the 3D hand pose. Even though glove-based solutions present a much more robust, accurate and reliable solution, they usually are quite expensive and in applications that are meant for direct use of the public, or manipulation of objects is involved it presents a less intuitive and natural solution due to all the equipment mounted on the hand.

This led to an increasing interest in vision-based solutions since these can work with either very few markers placed on the hand or with the hand completely free. Considering the software and hardware developments of the last few years, especially with the resurgence of machine learning algorithms in computer vision and the appearance of low-cost depth cameras, it became possible to develop very powerful solutions using a very small set of devices. These types of systems are normally preferred when there is no need of extremely accurate and robust measurements, but an intuitive, practical and low-cost solution is required.

Openpose was the vision-based application selected for tracking the 3D hand pose during the inspection procedure for leather pieces. It only requires RGB images of the hands and outputs the 2D coordinates within the image of 21 keypoints that construct the hand skeleton. In order to get the 3D position and orientation of the hand, it was necessary to use a low-cost depth camera, namely the Kinect sensor from Microsoft.

Since Openpose is not capable of processing video stream in real time, first, it was necessary to develop a recording pipeline to store the recorded videos frame by frame to later feed them one by one to Openpose and guarantee that all frames are evaluated. With this camera was possible to obtain the depth value of each keypoint which along with 2D coordinates from Openpose allowed for the estimation of the 3D world coordinates of each hand keypoint using the basic pinhole camera model.

To define the hand position, one needs to select the keypoint on the hand that reliably describes the hand movements, and more importantly is less prone to being occluded during the movements. The depth ambiguity related to occlusions of the hand also interferes with the estimation of the hand orientation.

For that reason, the Iterative pose estimation method is proposed. This method predicts the right depth value for all the keypoints on the hand starting from the wrist, which means that the iterative process starts with the acquired values for the wrist and estimates the remaining points using hand model restrictions. In order to get accurate estimates for the depth values and corresponding 3D hand pose is necessary that the wrist is never occluded during the manipulation, and if this is guaranteed the method is capable of accurately estimating the 3D hand pose even when some of its parts are occluded.
With the estimated 3D world position for the keypoints, the hand's orientation is defined using a rotation matrix that is built using three keypoints from the hand. These points were carefully selected to better describe the hand's orientation relative to the leather piece and having in mind that this orientation will be for later implementation with a manipulator.
The results from the developed solution were divided into position and orientation. In terms of position, the system can capture the right features of the inspection trajectories that allow for an accurate manipulation of the leather pieces even in rough recording conditions, as the test made with a real manipulator can prove. If the recording conditions are improved, these results become less noisy and the estimated 3D world coordinates of the hand are also more accurate. However, the selection of the hand keypoint that describes the 3D world coordinates of the hand is a crucial factor if one wants good results, as this point must be selected to avoid occlusions as much as possible.
In the case of orientation, the proposed "**Iterative Pose Estimation Method"** is capable of accurately predicting the 3D hand pose of the hand with results very similar to the real hand. This is an important achievement of this work considering that it presents a robust way of compensating the depth ambiguity problems that arose from the use of only one camera to record the scene. With this method is possible to accurately estimate hand depth values even when occlusions occur. However, even with the previous method being capable of estimating robust 3D hand poses, the method defined for calculating/defining the hand orientation was still too sensitive to noise from those predictions and was not capable in some trajectories of providing stable and coherent results for the orientation. Even with most of the trajectories (position and orientation) being defined accurately only by our system, there were still some parts of the movements where unexpected changes occurred, mainly with respect to orientation. According to the tests made this was most likely due to the noise of our measurements (depth and 2D pose from Openpose) and to the approach defined to calculate the hand's orientation. Whenever complex changes in orientation occurred during the inspection movements, our method was not capable of accurately replicating the same variations.

Considering the difficulties of our approach, it was explored a possible alternative where information about the deformation of the leather piece is also considered during the movements. This information was useful to extract the orientation of the leather piece at the contact point with the hand which provided better results than our previous method. This new approach even allowed for a successful real test with a robot manipulating a leather piece. This indicates that a more robust solution could include another stream of information about the deformation of the leather piece during the manipulation that would complement and correct the results from our system. A solution like this should be studied in the future.

6.2 Future work

The work made throughout this dissertation is expected to establish a solid ground base for a more robust system in the future. The developed solution shows promising results in terms of tracking the hand position while using only one camera to record the scene. However, it is still necessary to be careful with quality of the images recorded (hand occlusions and image noise) and the method is still too sensitive to the selection of the point on the hand that represents its position. In the future, to avoid problems with hand occlusions a more robust set of cameras could be implemented to record the inspection movements and see if it would improve the results and make the solution robust enough to deal with more complex trajectories. This strategy could also minimize the dependency of the results on the point chosen initially to define the hands position.

Regarding the hand orientation, the results were not as good as the ones for position. However, one of the main advances presented by this work is the proposed "**Iterative Pose Estimation Method"** that allows to accurately estimate the 3D hand skeleton only relying on the 3D position of the wrist and the 2D hand skeleton estimated by Openpose. Even with this method, the defined approach to calculate the hand's orientation was not capable of providing consistent and robust results. Two approaches were considered to improve this performance: first, study other possible approaches to define the hand orientation; second, maintain the current method but explore the introduction of different streams of information to help correct the poor predictions from our system. This information could be for example about the deformation of the leather piece at the point of contact with the hand.

One other thing that could be improved in the future is the system real-time performance of our system. For this matter, a quicker method could be implemented to initially find the hand within the images. This could save some processing time that Openpose is currently consuming with its robust tracking method that unnecessarily finds all body parts, including the hands.

REFERENCES

[1] "Manufacturing Next-generation Robotic Manipulation - SRI International." https://www.sri.com/blog-archive/manufacturing-next-generation-robotic-manipulation/ (accessed Nov. 24, 2021).

[2] S. Schaal, "Learning from demonstration," *Adv. Neural Inf. Process. Syst.*, pp. 1040–1046, 1997, doi: 10.1007/978-3-030-63416-2_300375.

[3] "RG2 Gripper - Flexible 2 Finger Robot Gripper | OnRobot." https://onrobot.com/us/products/rg2-gripper (accessed Dec. 17, 2021).

[4] P. Jiménez, "Survey on model-based manipulation planning of deformable objects," *Robot. Comput. Integr. Manuf.*, vol. 28, no. 2, pp. 154–163, 2012, doi: 10.1016/j.rcim.2011.08.002.

[5] J. Sanchez, J. A. Corrales, B. C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey," *Int. J. Rob. Res.*, vol. 37, no. 7, pp. 688–716, 2018, doi: 10.1177/0278364918779698.

[6] W. Wang, D. Berenson, and D. Balkcom, "An online method for tight-tolerance insertion tasks for string and rope," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2015-June, no. June, pp. 2488–2495, 2015, doi: 10.1109/ICRA.2015.7139532.

[7] B. Lu, H. K. Chu, and L. Cheng, "Dynamic trajectory planning for robotic knot tying," *2016 IEEE Int. Conf. Real-Time Comput. Robot. RCAR 2016*, pp. 180–185, 2016, doi: 10.1109/RCAR.2016.7784022.

[8] Y. Bin Jia, F. Guo, and H. Lin, "Grasping deformable planar objects: Squeeze, stick/slip analysis, and energy-based optimalities," *Int. J. Rob. Res.*, vol. 33, no. 6, pp. 866–897, 2014, doi: 10.1177/0278364913512170.

[9] A. Namiki and S. Yokosawa, "Robotic origami folding with dynamic motion primitives," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2015-Decem, pp. 5623–5628, 2015, doi: 10.1109/IROS.2015.7354175.

[10] B. Jia, Z. Pan, Z. Hu, J. Pan, and D. Manocha, "Cloth Manipulation Using Random-Forest-Based Imitation Learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2086–2093, 2019, doi: 10.1109/LRA.2019.2897370.

[11] A. Doumanoglou *et al.*, "Folding clothes autonomously: A complete pipeline," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1461–1478, 2016, doi: 10.1109/TRO.2016.2602376.

[12] M. C. Gemici and A. Saxena, "Learning haptic representation for manipulating deformable food objects," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 638–645, 2014, doi: 10.1109/IROS.2014.6942626.

[13] L. Zaidi, J. A. Corrales, B. C. Bouzgarrou, Y. Mezouar, and L. Sabourin, "Model-based strategy for grasping 3D deformable objects using a multi-fingered robotic hand," *Rob. Auton. Syst.*, vol. 95, pp. 196–206, 2017, doi: 10.1016/j.robot.2017.06.011.

[14] D. Mohr and G. Zachmann, "Hand pose recognition — Overview and current research," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8844, pp. 108–129, 2015, doi: 10.1007/978-3-319-17043-5_7.

[15] L. Dipietro, A. M. Sabatini, and P. Dario, "A survey of glove-based systems and their applications," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 38, no. 4, pp. 461–482, 2008, doi: 10.1109/TSMCC.2008.923862.

[16] W. Chen *et al.*, "A survey on hand pose estimation with wearable sensors and computer-vision-based methods," *Sensors (Switzerland)*, vol. 20, no. 4, 2020, doi: 10.3390/s20041074.

[17] J. J. L. Jr., "A Survey of Hand Posture and Gesture Recognition Techniques and Technology," Providence, 1999.

[18] K.-Y. Chen, S. N. Patel, and S. Keller, "Finexus," pp. 1504–1514, 2016, doi: 10.1145/2858036.2858125.

[19] D. J. Sturman and D. Zeltzer, "A Survey of Glove-based Input," vol. 161941010, pp. 30–39, 1994.

[20] M. Caeiro-Rodríguez, I. Otero-González, F. A. Mikic-Fonte, and M. Llamas-Nistal, "A systematic review of commercial smart gloves: Current status and applications," *Sensors*, vol. 21, no. 8, 2021, doi: 10.3390/s21082667.

[21] A. Rashid and O. Hasan, "Wearable technologies for hand joints monitoring for rehabilitation: A survey," *Microelectronics J.*, vol. 88, no. June 2017, pp. 173–183, 2019, doi: 10.1016/j.mejo.2018.01.014.

[22] J. Connolly, K. Curran, J. Condell, and P. Gardiner, "Wearable rehab technology for automatic measurement of patients with arthritis," *2011 5th Int. Conf. Pervasive Comput. Technol. Healthc. Work. PervasiveHealth 2011*, pp. 508–509, 2011, doi: 10.4108/icst.pervasivehealth.2011.246010.

[23] B. O' Flynn *et al.*, "Integrated Smart Glove for Hand Motion Monitoring," *SENSORCOMM 2015 Ninth Int. Conf. Sens. Technol. Appl.*, no. c, pp. 45–50, 2015.

[24] "Senso Glove - The best controller for AR/VR." https://senso.me/ (accessed Apr. 15, 2021).

[25] "Cobra Motion Capture Gloves | AiQ Synertial." https://www.synertial.com/cobra-gloves (accessed Apr. 15, 2021).

[26] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," *Comput. Vis. Image Underst.*, vol. 108, no. 1–2, pp. 52–73, 2007, doi: 10.1016/j.cviu.2006.10.012.

[27] J. M. Palacios, C. Sagués, E. Montijano, and S. Llorente, "Human-computer interaction based on hand gestures using RGB-D sensors," *Sensors (Switzerland)*, vol. 13, no. 9, pp. 11842–11860, 2013, doi: 10.3390/s130911842.

[28] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Efficient model-based 3D tracking of hand articulations using Kinect," no. January, pp. 101.1-101.11, 2011, doi: 10.5244/c.25.101.

[29] E. Thabet, F. Khalid, P. S. Sulaiman, and R. Yaakob, "Algorithm of local features fusion and modified covariance-matrix technique for hand motion position estimation and hand gesture trajectory tracking approach," *Multimed. Tools Appl.*, vol. 80, no. 4, pp. 5287–5318, 2021, doi: 10.1007/s11042-020-09903-5.

[30] D. Mohr and G. Zachmann, "A Survey of Vision-Based Markerless Hand Tracking Approaches," *Prepr. Submitt. Comput. Vis. Image Underst.*, 2013.

[31] C. Keskin, F. Kiraç, Y. E. Kara, and L. Akarun, "Hand pose estimation and hand shape classification using multi-layered randomized decision forests," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7577 LNCS, no. PART 6, pp. 852–863, 2012, doi: 10.1007/978-3-642-33783-3_61.

[32] D. Tang, T. H. Yu, and T. K. Kim, "Real-time articulated hand pose estimation using semi-supervised transductive regression forests," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 3224–3231, 2013, doi: 10.1109/ICCV.2013.400.

[33] H. Liang, J. Yuan, and D. Thalmann, "Parsing the hand in depth images," *IEEE Trans. Multimed.*, vol. 16, no. 5, pp. 1241–1253, 2014, doi: 10.1109/TMM.2014.2306177.

[34] D. Tang, H. J. Chang, A. Tejani, and T. K. Kim, "Latent regression forest: Structured estimation of 3D hand poses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 7, pp. 1374–1387, 2017, doi: 10.1109/TPAMI.2016.2599170.

[35] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 824–832, 2015, doi: 10.1109/CVPR.2015.7298683.

[36] C. Wan, A. Yao, and L. Van Gool, "Hand pose estimation from local surface normals," *Lect. Notes*

Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9907 LNCS, pp. 554–569, 2016, doi: 10.1007/978-3-319-46487-9_34.

[37] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Trans. Graph.*, vol. 33, no. 5, 2014, doi: 10.1145/2629500.

[38] A. Sinha, C. Choi, and K. Ramani, "DeepHand: Robust Hand Pose Estimation by Completing a Matrix Imputed with Deep Features," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 4150–4158, 2016, doi: 10.1109/CVPR.2016.450.

[39] L. Ge, Y. Cai, J. Weng, and J. Yuan, "Hand PointNet: 3D Hand Pose Estimation Using Point Sets," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 8417–8426, 2018, doi: 10.1109/CVPR.2018.00878.

[40] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "Real-Time 3D Hand Pose Estimation with 3D Convolutional Neural Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 956–970, 2019, doi: 10.1109/TPAMI.2018.2827052.

[41] F. Zhang *et al.*, "MediaPipe Hands: On-device Real-time Hand Tracking," 2020, [Online]. Available: http://arxiv.org/abs/2006.10214.

[42] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 4645–4653, 2017, doi: 10.1109/CVPR.2017.494.

[43] C. Zimmermann and T. Brox, "Learning to Estimate 3D Hand Pose from Single RGB Images," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 4913–4921, 2017, doi: 10.1109/ICCV.2017.525.

[44] J. Y. Chang, G. Moon, and K. M. Lee, "V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation from a Single Depth Map," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 5079–5088, 2018, doi: 10.1109/CVPR.2018.00533.

[45] U. Iqbal, P. Molchanov, T. Breuel, J. Gall, and J. Kautz, "Hand Pose Estimation via Latent 2.5D Heatmap Regression," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11215 LNCS, pp. 125–143, 2018, doi: 10.1007/978-3-030-01252-6_8.

[46] L. Ge *et al.*, "3D hand shape and pose estimation from a single RGB image," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 10825–10834, 2019, doi: 10.1109/CVPR.2019.01109.

[47] A. Tagliasacchi, M. Schröder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly, “Robust articulated-ICP for real-time hand tracking,” *Eurographics Symp. Geom. Process.*, vol. 34, no. 5, pp. 101–114, 2015, doi: 10.1111/cgf.12700.

[48] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, “Realtime and robust hand tracking from depth,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1106–1113, 2014, doi: 10.1109/CVPR.2014.145.

[49] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, “Markerless and efficient 26-DOF hand pose recovery,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6494 LNCS, no. PART 3, pp. 744–757, 2011, doi: 10.1007/978-3-642-19318-7_58.

[50] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, “Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints,” *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2088–2095, 2011, doi: 10.1109/ICCV.2011.6126483.

[51] M. D. La Gorce, S. Member, D. J. Fleet, S. Member, and N. Paragios, “Model-Based 3D Hand Pose Estimation from Monocular Video,” vol. 33, no. 9, pp. 1793–1805, 2011.

[52] L. Ballan and A. Taneja, “Motion Capture of Hands in Action using Discriminative Salient Points.”

[53] D. Tzionas, A. Srikantha, P. Aponte, and J. Gall, “Capturing Hand Motion with an RGB-D Sensor, Fusing a Generative Model with Salient Points,” pp. 1–12.

[54] S. Sridhar, A. Oulasvirta, and C. Theobalt, “Interactive Markerless Articulated Hand Motion Tracking Using RGB and Depth Data.”

[55] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Second. Cambridge University press, 2003.

[56] J. J. Craig, “Spatial descriptions and transformations,” in *Introduction to Robotics Mechanics and Control*, Third., Pearson Education International, 2005, pp. 19–62.

[57] S. A. van D. Skogstad, K. Nymoen, M. E. Høvin, S. Holm, and A. R. Jensenius, “Filtering Motion Capture Data for Real-Time Applications,” *Proc. 2013 Conf. New Interfaces Music. Expr. (NIME 2013)*, no. January, pp. 142–147, 2013, [Online]. Available: https://www.duo.uio.no/handle/10852/37950.

[58] M. Bogdan and M. Panu, “LabVIEW modeling and simulation, of the digital filters,” *2015 13th Int. Conf. Eng. Mod. Electr. Syst. EMES 2015*, no. August, 2015, doi: 10.1109/EMES.2015.7158411.

[59] S. kumar Are, M. R. Thangalla, and S. Gajjala, “- SOFTWARE IMPLEMENTATION OF DIGITAL

FILTERS," Blekinge Institute of Technology, 2008.

[60] M. Oberweger, G. Riegler, P. Wohlhart, and V. Lepetit, "Efficiently creating 3D training data for fine hand pose estimation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 4957–4965, 2016, doi: 10.1109/CVPR.2016.536.

[61] S. E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 4724–4732, 2016, doi: 10.1109/CVPR.2016.511.

[62] L. Caruso, R. Russo, and S. Savino, "Microsoft Kinect V2 vision system in a manufacturing application," *Robot. Comput. Integr. Manuf.*, vol. 48, no. April, pp. 174–181, 2017, doi: 10.1016/j.rcim.2017.04.001.

[63] H. Gumgumcu and D. Laumer, "Hand Tracking using Kalman Filter for Safe Human-Robot Interaction," ETH Zurich, 2017.

[64] L. Valgma, "3D reconstruction using Kinect v2 camera," 2016.

[65] R. Brancati, C. Cosenza, V. Niola, and S. Savino, *Experimental measurement of underactuated robotic finger configurations via RGB-D sensor*, vol. 67, no. February 2019. Springer International Publishing, 2019.

[66] R. Li, H. Wang, and Z. Liu, "Survey on Mapping Human Hand Motion to Robotic Hands for Teleoperation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8215, no. 2019, 2021, doi: 10.1109/TCSVT.2021.3057992.

[67] P. Panteleris, I. Oikonomidis, and A. Argyros, "Using a single RGB frame for real time 3D hand pose estimation in the wild."

[68] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 4733–4742, 2016, doi: 10.1109/CVPR.2016.512.

[69] T. Hastie, R. Tibshirani, and J. Friedman, "Statistical Decision Theory," in *The Elements of Statistical Learning*, Second., California: Springer London, 2008, pp. 18–22.

[70] A. Cutler, D. R. Cutler, and J. R. Stevens, "Ensemble Machine Learning," *Ensemble Mach. Learn.*, 2012, doi: 10.1007/978-1-4419-9326-7.

[71] B. B. Traore, B. Kamsu-Foguem, and F. Tangara, "Deep convolution neural network for image recognition," *Ecol. Inform.*, vol. 48, no. October, pp. 257–268, 2018, doi: 10.1016/j.ecoinf.2018.10.002.

[72] "Magia do Operador Sobel." https://ichi.pro/pt/magia-do-operador-sobel-206419103773197

(accessed Nov. 19, 2021).

[73] A. Amidi and S. Amidi, “CS 230 - Convolutional Neural Networks Cheatsheet.” https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#overview (accessed Nov. 19, 2021).

[74] K. Leung, “How to Easily Draw Neural Network Architecture Diagrams | by Kenneth Leung | Towards Data Science,” 2021. https://towardsdatascience.com/how-to-easily-draw-neural-network-architecture-diagrams-a6b6138ed875 (accessed Nov. 19, 2021).

[75] F. Dornaika, S. E. Bekhouche, and I. Arganda-Carreras, “Robust regression with deep CNNs for facial age estimation: An empirical study,” *Expert Syst. Appl.*, vol. 141, p. 112942, 2020, doi: 10.1016/j.eswa.2019.112942.

[76] Prabhu, “Understanding of Convolutional Neural Network (CNN) — Deep Learning | by Prabhu | Medium.” https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 (accessed Nov. 22, 2021).

Appendix I – Theory behind Discriminative methods for 3D Hand tracking

RANDOM FOREST ALGORITHM

The random forest model is within the scope of machine learning as it is the case for convolutional neural networks. This algorithm can be used in both classification and regression problems, which makes it of great interest for hand pose estimation.

As the name suggests, a Random Forest is a tree-based ensemble with each tree considering a collection of random variables/features.

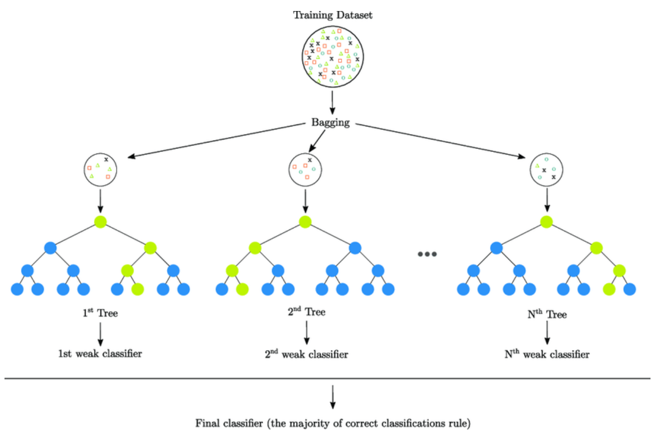


Figure 75- Working principle of the random forest algorithm, in this case for a classification problem.

More formally, for the case of regression, a p-dimensional random variable vector $X = (X_1, \ldots\ldots, X_p)$ representing the real-valued input or the variables used by the predictor to estimate the random variable Y representing the real-valued response, assuming an unknown joint probability distribution $P_{XY}(X, Y)$. Then, the goal is to find a prediction function $f(X)$ to predict Y. The prediction function is determined by a loss function $L\big(Y, f(x)\big)$ and defined to minimize the expected prediction error. In [69] the solution for minimizing the expected error pointwise is derived based on the estimated value of the loss function, some properties of the estimation operator and conditional probability. The solution obtained for the regression problem is,

$$f(x) = E(Y|X = x) \tag{0.1}$$

called the regression function, then the prediction of Y at any $X = x$ is the conditional mean $\mu_{Y|X=x}$ whenever the loss function is the average squared error. This solution sums up the behaviour of this algorithm. However, to have a clearer vision of how this applies in practice it's necessary to understand how a decision tree works, since these are the basic elements of the random forest model. To help the process let's consider an example case from [70] where the goal is to predict the level of prostate-specific antigen (lpsa) of patients, the response variable, based on other two variables through the predictor. For this example, the variables are the log prostate weight (lweight) and log cancer volume (lcavol).

All the training data is stored at the root node, an array filled with multiple trios, where each trio is composed by a response value (lpsa) and the corresponding variables lweight and lcavol. The next step is to split all these different trios between two more nodes according to some criterion as in Figure 76. The splitting process continues down the tree grouping the data into smaller subsets according to diffrent criterion defined by the designer, and it is in these criterions that the model can be optimized to yield more accurate predictions. For that, a principle must be defined to update the thresholds used to split the data at the nodes, these updates must be done in such a way that the error between the predictions and real values its decreased (ideally until it reaches 0), for example using the mean square error (see [70] for more details).

Since the goal is to find clusters of information that relate significantly the three variables involved it is necessary to stop the splitting process somewhere, therefore a final condition must be defined, for example, establishing that a node can't have less than three trios. This leads to the terminal nodes of the decision tree, which are responsible for the predictions of that specific tree.

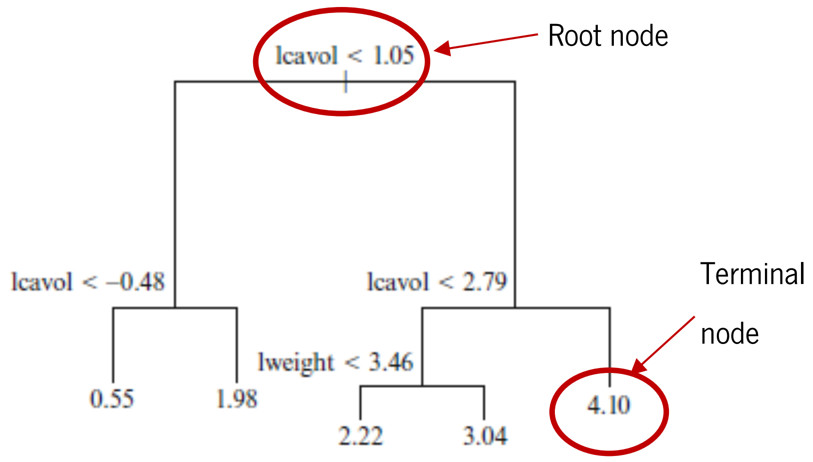


Figure 76- Diagram of regression tree for two-dimensional prostate cancer data. From [70].

At each terminal node is possible to find all the final subsets/clusters that resulted from all the splitting done up the tree, and at that point is possible to calculate the predicted lpsa value at each terminal node, namely the average value of the response values (lpsa) of the observations (trios) that fell into each terminal node. This applies equation (0.1, since it is being applied a conditional mean of all response values Y. If more trees are considered, then the final output of the random forest model is the mean of the prediction made by each tree.

CONVOLUTION NEURAL NETWORKS (CNNS)

Convolution neural networks (CNNs) are a class of Deep Neural Networks that can recognize and classify features and are widely used for analysing visual images. During this section it is assumed that the reader as some basic understanding of the components of neural networks and how they work, terms such as neurons, weights, bias, and layers should be familiar, since the goal for this section is only to give some basic understanding on CNNS, and not to explain the whole concept of neural networks in general. Convolution neural networks combine the basic working principles of neural networks and convolution.
The term 'Convolution" denotes the mathematical transformation on two functions (u and v) as it generates a third function, which is normally viewed as a transformed version of one of the initial functions. This third function is usually written as $u * v$, and for two complex or real functions, according to [71] its calculation is made with equation (0.2).

$$(u * v)(x) = \int_{-\infty}^{+\infty} u(x-t)v(t)dt \tag{0.2}$$

However, for the case of discrete convolution the calculations can be seen as matrix multiplication [71]. Since digital images can be represented as matrices of numbers, convolution becomes very useful to manipulate the information conveyed in these images, for example to find interesting information about them.

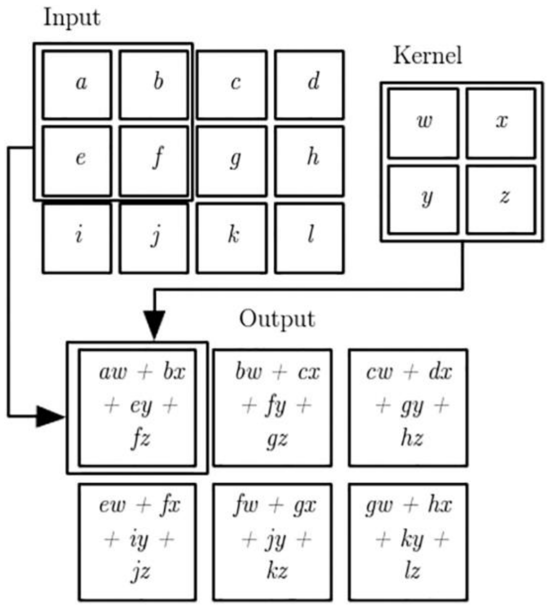


Figure 77- Example of discrete convolution with kernel 2X2. From [71]

In computer vision this is usually called a filtering operation, where a filter/kernel, is used to perform convolution over an image which generates a different one, where interesting features from the initial image are highlighted. These features can be, for example, vertical edges, sharp corners, or round edges according to the filter used. One common filter is the sobel operator which is a filter used to find the edges of an image as in Figure 78.

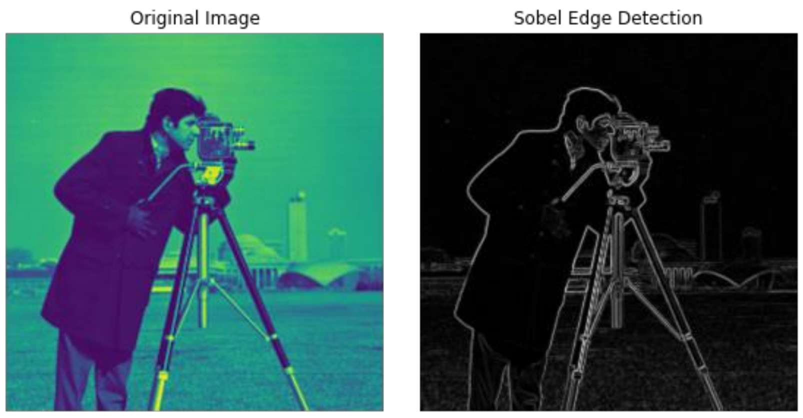


Figure 78-Example of the results of convolving a sobel operator over an image [72].

The convolution satisfies some algebraic properties (e.g., commutativity, associativity, distributivity and multiplicative identity) in order to ensure that the characteristics of their corresponding geometric pictures are preserved.

In a filtering operation, a kernel/filter is defined considering the type of features one intents to highlight on the image (for example an edge type in the image), then the kernel acts like a sliding window across the image and at each iteration convolution is calculated (Figure 77). The result of this operation is another image that acts as a feature map, highlighting the desired features. The procedure described is executed in the convolution layers of CNNs which usually follow the general architecture composed of the following layers: Convolution layer, Pooling layer, ReLU layer and Fully connected layer.

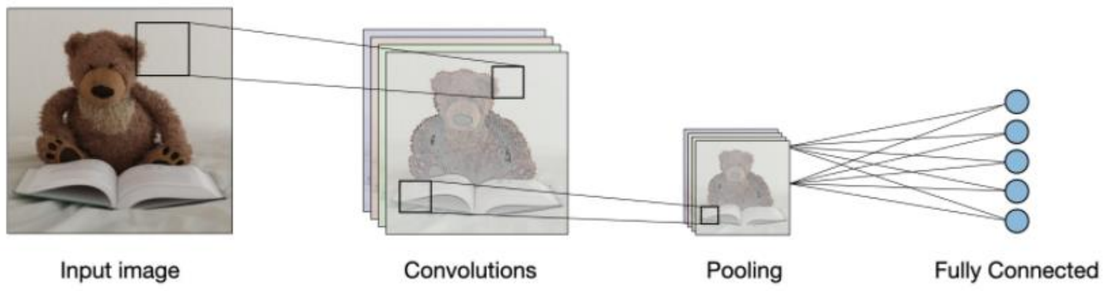


Figure 79 -General architecture of a traditional CNN [73].

Convolution layer:

This is the most important layer, and where the real power of this type of networks stands. It is here that the features from the input image are extracted to further combine them and establish a map between them and some final decision, classification, or value. These networks can have multiple convolution layers that highlight features in intermediate feature maps elevating the level of abstraction even more. The intention is to further refine the information that can be useful for the network to take conclusions out of it.

Depending on what is desired it might even be useful to consider more than just one filter in each layer and that way find multiple features, as many as the number of filters considered. The number of filters used at a layer defines the depth of its corresponding output, for example, if one uses 6 filters at a layer the corresponding output will be 6 more images that will go as input for following layers. In Figure 80, 64 filters were used in the first layer generating 64 output feature maps with a size of 224 x 224, in the folowwing layers only one or two filters were used.

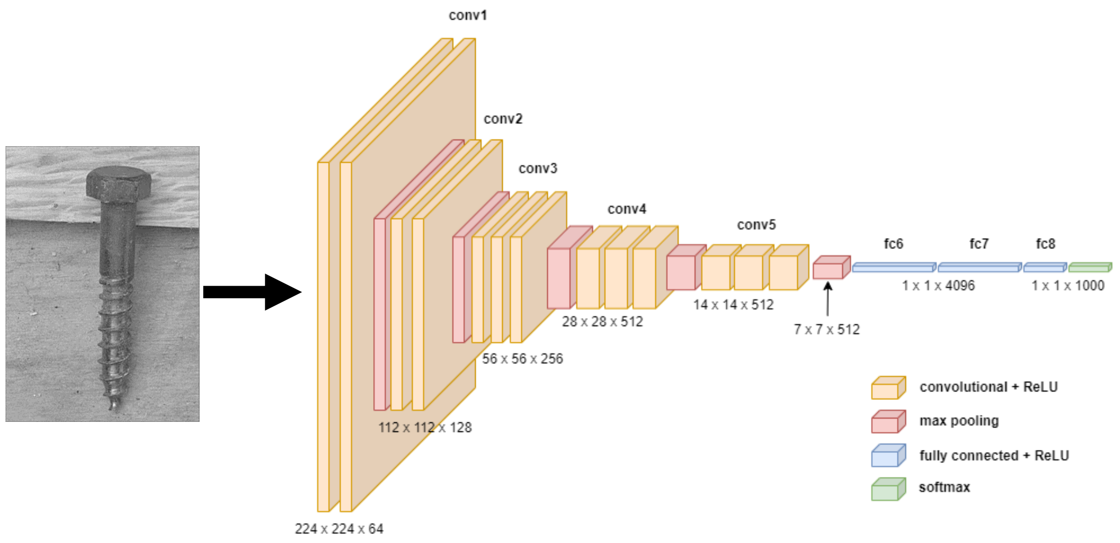


Figure 80- Example of a CNN architecture with multiple filters ate each convolution layer [74].

Besides the filter's depth there are other two hyper parameters that can control/change the output of a convolution layer, they are: the stride and zero-padding.

The stride denotes the number of pixels by which the filter window moves after each convolution operation as in Figure 81. Stride controls both overlapping of the receptive fields and the size of the output feature maps.

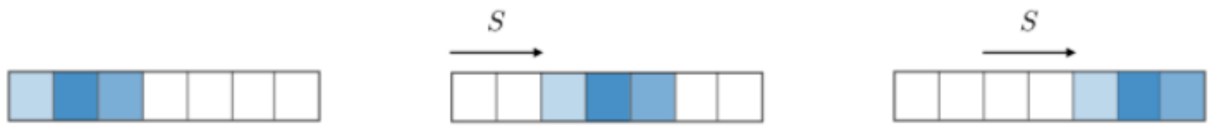


Figure 81-Example of filtering with stride equal to 2 [73].

Zero padding is useful to pad the input images with zeros on the border to control the output's spatial size. Considering that the convolution operation eliminates some data in the borders of the image (see

Figure 77), if there is the need to exactly preserve its spatial size for the following layers, there must be data added to those borders in order for the output to maintain its dimensions. The least influential type of information to add is zero.

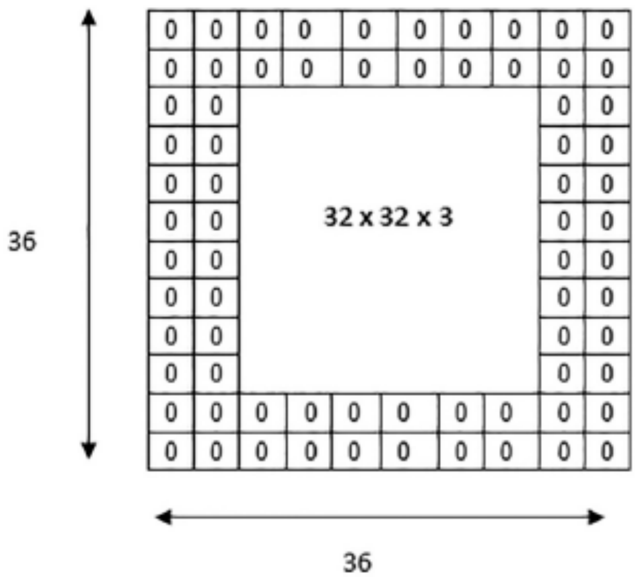


Figure 82- Example of zero padding process [71].

ReLU layer:

ReLU stands for the rectified linear unit, which is an activation function typically used in CNNs, other examples of activation functions are the hyperbolic tangent (tanh) and logistic sigmoid. Once the feature maps are extracted, the next step is to move them to a ReLU layer where an element-wise operation is performed and sets all the negative pixels to 0 (equation (0.3)). It introduces non-linearity to the network, and the generated output is a rectified feature map.

$$f(x) = \max(0, x) \tag{0.3}$$

Pooling layer:

The task of this layer consists of simplifying or reducing the information generated in the convolution layers. One way of doing this is reducing the number of pixels on the feature maps generated at each layer, which also leads to a reduction of the amount of memory needed to store all the information. There are three types of pooling: the first one is the average pooling, the second is the L2-norm pooling and finally the one of most popular use is the max pooling because of its speed and improved convergence. This basically takes an “empty” filter (normally of size 2×2) (Figure 83) and a stride of the same length. It then applies it to all input images and outputs the maximum number in every sub region that is covered by the filter which convolves around.

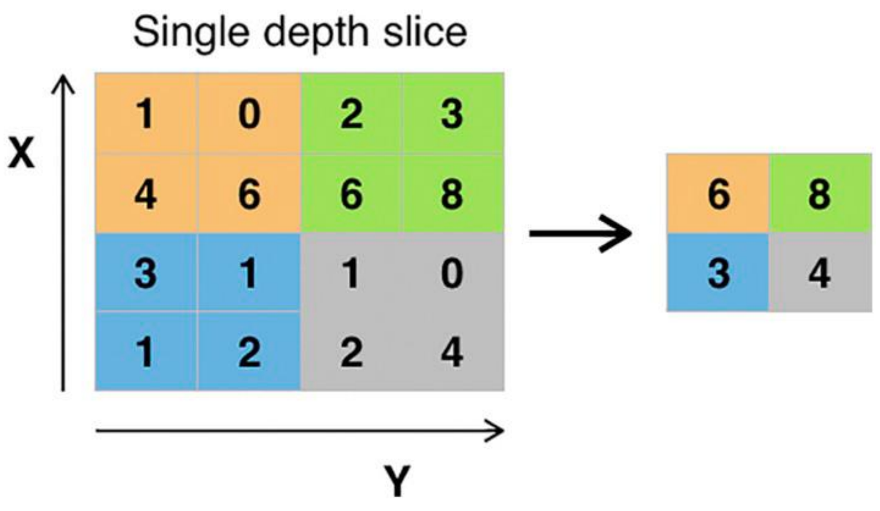


Figure 83- Illustration of the max pooling procedure to down sample data [71].

All the layers mentioned until now are responsible for learning the features from the initial input image, which is common to every convolution neural network, after getting all the relevant features, a task specific network can be built to solve the desired problem. In this part come the following layers.

Fully connected layer:

This layer basically takes as input the output of the last pooling layer and flattens it to a one-dimensional array. This is possible because of the drastic down sample performed over the previous layers making the feature maps so small that can be represented as vectors. Each of the inputs is connected to every one of the outputs hence the term “fully connected”. It is the last layer, in general after the last pooling layer in CNN process and outputs an N dimensional vector where N is the number of classes that the program must choose. Fully connected layers perform like a traditional neural network and contain about 90% of the parameters in CNN.

It allows us to feed forward the neural network into a vector with a predefined length. For image classification, we could feed forward the vector into certain number of categories. The output is also a vector of numbers. For a regression problem one could introduce for example a linear regression function after the classification step as in [75].

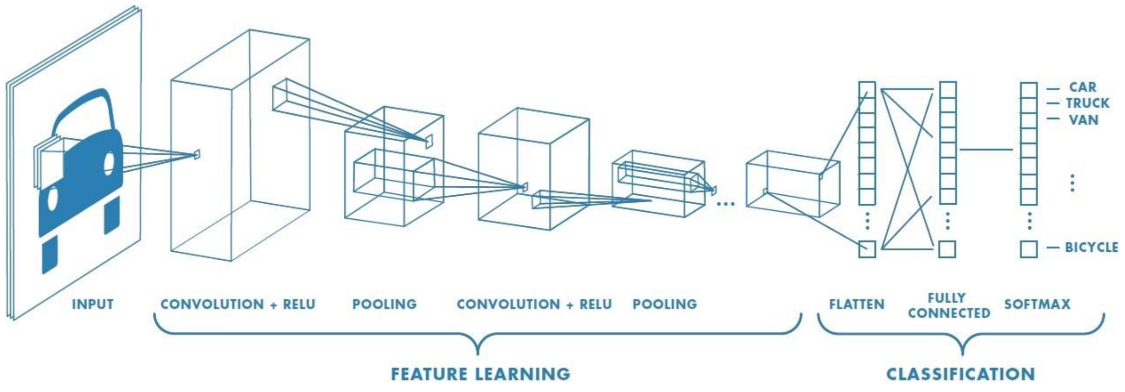


Figure 84-Two main blocks of a convolution neural network [76].

After reaching the end of the neural network one gets its prediction, however it is necessary to understand how well the network performed and how prepared it may be for future predictions. In this context an extra layer can be considered which is the loss layer.

This stage of the model uses functions that take in the model's output and the target, and it computes a value that measures the model's performance. These measurements can be used to help the model learn how to make more accurate predictions and this process is called training.

There are two stages to train the network: a forward stage and a backward stage. Firstly, the main goal of the forward stage is to represent the input image with the current parameters (weights and bias) in each layer. Then the prediction output is used to compute the loss cost to the ground truth labels. Second, based on the loss cost, the backward stage computes the gradient of each parameter with chain rules. All the parameters are updated based on the gradients and are prepared for the next forward computation.

Backpropagation is a principle used to compute the gradient of the loss function (calculates the cost associated with a given state) with respect to the weights in a CNN. The weight updates of backpropagation are usually performed through the stochastic gradient descent.