Universidade de Aveiro Departamento de Engenharia Mecânica
2016

Luís Miguel Martins Almeida

Interação Humano-Robô para a Transferência de Objetos

Human-Robot Interaction for Object Transfer

Universidade de Aveiro Departamento de Engenharia Mecânica
2016

Luís Miguel Martins Almeida

Interação Humano-Robô para a Transferência de Objetos

Human-Robot Interaction for Object Transfer

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob a orientação científica de Vítor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro e sob co-orientação de Filipe Miguel Teixeira Pereira da Silva, Professor Auxiliar do Departamento de Eletrónica Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor Jorge Augusto Fernandes Ferreira
Professor Auxiliar da Universidade de Aveiro

vogais / committee

Prof. Doutor João Paulo Morais Ferreira
Professor Adjunto do Instituto Superior de Engenharia de Coimbra (arguente)

Prof. Doutor Vítor Manuel Ferreira dos Santos
Professor Associado da Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Agradeço em primeiro lugar ao Professor Vítor Santos pela orientação, motivação e por todo o apoio dado na realização deste trabalho.
Agradeço à minha família pelo apoio e ajuda incondicional nesta jornada.
Não menos importante, agradeço também aos meus amigos por todos os momentos inesquecíveis que passámos juntos.

palavras-chave

Interação Humano-Robô, sensor 3D, sensor de força, pré-interação, interação por contacto, transferência, braço robótico, ROS.

resumo

Os robôs entram em contacto físico com os humanos sob uma variedade de circunstâncias para realizar trabalho útil. Esta dissertação tem como objetivo o desenvolvimento de uma solução que permita um caso simples de interação física humano-robô, uma tarefa de transferência de objetos. Inicialmente, este trabalho apresenta uma revisão da pesquisa corrente na área da interação humano-robô, onde duas abordagens são distinguíveis, mas simultaneamente necessárias: uma aproximação pré-contacto e uma interação pós-contacto. Seguindo esta linha de pensamento, para atingir os objetivos propostos, esta dissertação procura dar resposta a três grandes problemas: (1) O controlo do robô para que este desempenhe os movimentos inerentes à tarefa de transferência, (2) a pré-interação humano-robô e (3) a interação por contacto. As capacidades de um sensor 3D e de sensores de força são exploradas com o objetivo de preparar o robô para a transferência e de controlar as ações da garra robótica, correspondentemente. O desenvolvimento de arquitetura software é suportado pela estrutura Robot Operating System (ROS). Finalmente, alguns testes experimentais são realizados para validar as soluções propostas e para avaliar o desempenho do sistema. Uma possível transferência de objetos é alcançada, mesmo que sejam necessários alguns refinamentos, melhorias e extensões para melhorar o desempenho e abrangência da solução.

keywords

Human-Robot Interaction, 3D sensor, force sensor, pre-interaction, interaction by contact, handover, robotic arm, ROS.

abstract

Robots come into physical contact with humans under a variety of circumstances to perform useful work. This thesis has the ambitious aim of contriving a solution that leads to a simple case of physical human-robot interaction, an object transfer task. Firstly, this work presents a review of the current research within the field of Human-Robot Interaction, where two approaches are distinguished, but simultaneously required: a pre-contact approximation and an interaction by contact. Further, to achieve the proposed objectives, this dissertation addresses a possible answer to three major problems: (1) The robot control to perform the inherent movements of the transfer assignment, (2) the human-robot pre interaction and (3) the interaction by contact. The capabilities of a 3D sensor and force/tactile sensors are explored in order to prepare the robot to handover an object and to control the robot gripper actions, correspondingly. The complete software development is supported by the Robot Operating System (ROS) framework. Finally, some experimental tests are conducted to validate the proposed solutions and to evaluate the system's performance. A possible transfer task is achieved, even if some refinements, improvements and extensions are required to improve the solution performance and range.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

Research in robotics has been playing a relevant role in Human-Robot Interaction (HRI) field. Robots are designed and built to complement human abilities. In countless future joint-action scenarios, humans and robots will have to interact physically in order to profitably collaborate. Ideally, this interaction should be intuitively simple for humans, yet it requires some degree of learning and adaptation [1]. In many operations, it is desirable to exploit the force capabilities of robots by directly combining them with the human skills, hence leading to human enhancement. This challenge requires a great perception of human environment and intentions in order to respond to them intuitively, competently and harmlessly [2]. The transfer of objects between humans and robots is a major way to coordinate activity and cooperatively perform advantageous work [3]. The task of handing an object to a robot presents a unique set of challenges for fluent human-robot interaction [4].

This dissertation aims to design and evaluate robotic systems for use by or with humans, involving technical and scientific knowledge in numerous areas of engineering such as robotics, mechanics, electronics, control and programming. This work considers an object transfer task between a robotic manipulator and a human collaborator.

1.2 Problem Formulation and Approach

This work requires both the human and the robot to be aware of the handover situation. The robot is considered to be static and waiting, in a favorable position, so the user can give an object to it or take an object from it. Firstly, a robotic arm motion control is necessary to enable the robot to perform the inherent movements of the transfer assignment. Secondly, a visual servoing system is required to enable the robot to adaptively approach the human hand, using common vision functions for robotic applications, such as object detection and human tracking. Furthermore, a contact and force-based interaction system has to be implemented to empower the robot to grasp and also release the object based on the force/tactile sensors feedback.

1.3 Objectives

The main objective of this dissertation is the interaction between a human and a robot with the specific purpose to transfer objects from one to another. In order to reach this purpose, this work is organized in the following major sub-goals:

1. Hardware and software infrastructure installation and familiarization required to perceive the interaction between a robotic arm and a human being.

2. Study of management mechanisms of pre-contact approximation between a robotic arm and a human arm using 3D vision.

3. Study and development of interaction mechanisms by contact between a robotic manipulator and a human being, with the concrete goal of object handover between human hand and robot gripper.

1.4 Dissertation Structure

This dissertation is divided in seven chapters, including the present chapter. Chapter 2 presents a review of modern developments connected to this work, such as HRI approaches, robotic manipulators acting in Human environments and object transfer methodologies. Chapter 3 provides an overview of the experimental set-up, describing some specifications of the Cyton Gamma 1500, the 3D sensor and the force sensors used for the proposed task. Additionally, the required software tools are detailed, and a view of the proposed overall system architecture is addressed. Chapter 4 presents the required steps to execute the robot motion control. The Cyton Gamma 1500 kinematics is also analysed and evaluated in this chapter. Chapter 5 is dedicated to the study and analysis of vision system for human hand tracking, as well as object detection. Chapter 6 describes the force-based controller development, implementation and evaluation. Additionally, a brief overview of the final overall system and ROS architecture is presented. Finally, in chapter 7 the conclusions and future work are presented.

Chapter 2

State-of-the-Art

This chapter introduces a brief study of modern developments related to this work, such as Human-Robot Interaction (HRI) approaches, robotic manipulators acting in Human environments and object transfer methodologies. Understanding current anthropomorphic robots that usually work in complex environments, side by side with Humans, it is really helpful to gather general information that leads to a better solution for the stated problem.

2.1 Human-Robot Interaction

"The HRI problem is to understand and shape the interactions between one or more humans and one or more robots" [5]

To promote HRI with the goal to transfer objects, it is important to study two different approaches: pre-contact and post-contact interaction. Pre-contact is based on 3D vision systems, with the intention of detecting any target in the surrounding environment (e.g., people, objects...), and then adjust robot End-Effector (EE) to further the contact. Contact itself requires hardware that allows the robot to "feel" the contact/touch.

Like humans, robots typically use the sensory information linked to the senses of vision and touch in order to interact with the objects in their environment. The sense of vision is usually implemented in robotics through cameras, and the sense of touch is achieved with force and tactile sensors. Vision grants the global information which is needed to locate the objects in the environment and compute their relative spatial relations. Still, touch provides the local information which is required to characterize the way the robot contacts the objects. By mixing the global information with the robot controller, it is possible to avoid undesired obstacles or to reach the target objects. On the other hand, local information can be used to manipulate the contacting object or to explore it so that its surface properties are extracted for its identification [6].

2.1.1 Pre-Contact Systems

Commonly in robotics, a motion capture system is used to perceive the pre-contact approximation between a robot and his surroundings. This system consists of a mechanism that is able to measure the position and orientation of an object in the environment. The well-known and most used systems to capture human motion are: mechanical, magnetic and optical systems [7]. Opposite to mechanical and magnetic systems that usually require the target to have special equipment attached to its structure, the optical system can allow the capture of motion done completely via software. RGB-D sensors such as the Microsoft Kinect

or the Asus Xtion revolutionized the pre-contact research scene, since the sensor produces a dense 3D point cloud in a plausible quality at a low cost [8]. The visual information extracted from the sensor promotes the motion of the robot, combining techniques such as image processing, computer vision and control theory [6].

2.1.2 Contact Systems

To perceive the interaction by contact a force system is required. Usually the force system uses the wrist force/torque sensor and/or tactile sensors or load cells at the gripper fingertips as the feedback devices [9], [10], [11]. These are commonly used to enable robots to perform numerous and diversified tasks, such as object manipulation and object handover. Additionally, sensors are the key to allow a robot to act safely while interacting with humans, which is the first concern in this matter.

2.2 Robotic Manipulators

In this section the most relevant assistive robots for the purpose of this dissertation, such as PR2, Domo and NAO robot are outlined. These manipulators are used in many studies involving the collaboration between Humans and Robots. The main purpose here is to review these works by highlighting the relevant characteristics and technologies linked to transference.

2.2.1 PR2

The PR2, presented in Figure 2.1, is a two-armed wheeled robot of size similar to a human, developed by Willow Garage, designed for both navigation and manipulation. This robot combines the mobility to navigate human environments and the dexterity to grasp and manipulate objects in those environments. It has two 7 DOF arms with a payload of 1.8 kg. PR2 has a head-mounted Kinect, that is used in conjunction with the Point Cloud Library, increasing the abilities to sense the environment. The PR2 robot is fully integrated with ROS, providing the power of all the ROS developer tools and out-of-the-box functionality for everything from full system calibration to manipulation [12].

Jim Mainprice et al [13] studied the motion planning of handovers in cluttered workspaces using a PR2 robot. The tactile information acquired by the pressure-sensors installed in its fingertips helps them to discuss the problem of finding good object handover configurations.

Figure 2.1: PR2 Robot [14].

2.2.2 Domo

Domo, pictured in Figure 2.2, is an experimental robot made by MIT designed to interact with humans. It has a total of 29 active degrees of freedom (DOF), of which 22 DOF use force controlled and compliant series elastic actuators, providing the robot with the proprioceptive sense of joint torque. This allows the robot to safely act in several complex environments, being capable of many different grasps and movements. Also, thanks to the two cameras mounted on its head and the visual processing system, Domo is able to analyse the size and shape of an object to prepare for interaction [3].

Aaron Edsinger [15] developed an application that enables Domo to help a user to place objects on a shelf. Domo's hands modular force sensing compliant actuators allows it to gain force knowledge during the transfer task.

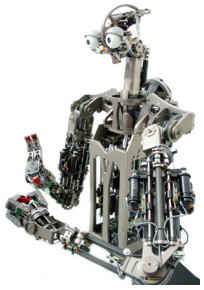


Figure 2.2: Humanoid robot Domo [16].

2.2.3 NAO

NAO Robot (illustrated in Figure 2.3), developed by Aldebaran Robotics, is a biped robot with 25 DOF with the ability to move, feel, hear and speak, see and connect to the internet. Its humanoid shape and inertial unit enables it to move and adapt to the world around maintaining the balance. It is equipped with two cameras that help recognize shapes and objects. The plentiful sensors and sonar installed enables it to perceive the environment and get its demeanour [17].

Judith Müller et al [18] presented a object manipulation system for the NAO robot that is capable of grasp and/or give an object to a human. To grasp an object, NAO tries to fully close its hand. The difference between the expected and the actual finger position indicates if the object has been grasped. In a object releasing task, unlike other robots, NAO does not use force sensors in its fingers to detect the traction force. By reducing the stiffness of the arm joints, it is possible to detect this force by an unexpected motion of the arm joints.

Figure 2.3: Humanoid robot NAO [19].

2.3 Related Work

HRI for object transfer solutions have been proposed over the years. Papanikolopoulos and Khosla [20] studied the task of a human handing an object to a robot. The experimental results show that human subjects naturally control the objects orientation and position, without any special preparation, to match the robots gripper configuration. The human instinctively adapts to simplify the task of the robot. Shibata et al [21] conducted a research on cooperative object handovers by studying the trajectories of the hands of the human collaborators. Maya Cakmak [22] implemented a hand-over action in three phases: approach, signal and release. In the first phase, the robot navigates towards the receiver, in the second, it makes a signal meaning that it is ready to hand-over. The release phase refers to the moment when it senses the object being pulled. Maya Cakmak [22] also studied different hand-over configurations so that robot can choose the one that fits humans best. The robot should try to transfer objects in the default orientation (object orientation commonly viewed in everyday environments), without compromising safety, visibility and comfort. Sato et al [23] have proposed a robot system capable of imitating human task trajectories by making use of human data for several daily object transfer tasks. Nagata et al [24] have studied the task of object handovers using contact point information by using the force torque data on a multi-fingered robot hand. They described a robust adaptive grasping strategy. Patrizia Basili [25] investigated two experimental set-ups in order to define a better human approach criteria. The object transfer mostly occurs at the midpoint between both subjects and the object is lifted approximately 1.2 s before the actual hand-over.

2.4 Overview

To successfully achieve the major goal of this dissertation, some gains from this state of art study have to be considered. It is important to adapt and ease the robot work, performing a large collaboration and adaptation while transferring the objects with it. The task has to profitably accomplish two dominant phases: pre-interaction employing vision systems and post-contact applying force-based interaction systems. The capabilities of a 3D sensor have to be explored in order to prepare the robot to handover an object. Relatively to physical interaction, many concepts must be taken in account (e.g. user safety, easy-to-understand and design, robot performance and limitations, and force hardware capabilities.). To significantly improve the performance of human-robot object handover task, gripper orientations can be pre-defined. In addition, a signal or order must be implemented to dictate when to enter in the transfer state. Normally, to transfer objects between a human and a robot by contact, the robotic arm is installed with tactile sensors in its gripper fingers and/or a force sensor in its last joint, usually the wrist joint. For this work to be possible, the diversity of objects to be transferred have to be limited and appropriate to use with the available hardware.

Chapter 3

Experimental Setup

This chapter presents the hardware and software technologies required to achieve the proposed goals of this dissertation. It is of crucial importance to know and understand available technologies and means, so it becomes possible to take full advantage of them throughout their implementation. The main hardware components consist of an anthropomorphic robotic arm (Figure 3.1), a Kinect sensor, an ATI mini40 transducer, two FSR and a central processing unit (PC-based). The software architecture is based on the Indigo version of Robot Operating System (ROS) framework under Linux, using C/C++ and python programming environment. ROS provides a wide list of libraries and tools to support the development of new robot applications. Few tools, such as MoveIt!, KDL and OpenNI, are briefly explained, since they are necessary to detect what is happening in the environment and how the robot is moving so that the robot's behaviour can be adapted accordingly [6]. Additionally, a high level proposed system architecture is illustrated to clarify how both hardware and software tools communicate and interact with each other.

Figure 3.1: Cyton Gamma 1500 manipulator.

3.1 Cyton Manipulator Arm

Introduced by Robai corporation, cyton gamma 1500 offers increased joint torques compared to other models. Cyton robot has 7 degrees of freedom and also a servo to control the gripper. With more than six axes, this robot is considered kinematically redundant, allowing it to place the end effector at a position and orientation in a limitless number of ways. Al-

though it makes the kinematics of the arm more difficult, it enables the arm to reach around obstacles with a fluid motion and higher accuracy. Figure 3.2 shows a representation of the cyton gamma 1500 joints and also its dimensions.

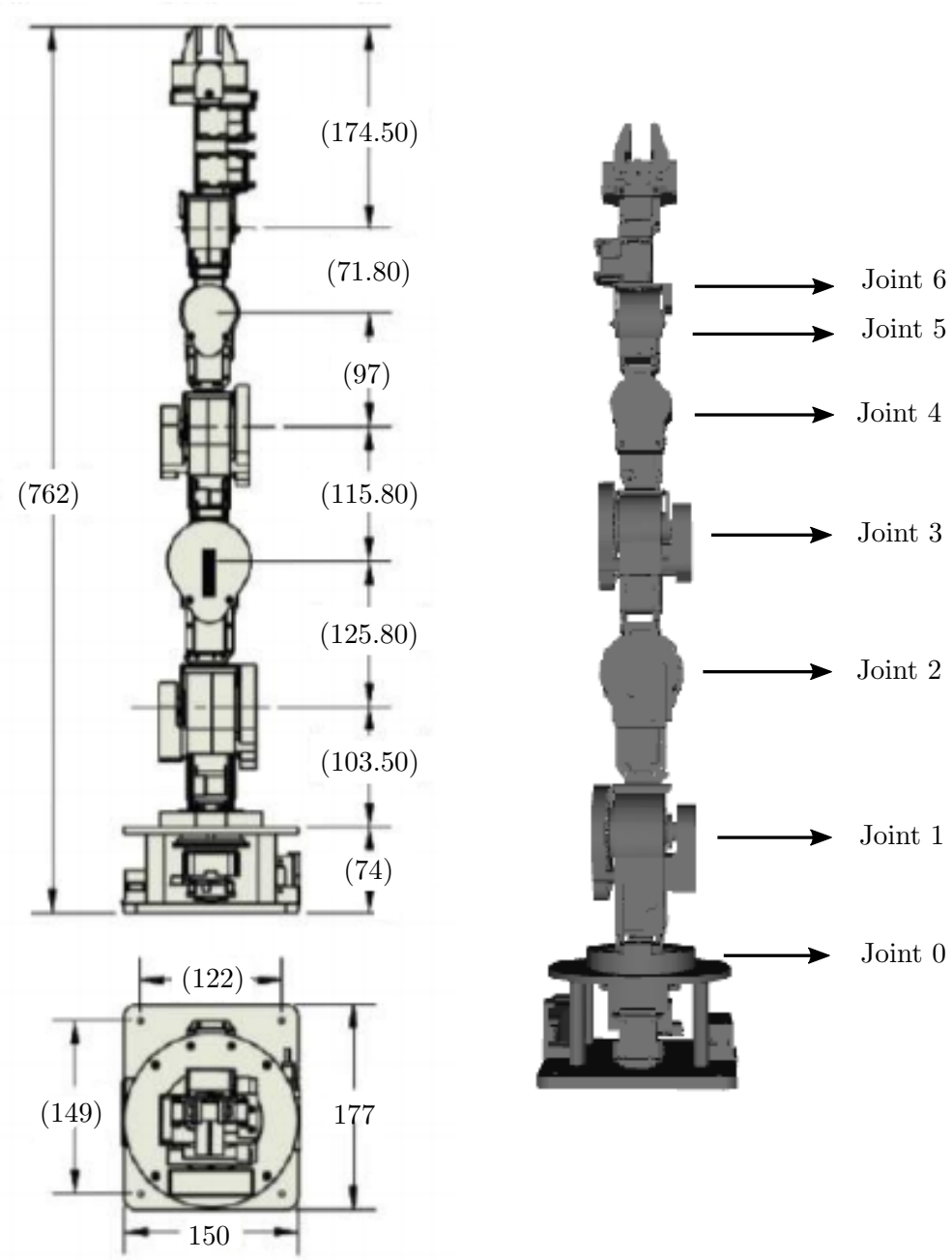


Figure 3.2: Cyton Gamma 1500 dimensions[1].

Table 3.1 describes some technical specifications of the cyton gamma 1500 manipulator [26]. The specifications are mainly related to robot physical properties, performance and control.

[1]Image adapted from `http://www.robai.com/assets/Cyton-Gamma-1500-Arm-Specifications_2015.pdf`

Table 3.1: Cyton Gamma 1500 general technical specifications.

Specifications	
Total weight	3 Kg
Payload at full range	1200 g
Reach	68 cm
Maximum linear speed	45 cm/sec
Maximum speed (free move)	70 cm/sec
Repeatability	±0.5 mm
Gripper opening range	3.5 cm
Input voltage	100-240 V AC
Control interface	USB or RS485

Table 3.2 describes joint type, angle and velocity limits, as well as the correspondent servo model [26].

Table 3.2: Cyton Gamma 1500 joint specifications.

Joint Name	**Joint type**	**Angle limits (degrees)**	**Velocity limits (degrees/s)**	**Servo model**
Shoulder roll (Joint 0)	Spin	-150 to 150	75	MX-64
Shoulder pitch (Joint 1)	Articulate	-105 to 105	75	MX-64
Shoulder yaw (Joint 2)	Articulate	-105 to 105	75	MX-64
Elbow pitch (Joint 3)	Articulate	-105 to 105	65	MX-28
Wrist yaw (Joint 4)	Articulate	-105 to 105	110	MX-28
Wrist pitch (Joint 5)	Articulate	-105 to 105	330	MX-28
Wrist roll (Joint 6)	Spin	-150 to 150	330	MX-28

3.2 Servo Motors

The servo motors and their controllers are responsible for all robot movements. Cyton gamma 1500 has two different servo models developed by Dynamixel, MX-64 (Figure 3.3a) and MX-28 (Figure 3.3b).

(a) MX-64[2]. (b) MX-28[3].

Figure 3.3: Cyton servo motors models.

These servos are connected using a daisy chain mechanism and are queried and controlled with an advanced serial protocol. Over 50 read/write parameters are available including position, temperature, load, input voltage feedback and PID tuning. The dynamixel MX series servos feature a contactless (Magnetic) position encoder and PID control for superior accuracy and reliability. It also offers 12bit (4096) position resolution and can operate in the 360 degrees range with 0.088° resolution [27]. The servos specifications are described in table 3.3.

Table 3.3: Dynamixel servos specifications.

Model	**Gear Ratio**	**Network Interface**	**Position Sensor**	**Stall Torque (N.m)**
MX-64	200:1	TTL / RS-485	Contactless Absolute Encoder (12bit, 360 degrees)	6.0 (at 12V, 4.1A)
MX-28	193:1			2.5 (at 12V, 1.4A)

Figure 3.4 presents an overall PID structure of dynamixel controller. Equation 3.1 shows how the proportional, integral and derivative gain are calculated correspondingly.

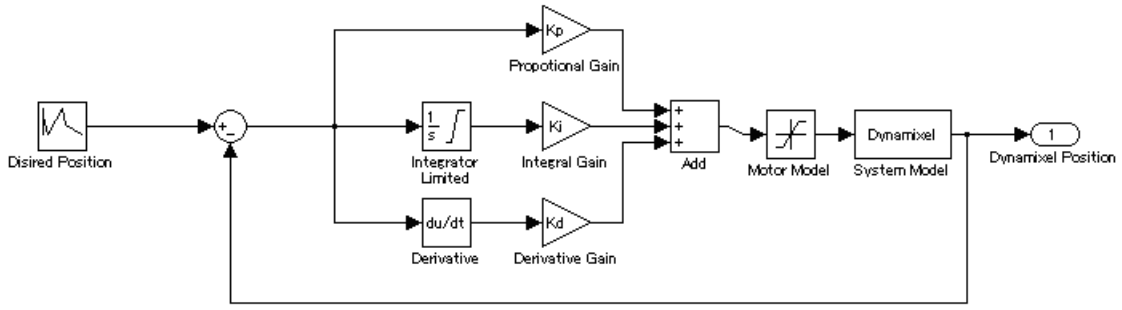


Figure 3.4: Dynamixel servos PID control structure [28].

$$K_p = \frac{PGain}{8}; \quad K_i = \frac{IGain \times 1000}{2048}; \quad K_d = \frac{DGain \times 4}{1000} \tag{3.1}$$

3.3 Kinect Sensor

The Kinect sensor, shown in Figure 3.5, is a composite device consisting of a colour camera (RGB) and a depth sensor which contains an infrared (IR) projector and an IR camera.

[2] http://support.robotis.com/en/product/dynamixel/mx_series/mx-64.htm
[3] http://support.robotis.com/en/product/dynamixel/mx_series/mx-28.htm

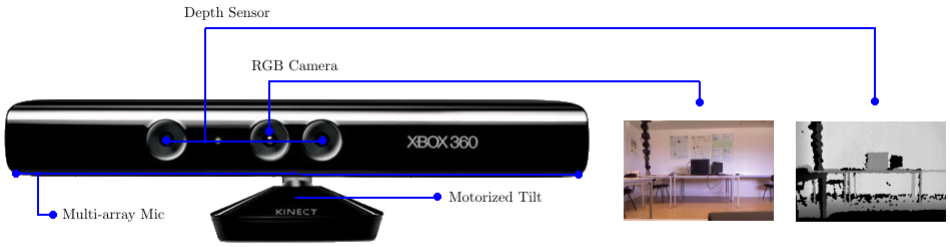


Figure 3.5: Microsoft Kinect Xbox 360.

The estimation of depth is based on an internal triangulation process [8]. The IR projector casts an IR speckle dot pattern into the 3D scene while the IR camera captures the reflected IR speckles. The geometric relation between the IR projector and the IR camera is obtained through an offline calibration procedure. The depth of a point can be deduced from the relative left-right translation of the dot pattern of projected dots. This translation changes, depending on the distance of the object to the camera-projector plane [29]. Figure 3.6 illustrates this procedure.

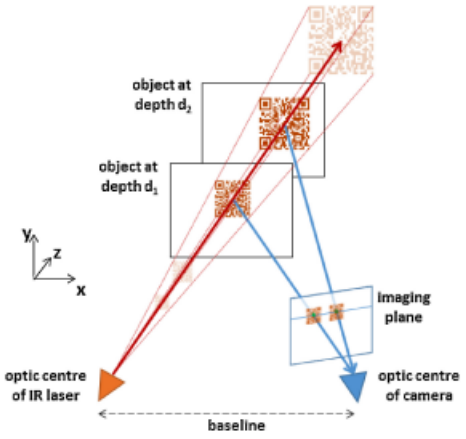


Figure 3.6: Illustration of Kinect depth measurement [29].

The main nominal specifications are shown in Table 3.4.

Table 3.4: Microsoft Kinect 3D-depth sensor

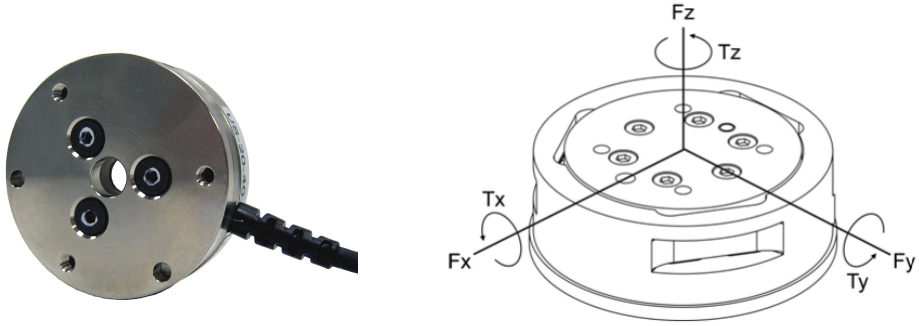
Property	**Value**
Field of Application	Indoor
Field of View	57° horz.. 43° vert
Frame rate	30 Hz
Resolution, colour stream	VGA (640x480)
Resolution, depth stream	QVGA (320x240)
Nominal depth range	0.8 m - 3.5 m
Mechanized tilt range	±28°
Interface	USB

3.4 Force Sensors

Force sensors work as communication media of force interaction between robots and humans or environment. In this section the sensors used to gather force contact between one or more objects and one robot arm are described.

3.4.1 ATI Mini 40

ATI mini40 transducer (shown in Figure 3.7a) has a compact, low profile design and is typically applied to robotic-hand research. This force and torque (F/T) sensor uses silicon strain gages to sense forces. The transducer's silicon strain gages provide high noise immunity and allow high overload protection [30].



(a) ATI Mini40 transducer. (b) Applied force and torque vector.

Figure 3.7: F/T sensor.

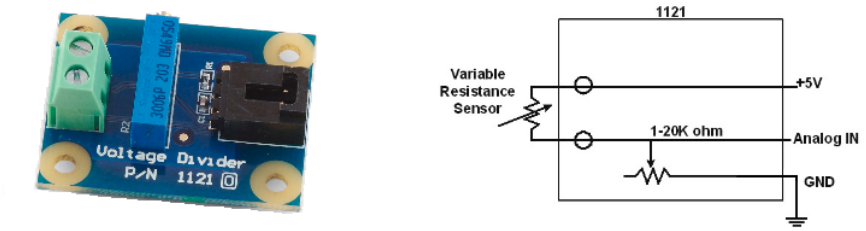
This device measures the outputting forces and torques in 3 directions around three axes, as can be seen in Figure 3.7b. Table 3.5 presents the analog output range and sensitivity of mini40 transducer within the operating range of ±10V.

Table 3.5: Analog output range and sensitivity of ATI mini40 transducer.

Analog Output Range			**Analog Sensitivity**		
Fx,Fy (N)	Fz (N)	Tx,Ty,Tz (N.m)	Fx,Fy (N)	Fz (N)	Tx,Ty,Tz (N.m)
±80	±240	±4	8	24	0.4

3.4.2 Force-Sensing Resistor

Force-sensing resistors (FSR) consist of a conductive polymer, thin and flat which changes resistance in a predictable manner subsequent of force or pressure applied to its surface, taking into account that increasing the force results in a lower resistance [31]. This variation is caused by non-conducting particles touching the conducting electrodes of the film. These sensors are relatively easy to use and inexpensive, however, their low precision leads to a measurement error up to 25% [31]. Commonly, a voltage divider (Figure 3.8a) is implemented to use an FSR's output. This simple circuit (Figure 3.8b) turns a large voltage into a smaller one using only 2 series resistors and an input voltage.



(a) FSR adapter (voltage divider). (b) FSR adapter circuit scheme.

Figure 3.8: FSR adapter [32].

In this work two FSR 402 short (Interlink Electronics, Figure 3.9) are implemented. This device has a force sensitivity range of ∼0.2 - 20N with a continuous (analog) force resolution.

Figure 3.9: FSR 402 Short.

3.5 Software Development Tools

In this section the main software tools used throughout this work are highlighted. Firstly, ROS is briefly introduced emphasizing its packages. Secondly, tools like MoveIt! and KDL are referred, given their importance to robot inverse kinematics calculations. Additionally, the operability of OpenNI package is explored, since this has a great impact on the pre-contact approach.

3.5.1 Robot Operating System

The ROS framework is the unifying element of this dissertation. It has revolutionized the developers community, providing it with a set of tools, infrastructures and best practices to build new applications and robots. A key pillar of the ROS effort is the notion of not re-inventing the wheel by providing easy to use libraries for different capabilities like navigation, manipulation, control (and more) [33]. The core software is provided and maintained by Willow Garage and the open-source community. In the matter of this work it defines the interaction between the Cyton Gamma 1500 manipulator, the Kinect sensor and two Force Sensitive resistors, using the functionality already present on the ROS platform and other software personally developed.

ROS Communication

Communication is based on the TCP/IP protocol with each node connecting to a socket. The server is administrated by a master that handles all the connection and addressing details. To better understand ROS communication, there are a few crucial concepts that the user must know: packages, nodes, master, messages, topics and services [34].

Packages : Software in ROS is organized in packages. A package structure commonly contains stored source code, libraries, binaries, a "manifest.xml" file which is a declaration of the package's dependencies and also information about the package sustainer. Additionally, each package incorporates a CMakeList.txt file with instructions for standard CMake compilation.

Nodes : Processes that can perform computation, execute tasks and communicate. An executed node has a unique id and a list of topics and services that are used to send or receive messages, as well as some additional connection parameters. Nodes can be written using C++ or Pyton languages provided by ROS libraries.

Master : A unique node that is launched when the network starts. It handles registrations, subscriptions and disconnections of every node. It also links all topics and/or services in order to make sure that all the messages reach their target successfully.

Messages : A message is simply a data structure containing typed fields. Although ROS has some standard types of fields (e.g. boolean, (un)signed int, float, string..), additional types can be found in many personal packages, resulting from combinations of these standard types.

Topics : Topics are used to send messages using a publish and subscribe semantic. A node can attach to a topic by its name, either as a publisher in order to send data or as a subscriber to receive these data. It is not possible to send different data kind from the one defined in the message template.

Services : Uses a request/response model. Services can only be advertised by one node, and receives data in response to the query it made.

ROS Visualization Tool (RVIZ)

Rviz is a 3D visualizer for displaying sensor data and state information from ROS. It is feasible to use this tool to simulate robot motion and compare those with the real robot movements. With the information extracted from RVIZ, it is possible to figure out if something went wrong.

3.5.2 MoveIt!

MoveIt! is an agnostic robot software for mobile manipulation, incorporating advances in motion planning, manipulation, 3D perception, kinematics, control and navigation. It provides an easy-to-use platform for developing advanced robotics applications [35]. Given the robot arm complexity, this tool, connected with kinematics and dynamics library (KDL) is used in this work to solve the inverse kinematics of the cyton gamma 1500.

3.5.3 KDL

The kinematics and dynamics library (KDL) is a numerical method based solver. It develops an application independent framework for modelling and computation of kinematic chains, such as robots and machine tools. It provides class libraries for kinematic chains, and also their motion specification and interpolation [36].

3.5.4 OpenNI

Open Natural Interaction (OpenNI) is an industry-led non-profit organization and open source software project focused on certifying and improving interoperability of natural user interfaces and organic user interfaces for Natural Interaction (NI) devices, applications that use those devices and middleware that facilitates access and use of such devices [37].

OpenNI Tracker

OpenNI tracker allows a user to track his own or others skeleton using a 3D sensor [38]. It provides the position, relative to the camera frame, of 15 human joints (illustrated in Figure 3.10, such as head, torso, elbows, hands and knees. This tracker combined with the kinect sensor is used in this work to track the user hand. The tracked coordinates are published as a set of transforms and used to send the robot to a place closer to the user's hand to promote the transference between them. The process to detect a user skeleton is fully automatic: it only requires the user to be standing in front of the camera with the majority of his body detectable.

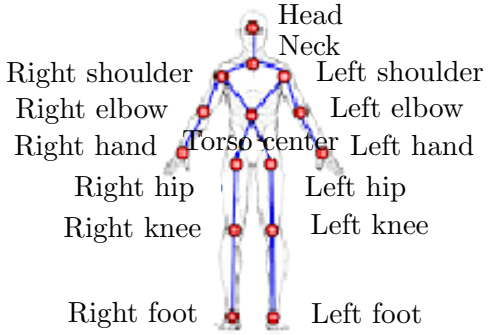


Figure 3.10: OpenNI joint map [39].

3.6 Proposed System Architecture

The study and knowledge of the available resources allow, at this stage, the development of a proposed system architecture. The proposed system architecture presents a succinct integration of the hardware and software required for this dissertation. This architecture, at hardware levels, contains a 3D sensor, a robotic manipulator and one or more force/tactile sensors. The software architecture is based on the Indigo version of the ROS framework under ubuntu 14.04. The necessary packages to reach the objectives use C/C++ and python programming languages. Figure 3.11 illustrates the proposed integrated system architecture.

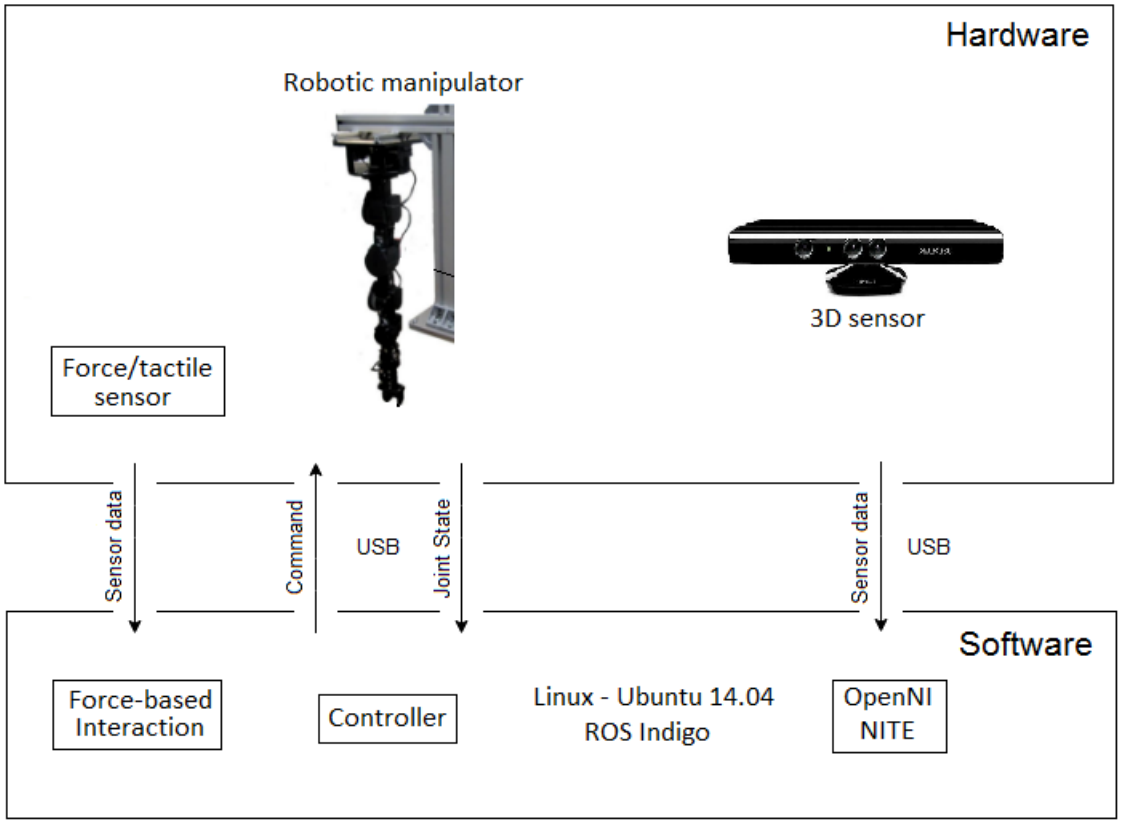


Figure 3.11: Proposed integrated system architecture.

Chapter 4

Robot Control

This chapter describes the arm motion control. Its main goal is to explore the 7DOF manipulator arm in terms of forward and inverse kinematics in synchronous mode. Given the complexity of this manipulator, the arm motion control solution is not found analytically. The method implemented creates a 3D robot model for Cyton gamma 1500 arm, and then creates a configuration file to integrate cyton with MoveIt!. To find the inverse kinematics solution a numerical solver that handles the process of finding joint values for a given end-effector position and orientation (KDL) and MoveIt! are used. The chapter starts by presenting how Cyton gamma 1500 arm joints are controlled.

4.1 Joint Controller

In this work, the Cyton gamma 1500 joints controller is fully based on the `dynamixel_motor` package. The ROS stack `dynamixel_motor` contains packages that are used to interface with the Robotis Dynamixel line of servo motors. This stack contains 3 main packages: `dynamixel_driver`, `dynamixel_controllers` and `dynamixel_msgs`.

The driver package of Robotis Dynamixel servos, `dynamixel_driver`, provides low level IO (Input/Output) communication between motors and PC. As a low level package, ROS users don't usually use this package directly, unless there is a need to re-write motors specifications, like rotation limits (CW and CCW angles), ID, baud rate, to name a few. Higher level specific robot joint controllers, as well as `dynamixel_controllers`, make use of this package.

`dynamixel_controllers` package works parallely with `dynamixel_driver` package and it is used to control every single Cyton servo. It incorporates a node that can be configured, services and a spawner script to start, stop and restart one or more controller plugins. It is possible to set the speed and torque for each motor making use of the available services inside this package. All configurable parameters can be loaded via a YAML[1]file.

`dynamixel_msg` package contains the messages that are used throughout this stack. In this work, these dynamixel messages are used to set goals for each joint and also receive feedback from them.

By making use of this stack, the developed package launches and starts controlling all

[1]Human-readable data serialization language.

Cyton arm servos. Listing 4.1 and 4.2 illustrate a generic structure for roslaunch files to launch and start the controll of cyton dynamixel servos.

Listing 4.1: Roslaunch structure to launch the Cyton servos.

```
<!-- -*- mode: XML -*- -->
<launch>
    <node name="dynamixel_manager" pkg="dynamixel_controllers" type="
       ↪ controller_manager.py" required="true" output="screen">
        <rosparam>
            namespace: dxl_manager
            serial_ports:
                pan_tilt_port:
                    port_name: "/dev/ttyUSB0"
                    baud_rate: 1000000
                    min_motor_id: 0
                    max_motor_id: 24
                    update_rate: 20
        </rosparam>
    </node>
</launch>
```

Listing 4.2: Roslaunch structure to start the Cyton servos.

```
<!-- -*- mode: XML -*- -->
<launch>
    <!-- Start tilt joint controller -->
    <rosparam file="$(find my_dynamixel_tutorial)/tilt.yaml" command="
       ↪ load"/>
    <node name="tilt_controller_spawner" pkg="dynamixel_controllers"
       ↪ type="controller_spawner.py"
          args="--manager=dxl_manager
                --port pan_tilt_port
                shoulder_roll_controller
                shoulder_pitch_controller
                shoulder_yaw_controller
                elbow_pitch_controller
                elbow_yaw_controller
                wrist_pitch_controller
                wrist_roll_controller
                gripper_controller"
          output="screen"/>
</launch>
```

4.2 Robotic Arm Description in ROS

ROS has a meta package called `robot_model`, which contains several relevant packages that help build the 3D robot model. This section describes the most crucial packages used to create a 3D robot model for Cyton gamma 1500. This 3D robot model is used with Moveit! to find kinematics solutions and it is uploaded to the ROS parameter server using the launch utility. All the important packages inside `robot_model` meta package that are used to build the Cyton 3D model are as follows [40]:

URDF: The Unified Robot Description Format (URDF) package contains a C++ parser, which is an XML format for representing a robot model. This XML contains specifications for

robot model state, link, joint, transmission, sensors, scenes and other required specifications.

joint_state_publisher: This node, inside the URDF package, reads the robot model description, finds all joints, and publishes joint values to all non-fixed joints using graphical user interface (GUI) sliders.

kdl_parser: Kinematic and Dynamics Library (KDL) is a ROS package that contains parsing tools to build a KDL tree from the URDF description. The kinematic tree is used to publish the joint states and also to get the forward and inverse kinematics of the robot.

robot_state_publisher: Robot state publisher reads the current robot joint states and publishes the 3D poses of each robot link using the kinematics tree previously built from URDF. Each pose is published as ROS tf (transform), which is the relationship amid coordinate frames of a robot.

xacro: Xacro stands for XML Macros. It contains some add-ons to make URDF readable, briefer, and can be used for building complex robot descriptions. ROS has tools that allow to convert xacro to URDF.

By making use of these packages and the Cyton CAD model, it is possible to create the 3D Cyton model. Figure 4.1 shows the 3D cyton gamma 1500 arm model represented in rviz.

Figure 4.1: 3D Cyton gamma 1500 model in rviz.

4.3 MoveIt! Cyton Configuration

To use MoveIt! with the Cyton arm, it is necessary to generate compatible configuration files for MoveIt!, by using MoveIt! Setup Assistant. These configuration files are generated from the URDF file and, despite their several applications, in this work they are mainly used for kinematics calculations. The relevant files generated for this purpose are the following:

Controllers configuration file: List of Cyton arm controllers.

SRDF file: SRDF is the compact term for Semantic Robot Description Format. This format represents Cyton structure information that is not in the URDF file, and has a semantic aspect. SDRF is a complete description for everything from the world level down to the robot level.

Joint limits file: This file allows the dynamics properties specified in the URDF to be overwritten or augmented as needed, such as velocity and acceleration limits.

Joint names file: Specifies all cyton joint names to be used in MoveIt! for all calculations.

Kinematics file: Main configuration file for cyton kinematics. Here the kinematics solver plugin (KDL_plugin), solver search resolution, solver time-out and solver attempts are specified.

4.4 Kinematic Analysis

To control a robotic manipulator it is essential to know its kinematics. This section addresses the Forward Kinematics (FK) and Inverse Kinematics (IK) of the 7 DOFs robotic arm.

4.4.1 Forward Kinematics

The Direct Kinematics (DK) of a robotic arm is the determination of end-effector position and orientation as a function of the joint angles. Denavit and Hartenberg developed a convention to obtain the transformation matrix that represents the end-effector related to the global reference. From the DH parameters it is possible to determinate the Cartesian position of the end-effector in terms of joint angles $(\theta_1, \theta_2, ..., \theta_7)$. Figure 4.2 presents the coordinate frame for Cyton Gamma 1500 and table 4.1 presents the DH-parameters for Cyton Gamma 1500.

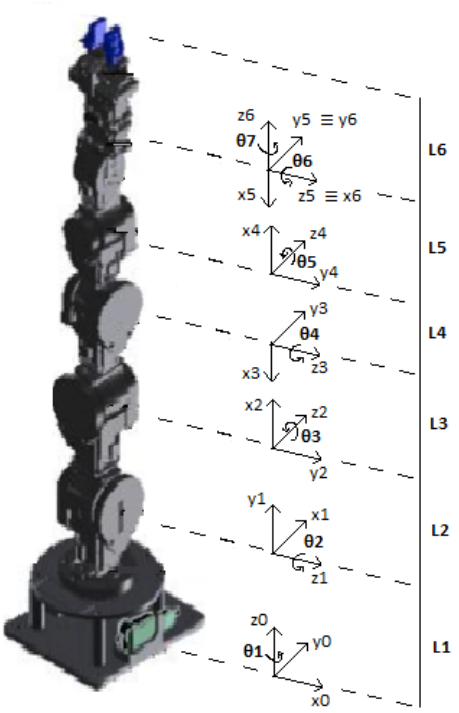


Figure 4.2: Coordinate frames of Cyton Gamma 1500.

Table 4.1: DH parameters of 7 DOF cyton arm.

Link	θ_i	L_i	d_i	α_i
0	$\theta_1 + \frac{\pi}{2}$	0	L_1	$\frac{\pi}{2}$
1	$\theta_2 + \frac{\pi}{2}$	L_2	0	$\frac{\pi}{2}$
2	$\theta_3 + \pi$	L_3	0	$\frac{\pi}{2}$
3	$\theta_4 + \pi$	-L_4	0	$\frac{\pi}{2}$
4	$\theta_5 + \pi$	L_5	0	$\frac{\pi}{2}$
5	$\theta_6 - \frac{\pi}{2}$	0	0	-$\frac{\pi}{2}$
6	θ_7	0	L_6	0

Synchronous Movement

To reduce Cyton's vibrations and overload problems it is necessary to have a synchronous movement. This synchronized method of joint control allows smoother motion and precise control, which provides better performance of the robotic arm. In order to have a synchronous movement, a node is created to subscribe the joint states. With these joint states ($j\alpha s$) and the new desired joint angles ($j\alpha d$), it is possible to calculate the velocity for each joint. To calculate these velocities, first, using the joint states and desired angles the displacement ($disp(i)$) is calculated. By mixing a pre-defined max velocity (ω_{pre}) for each joint and the maximum displacement ($disp_m$), it is possible to get the time of execution (t_e). Dividing each joint displacement by the time of execution, joints are now synchronized, meaning that they will start and finish at the same instant.

Being $[n] = \{1, 2, ..., 7\}$, the velocity for each joint ($\omega(i)$) can be calculated as expressed by the following equations (4.1, 4.2 and 4.3):

$$disp_m = \max_{i \in [n]} (|j\alpha d_i - j\alpha s_i|) \tag{4.1}$$

$$t_e = \frac{disp_m}{\omega_{pre}} \tag{4.2}$$

$$\omega(i) = \frac{disp(i)}{t_e} \tag{4.3}$$

Foward Kinematics ROS Structure

Figure 4.3 presents an overall ROS structure of forward kinematics for Cyton arm. Cyton_JS node subscribes all joint states and publishes them into the CJS_D topic. Cyton_FK node subscribes from this topic, and publishes the joint angles and the calculated velocities for each joint. Dynamixel_manager node is then responsible to make the joints move synchronously to the desired positions.

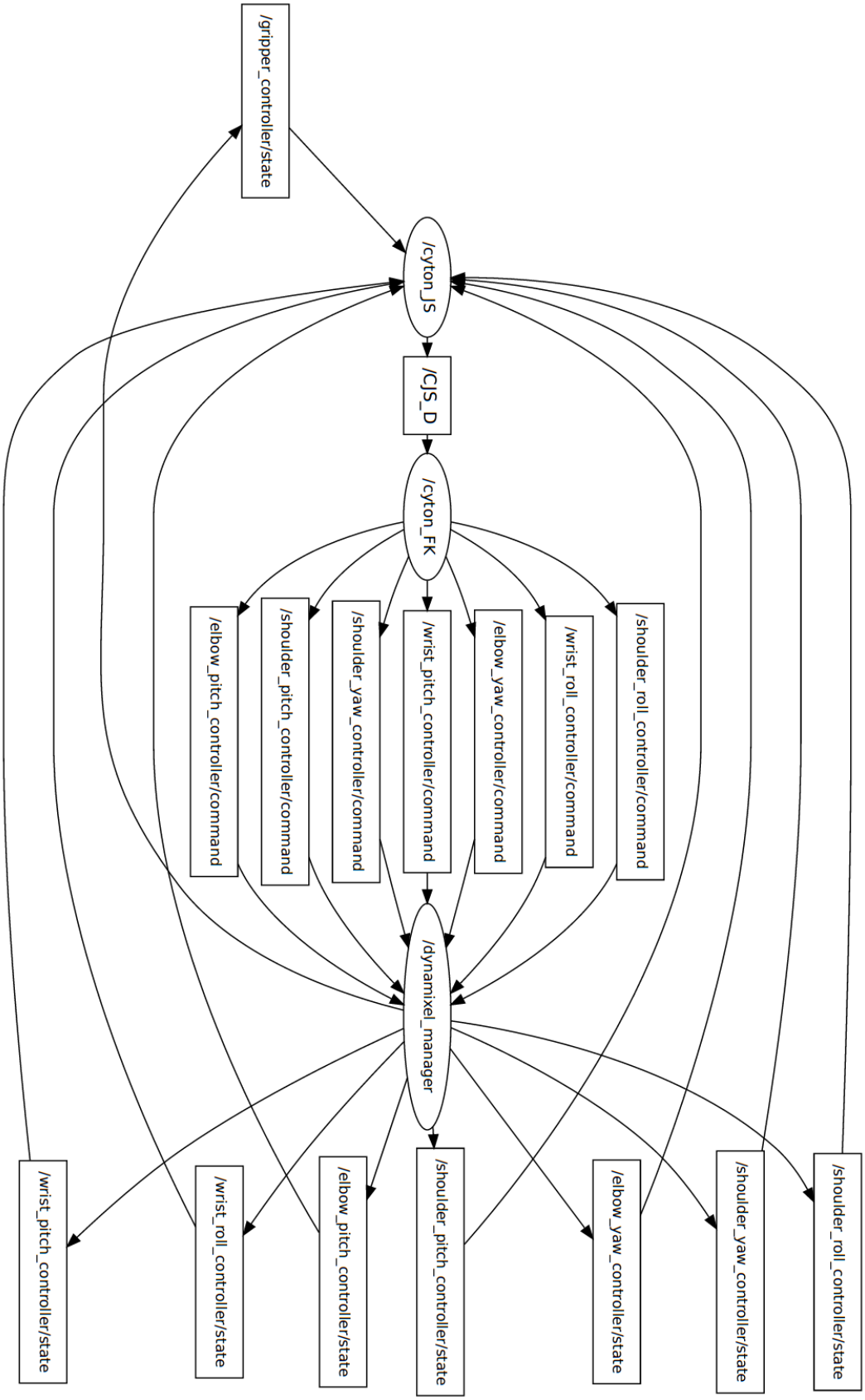


Figure 4.3: Overall ROS nodes and topics of Cyton foward kinematics calculation.

4.4.2 Inverse Kinematics

Inverse Kinematics (IK) is the method of determining a set of joint angles that will satisfy a given end-effector pose in the Euclidean space. Joint angle values are needed by the robot's motion controller in order to move the end- effector of a robot to a desired point in space, or through multiple points, in case of a trajectory. The IK implementation is based on the least squares solution. This method is a standard approach in regression analysis to the approximate solution of overdetermined systems. "Least squares" means that the overall solution minimizes the sum of the squares of the errors made in the results of every single equation.

Least Squares Solution

Consider an n-jointed kinematic chain. Each Cartesian coordinate is a function of each of the joint angles. This can be represented in matrix form by equation 4.4,

$$Y = F(X), \tag{4.4}$$

where Y represents the vector of Cartesian coordinates and X the vector of joint values. The partial derivatives of each Cartesian coordinate with respect to each joint angle lead, in matrix form, to the equation 4.5

$$\dot{Y} = J(X)\dot{X}, \tag{4.5}$$

where $J(X)$ is the Jacobean of F. The solution of this equation is that of a linear least squares problem that minimizes $\left\|\dot{Y} - J(X)\dot{X}\right\|$.

KDL Implementation

To implement the inverse kinematics for Cyton gamma 1500, Kinematics and Dynamics (KDL) library of the Open Robotics Control Software (OROCOS) is used. KDL solves the linear least squares problem described above. This is done over a number of iterations, using Newton-Raphson (NR) for gradient descent minimization. The pseudo-inverse of the Jacobian is determined using Singular Value Decomposition (SVD).

Since this task has a practical use, joint limits have to be taken into account. For joint limits, the solver simply assigns the limiting value to the joint whose value increases beyond its limit as a result of the inverse kinematic solution.

Inverse Kinematics ROS Structure

Figure 4.4 presents an overall ROS structure to calculate the inverse kinematics for Cyton arm. The node Cyton_JS subscribes the joints state and the desired goal for EE position, and publishes the information on a topic. The Cyton_IK node reads from that topic, calculates a solution for each joint, and publishes it into the topic IKCJ_D.

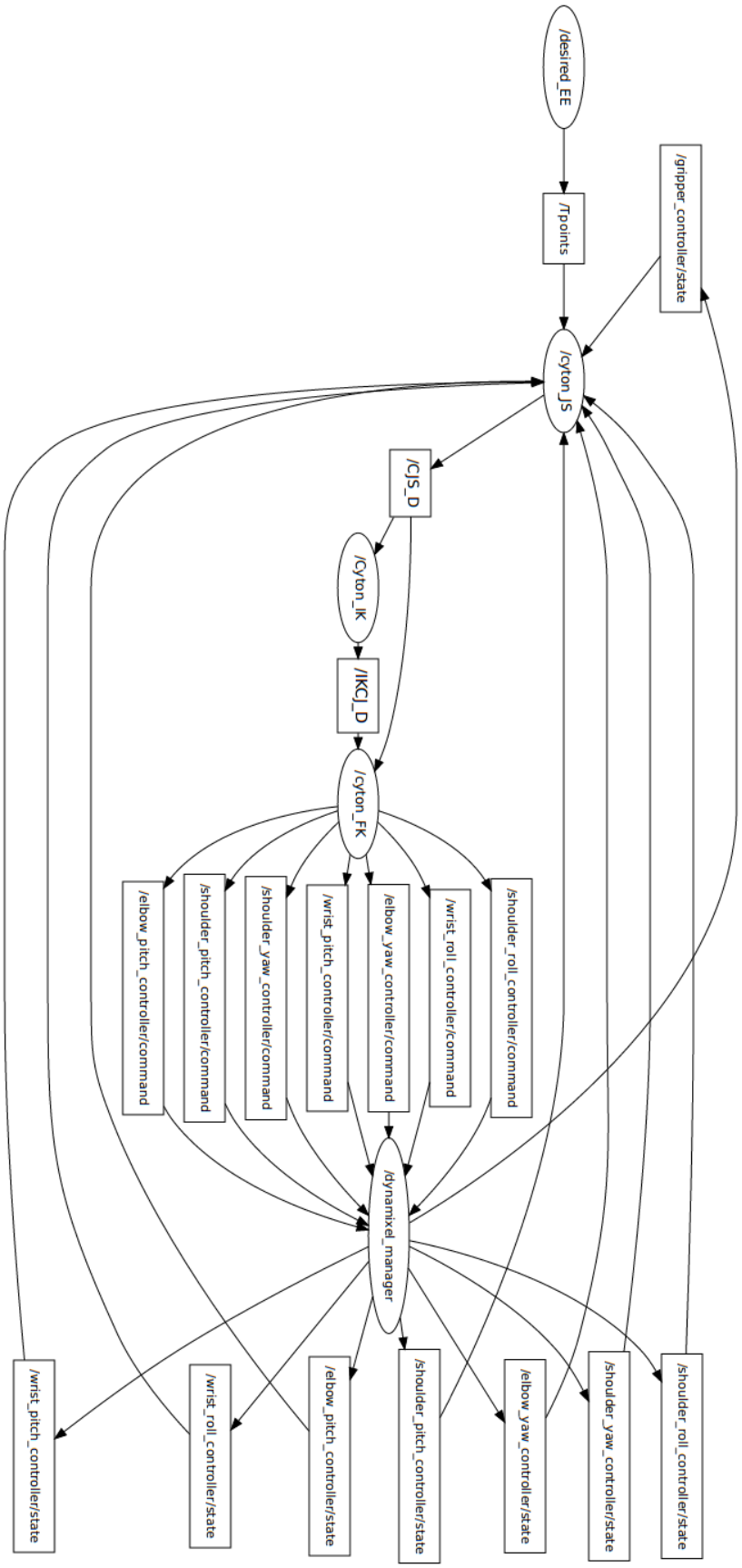


Figure 4.4: Overall ROS nodes and topics of Cyton inverse kinematics calculation.

4.4.3 Control Mode: Implementation

For this work, point-to-point or position control mode is used to control the arm motion. This mode is commonly used when the goal is to move the end-effector to a specified position regardless of the path, as intended for the object transfer task. The inverse kinematics algorithm implemented with MoveIt! calculates the desired joint angles to be sent to the robot, to satisfy the position where the user intends to transfer objects with it.

4.5 Evaluations

This section presents two main analysis. Firstly, a workspace analysis, given the need to find a solution in a workable time. Secondly, to evaluate the manipulator arm's motion in terms of reachability of the end-effector.

4.5.1 Workspace Analysis

Cyton gamma 1500 arm has a large workspace which leads to a large, not-suitable for inverse kinematics, calculation time. It is not reliable to look for all the solutions inside its workspace. To decrease the required time, it is necessary to limit the number of orientations for the end-effector. For this particular task, 4 orientations are used to improve the robot workspace, as well as the interaction between robot and user. These orientations are defined by the following rotation matrices (4.6, 4.7, 4.8 and 4.9).

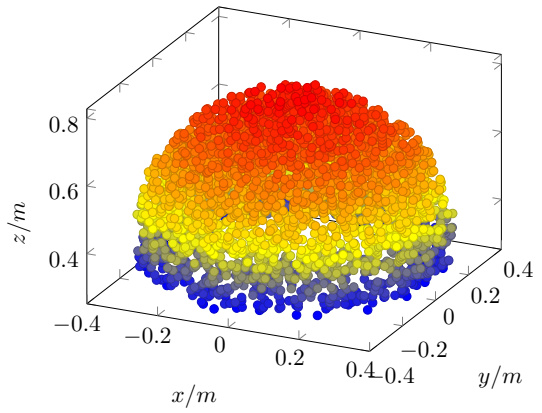
$$R1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

$$R2 = \begin{bmatrix} 0.7071 & 0 & -0.7071 \\ -0.7071 & 0 & -0.7071 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.7}$$

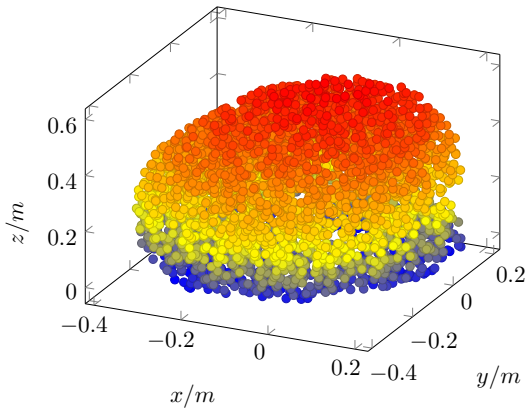
$$R3 = \begin{bmatrix} 0.7071 & 0 & 0.7071 \\ 0.7071 & 0 & -0.7071 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

$$R4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.7071 & -0.7071 \\ 0 & 0.7071 & 0.7071 \end{bmatrix} \tag{4.9}$$

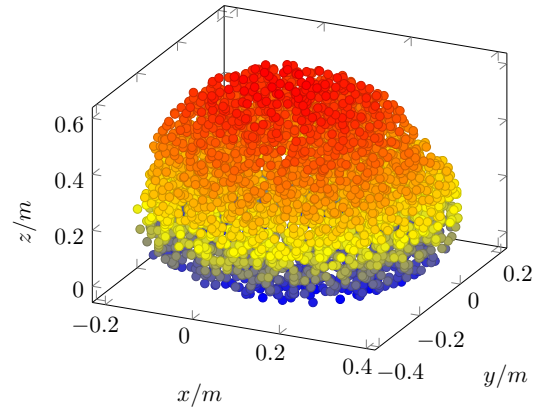
By using Cyton MoveIt! forward kinematics it is possible to calculate the workspace generated for these orientations. Figure 4.5, shows the available workspace for these 4 end-effector orientations considering a resolution of 0.01 m in Cyton's gripper x,y and z coordinates.



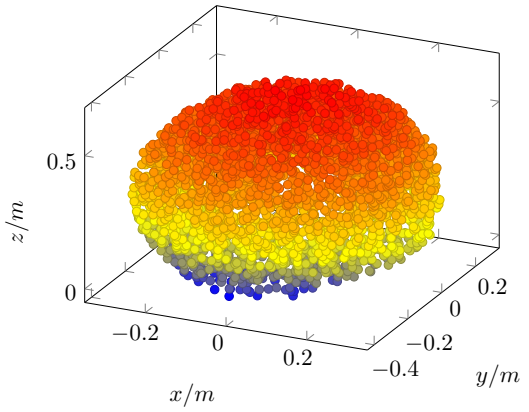
(a) Gripper orientation (R1).



(b) Gripper orientation (R2).



(c) Gripper orientation (R3).



(d) Gripper orientation (R4).

Figure 4.5: Limited workspace to decrease inverve kinematics calculation time.

The global workspace for robot-user interaction is presented in figure 4.6. This workspace results from the combination of the previous 4 end-effector orientations.

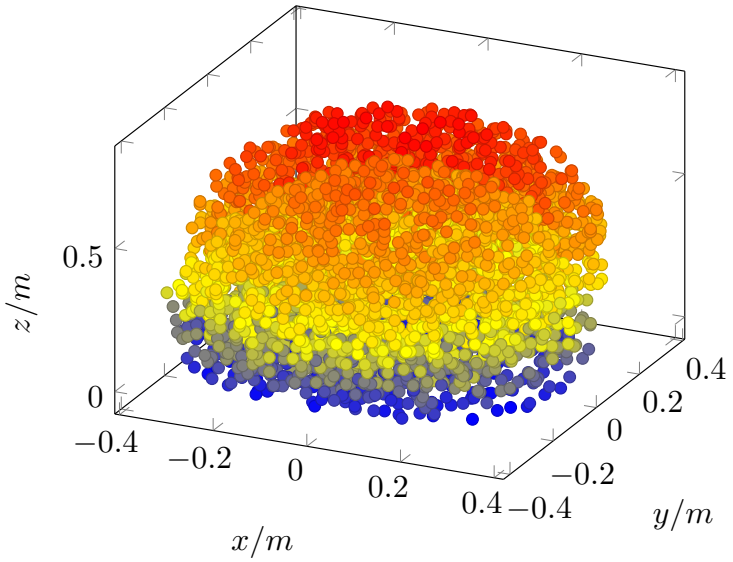


Figure 4.6: Global limited workspace to decrease inverse kinematics calculation time.

The maximum run rate for the inverse kinematics algorithm, in these conditions, is 10Hz. The time necessary to calculate Cyton inverse kinematics with this analysis is approximately 10 times better than the previous required time. Additionally, these orientations make the transfer task easier, since they allow the user to make contact with the sensor, using the object, without considerable manipulation.

4.5.2 Reachability Analysis

To evaluate the manipulator arm's behaviour, a reachability analysis is done. For this analysis, 11 random points inside Cyton workspace are used. In figure 4.7 it is possible to see how the arm behaves according to Cyton's gripper x,y and z coordinates.

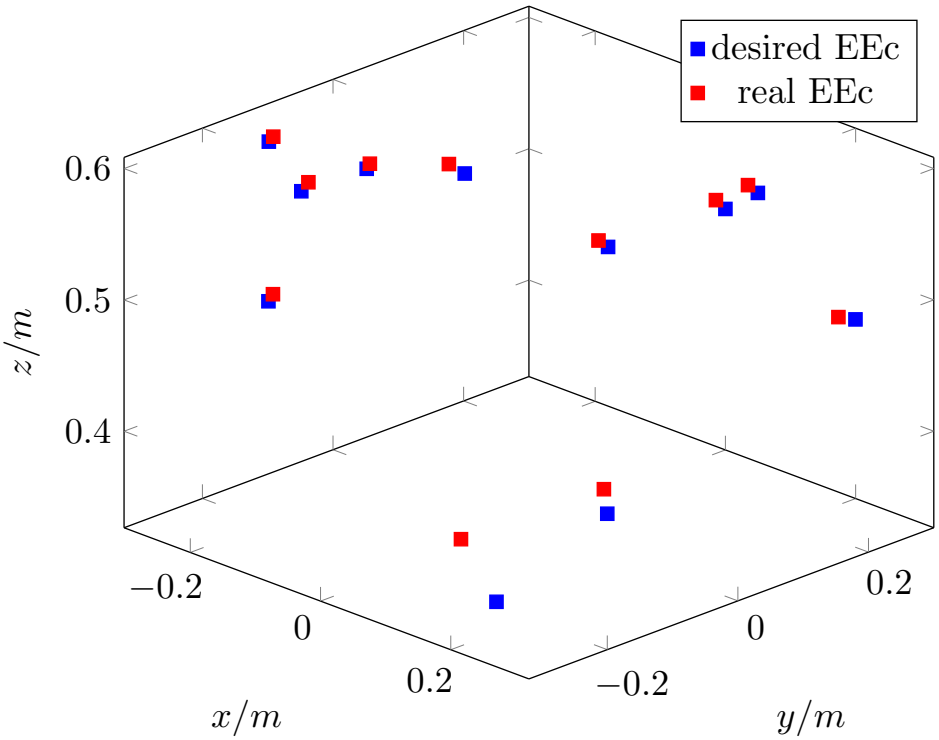


Figure 4.7: Robotic manipulator reachability test.

Figure 4.8 shows the joint performance for the points shown above.

EE coordinates and joint angles deviations are presented in Table 4.2. The EE average error is approximately 1% being that joint 1, 6 and 0 are the ones that contribute the most.

Table 4.2: EE coordinates and Joint angles deviation.

Average percentage error of deviations									
x	y	z	Joint 0	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
1.02	1.06	1.08	1.77	4.36	0.50	0.87	0.34	0.32	2.91

The results show that the accuracy of the robot control is very high.

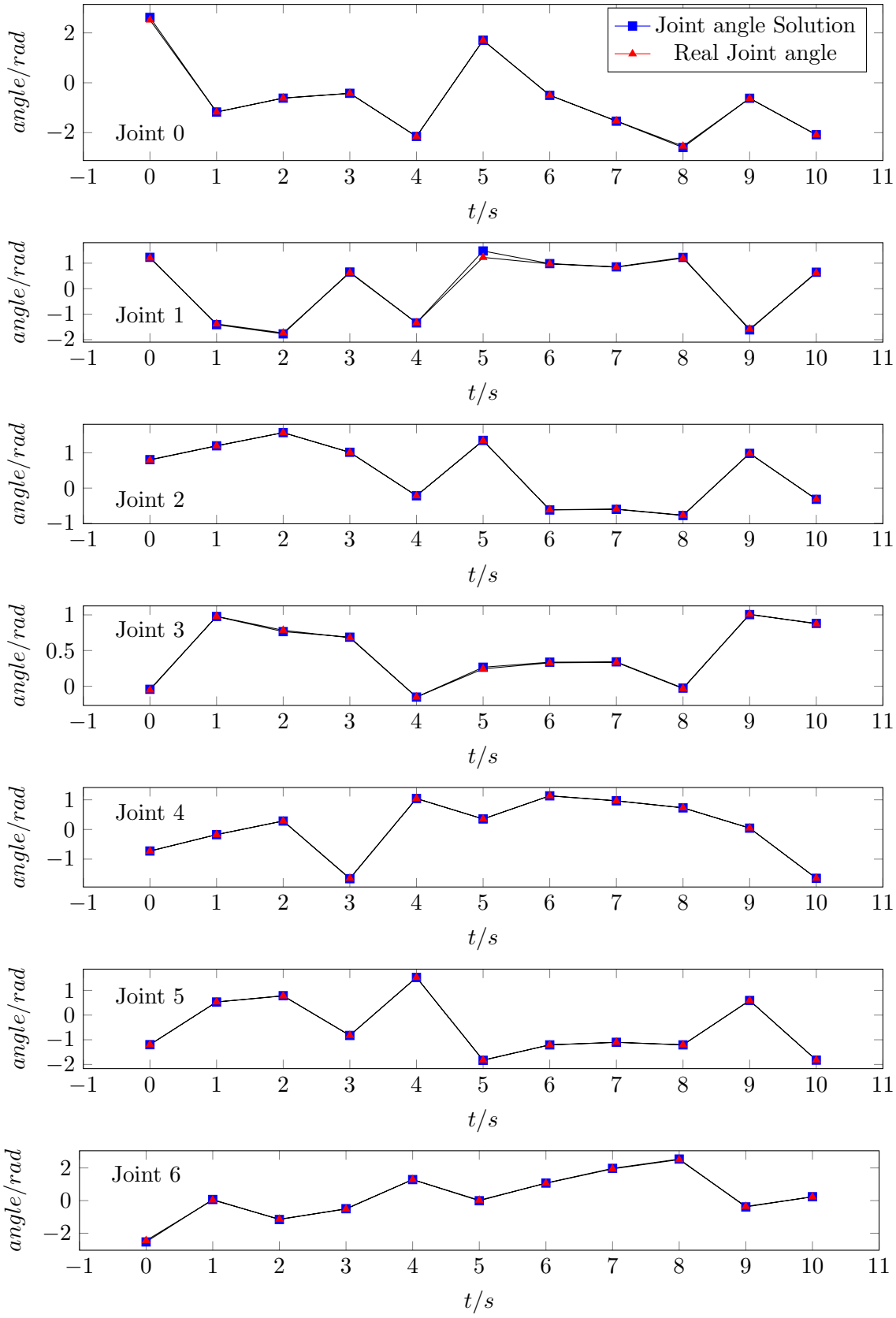


Figure 4.8: Robotic manipulator joint performance.

Chapter 5

Pre-contact Approximation using 3D Vision

This chapter presents all the work related to vision, using a Kinect sensor (RGB-D camera). Figure 5.1 presents the 3D sensor, with the corresponding axis, used throughout this work.

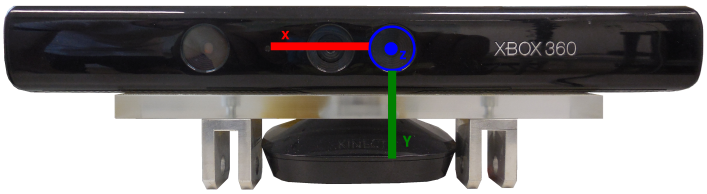
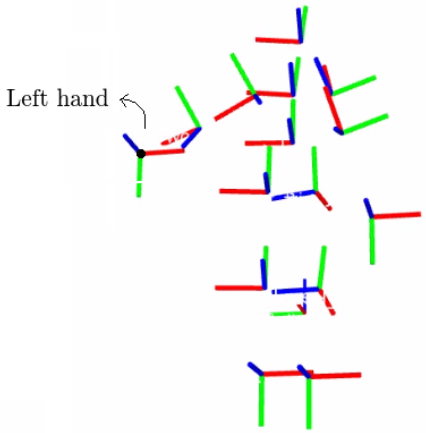


Figure 5.1: Kinect reference axis. The 3D axis colour red, green and blue refer to XYZ sequentially.

Firstly, vision system algorithms are implemented to allow the robot to move closer to the user, in order to promote the object transference. It can be seen as the pre-contact interaction between human and robot. Secondly, a object is detected so that robot knows when the user intends to enter in a transfer state. Two Kinect-Robot frame transformation approaches are calculated, implemented and compared so as to use the one that best fits the problem. Additionally, some experimental results are discussed. The performance of the implemented systems, such as 3D sensor accuracy analysis and object detection success rate are also evaluated in the end of this chapter.

5.1 Human Hand Tracking

The human hand tracking is based on openNI tracker package and NITE middleware library mentioned on chapter 3. Figure 5.2a illustrates the user skeleton joints, acquired with Rviz, in a particular position. Figure 5.2b represents, both at RGB and depth levels, the position from which is the skeleton joint tracked. For this work, only the left hand is tracked, and it is without considering its orientation. The user plays the role of adapting the orientation to better match the robot configuration, similar to what is done between humans.



(a) User skeleton joints.

(b) RGB and depth levels user images.

Figure 5.2: User hand tracking. Note: The skeleton joint map is horizontally inverted to facilitate the visualization.

The position of the hand is published as a tf transform and it is subscribed by a node (Kinect_HT) that handles the transformation of these coordinates from kinect to robot frame, and advertises them at the /EES_D topic. This topic containing the desired robot EE position is later used as the input for the inverse kinematic calculation. The ROS structure of the hand tracking is presented in Figure 5.3.

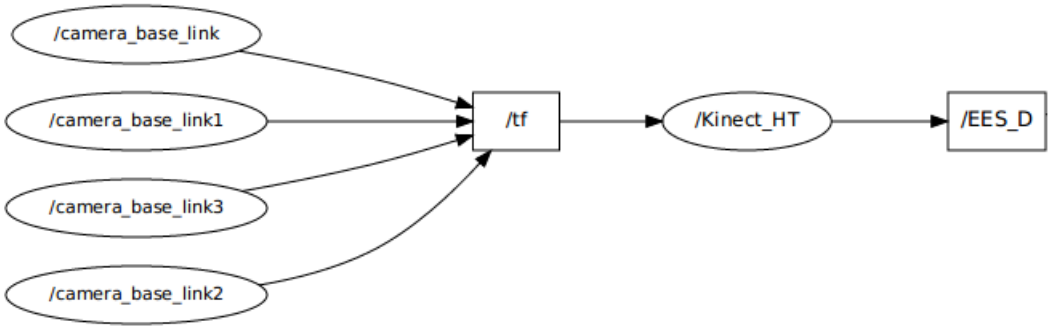


Figure 5.3: Hand tracking ROS structure.

5.2 Kinect-Robot Frame Transformation

To make robot end-effector position match the human hand, with the appropriated offset, it is required to transform the points read by kinect. Two methods to transform the kinect frame to the robot frame are implemented in order to evaluate the best solution. The first method (method 1) is done manually with a measurement tape. By measuring the translations and rotations along the three axes, the transformation is calculated. The second method (method 2) uses two point clouds: one for the kinect frame and the other for the robot frame. To feed the point clouds, the user hand has to match with the end-effector position. By using openNI tracker and robot forward kinematics, these points are recorded. The last step is to use the function "estimateRigidTransformation" to estimate the transformation.

Figure 5.4 illustrates how the Kinect is placed relatively to the robotic arm. Positioning the 3D sensor in this place leads to a better detection of the user hand. The workspace is within the 3D sensor field of view and nominal depth range. The layout also ensures that the robotic arm does not block the human hand relatively to the sensor.

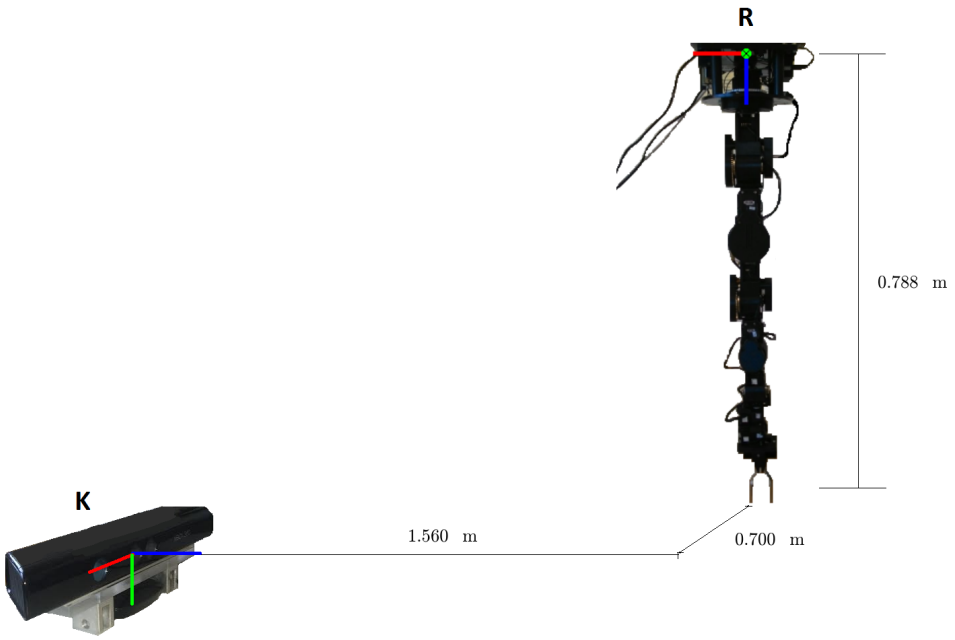


Figure 5.4: Kinect and robotic arm layout. R - Robot; K - Kinect.

Transform methods 1 and 2 between these two frames are represented by expressions 5.1 and 5.2 correspondingly .

$$ {}^{R}T_{K} = \begin{bmatrix} 0 & 0 & -1 & 1.560 \\ -1 & 0 & 0 & -0.700 \\ 0 & 1 & 0 & 0.788 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1} $$

$$ {}^{R}T_{K} = \begin{bmatrix} 0.028 & 0.019 & -0.999 & 1.561 \\ -1 & 0.007 & -0.028 & -0.652 \\ 0.007 & 1 & 0.019 & 0.763 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2} $$

5.3 Object Detection

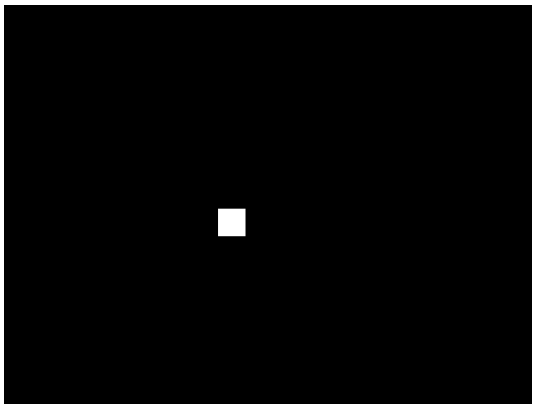
Object detection is a challenging task in the field of computer vision. The algorithms for detection have some limitations caused by illumination, occlusion, scales and background. The method implemented uses OpenCV features to handle image colour processing to detect the object. In order to distinguish the object from the background, they should have a significant colour difference. Hue, Saturation and Value (HSV) is a cylindrical coordinate representation of points in an RGB colour model. The object to be detected is the red cube presented in Figure 5.5.

Figure 5.5: Red cube.

Colour Based Detection

To detect the cube, a colour image is taken and converted to HSV colour-space using the limits corresponding to the object colour. In this particular case the values are within the following range: $140 \leq H \leq 179$; $179 \leq S \leq 255$; $126 \leq V \leq 191$. These limits range take into account environmental changes, namely luminosity shifts, which can lead to the emergence of spurious white pixels in the threshold image. Some morphological operations, like opening and closing operations (openCV functions: "cv::erode" and "cv::dilate"), are implemented to eliminate these unnecessary pixels as well as the noise produced by the sensor. Figure 5.6 presents a colour image of the 3D scene with the detected object (Figure 5.6a) and illustrates the treated threshold image (Figure 5.6b).

(a) Colour image of the 3D scene.



(b) Thresholded image.

Figure 5.6: Colour based object detection.

The implemented algorithm can be described by the following pseudo code:

Algorithm 1 Colour based detection.

```
for each RGB frame do
  threshold using HSV;
  morphological operations (erode and dilate) and area boundaries;
  if object detected then
    transfer state = true;
  else
    transfer state = false;
  end if
end for
```

5.4 Evaluations

This section has the objective to present some experimental results and evaluate the performance of the implemented systems.

5.4.1 Accuracy Analysis

In order to evaluate how accurate this algorithm works and to settle an offset between human hand and robot gripper (to avoid collisions), five fixed points inside robot workspace are marked and measured using a tape with an error of $\pm\, 0.5 * 10^{-3}$ m. To ease the marking, the points are aligned with the 3D sensor and spaced by 0.2 m along the Z axis, as illustrated in figure 5.7.

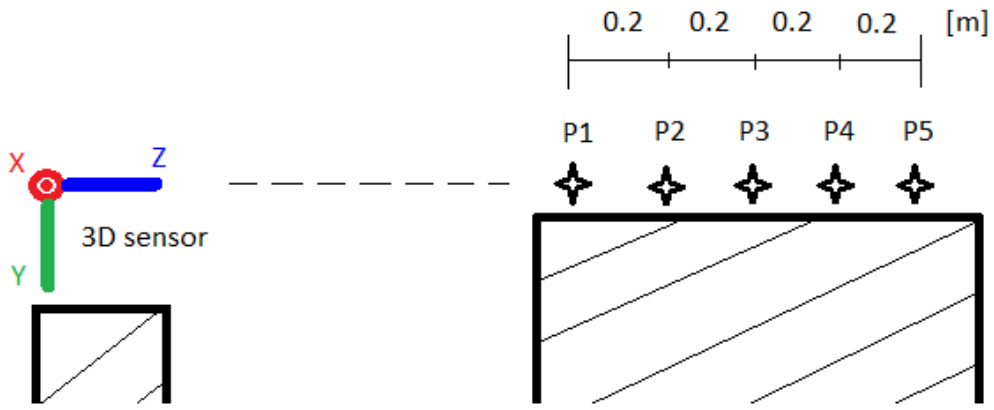


Figure 5.7: Experimental procedure to evaluate 3D sensor accuracy.

For each point, 100 readings of the hand positioned on those points are recorded. Table 5.1 presents the absolute average and max error of these readings.

Table 5.1: 3D sensor average and max error.

	Average error [m]	**Max error [m]**
x	0.0358	0.0647
y	0.0395	0.0571
z	0.0283	0.0477

To have a safe human-robot interaction, the offset must be settled according to the obtained absolute max error. Since this is a sensitive subject and the method implemented lacks precision, a research is made in order to support these results. Akansel et al [41] analysed the joint tracking errors of openNI using as ground truth data: the data extracted from Vicon motion capture system. The average joints errors showed up as usually more than 5 cm. According to these results and to the workspace in use, the offset must be larger than 7 cm in robot y axis to avoid collisions.

5.4.2 Frame Transformation Analysis

It is necessary to evaluate the two methods implemented to relate the kinect and the robot frames in order to choose the one that best fits the problem. A set of coordinates well known to the robot are used as the input for the transformation. Using the matrices given above (section 5.2) the coordinates are transformed and then compared with the expected ones. From table 5.2 it is clear that the method using a measurement tape gives better results than the point-cloud based method.

Table 5.2: Frame transformation methods comparison.

	Average error [m]	
	Method 1	**Method 2**
x	0.0546	0.0715
y	0.0567	0.0635
z	0.0495	0.0673

Concerning the first method, the average error showed up as approximately 5.36%, on average less 1.38% when compared with the second transform approach.

5.4.3 Object Detection Success

This evaluation indicates how practical the implemented method is to detect the object. It is well known that luminosity conditions change from one day to another, which is a problem when vision is concerned. Although the algorithm allows a user to manually change the HSV parameters, quite easily, the success rate is measured with fix parameters. The data was recorded 5 times daily for 10 days taking into consideration three conditions (50 samples for each condition):

(i) Only the environment luminosity changes are considered. Objects with a colour similar to the cube are removed from the 3D sensor field of view.

(ii) Environment luminosity changes plus red objects in the environment (such as: red clothes, red figures in wallpapers and red writing material).

(iii) The HSV parameters are adapted daily and the objects with a colour similar to the cube are removed from the 3D sensor field of view.

Table 5.3, presents the object detection success rate.

Table 5.3: Object detection success rate.

Object detection success [%]		
(i)	(ii)	(iii)
80	45	92

Using the HSV range presented above (subsection 5.3), and considering only the environment luminosity changes (condition (i)), the object is detected 80% of the time. One of the biggest issue is, if a user wears red clothes and/or there is red objects in the environment, which leads to a significantly lower sucess rate, 45%, even considering area boundaries. Being aware of not having red objects in the environment and changing the HSV parameters once a day, the success rate rises to approximately 92%. This method has the disadvantage proven with the results stated above, but shows up as a enough solution for this work. Else, if the environment conditions don't significantly change, this algorithm can always detect the object.

Chapter 6

Contact and Force-based Interaction

This chapter aims to present the work done to enable the transfer of objects by contact, between a human and a robot. Force-based actions are based on the processing of the forces and torques which are transmitted between the robot and the objects when they come into contact. These force/torque values, which are registered by force/torque sensors installed in the robot, are usually used as inputs for control laws which guarantee that contact force/torque are regulated towards a predefined reference suitable for the development of the robotic task [6].

The force/torque sensor ATI mini40 (figure 6.1a) fell during the installation in the arm, resulting in a damaged sensor. Although collecting data was an impossibility from then on, the study and work developed before the damage is briefly presented next. Additionally, two FSR's (figure 6.1b) are implemented to fulfil the need to fill the object between gripper's fingers. The chapter ends with a succinct experimental results discussion and a brief overview of the final overall system and ROS architecture.

(a) F/T ATI mini40. (b) FSR.

Figure 6.1: Force measurement hardware.

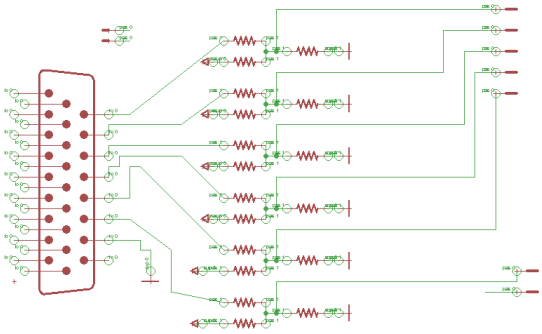
6.1 ATI Mini40 Integration

In order to use the ATI mini40 transducer it is necessary to use a device to acquire the signals sent by the sensor and then send it to a PC for later conversion into forces and torques. Additionally, a support is required to install the sensor in the robotic arm.

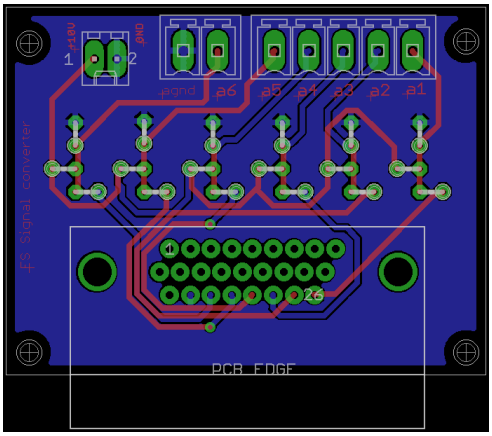
6.1.1 Signal Converter

An Arduino with Ethernet Shield is used to acquire the signals. Since the Arduino available can only read analogue inputs within a range of 0V-5V, a proper method to convert these signals is needed. A dedicate circuit was developed to perform signal conditioning, using EAGLE software. Figure 6.2 presents the developed schematic and board to convert these

signals. This circuit accepts as the input a plug DB26 where the wires, with the amplified signals, are set up. The output will link directly with the Arduino.



(a) Signal converter schematic.



(b) Signal converter board.

Figure 6.2: Signal conditioning circuit to interface the ATI Sensor.

The operating principle of the developed circuit is described in Figure 6.3. For each signal, when the input is +10V, the output voltage is 5V. On the other hand, -10V input results in a 0V output voltage. Anywhere in between, the input voltage maps linearly to the output.

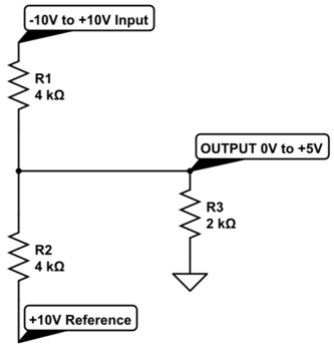


Figure 6.3: Electric conditioning operating principle.

6.1.2 Code Implementation

Since the Arduino board can now read the analogue signals, the next step is the code development. Firstly, an Arduino works as a TCP/IP server that accepts a connection from the client computer, and sends messages to it with the received signals. A Ethernet Shield is incorporated in the Arduino to be possible to communicate through TCP/IP. Secondly, the client computer receives the messages, decodes them and converts them into force and torques. Converting these signals into forces and torques is the most challenging task related to this force/torque sensor. Each gauge does not match a single force or torque in a single direction, which means that each of them depends on all six gauges. A complex matrix is implemented to assure a correct output. ATIDAQ provides a library that can be adapted for this particular assignment. This code library uses standard C to read calibration files, configure the transducer system, and convert voltages from data acquisition system into forces and torques, to name a few. The required matrix to transform voltages into forces and torques for this sensor is presented in equation 6.1.

$$T_M = \begin{bmatrix} 0.15294 & 0.06576 & -0.45575 & 12.04803 & 0.46830 & -12.41577 \\ 0.39540 & -14.52394 & -0.25924 & 7.05382 & -0.47846 & 7.08434 \\ 20.69524 & -0.07941 & 20.91555 & -0.42487 & 20.78888 & -0.80079 \\ 0.00139 & -0.08124 & 0.29133 & 0.03347 & -0.30289 & 0.04973 \\ -0.33941 & 0.00082 & 0.17312 & -0.06970 & 0.16798 & 0.06230 \\ 0.00138 & -0.17895 & 0.00772 & -0.17225 & 0.01091 & -0.17652 \end{bmatrix} \tag{6.1}$$

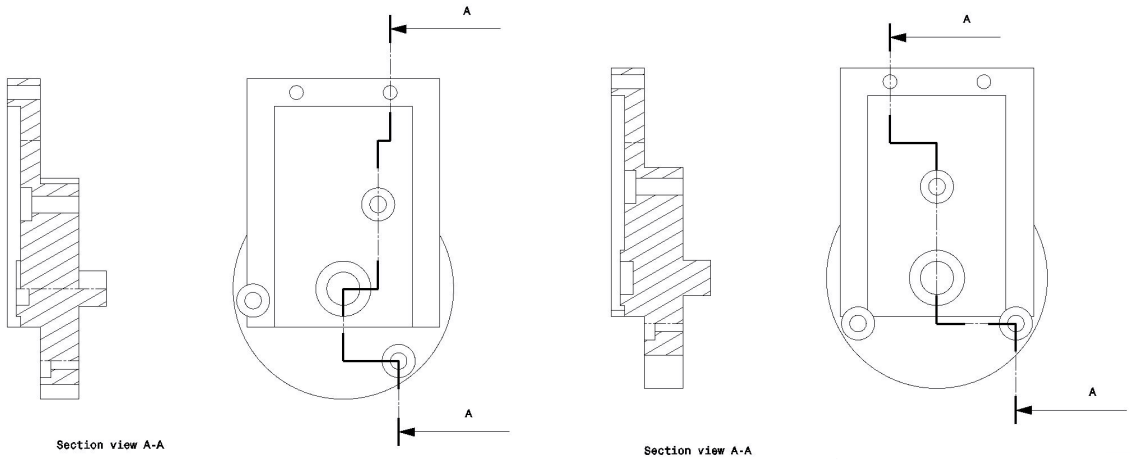
Multiplying the transform matrix by a vector of gauges readings (V), it is possible to find the forces and torques values for these readings, as highlighted in the equation 6.2.

$$Ft^\top = T_M \times V^\top, \tag{6.2}$$

where $Ft^\top$ is the transpose vector of $Ft = [F_x \;\; F_y \;\; F_z \;\; T_x \;\; T_y \;\; T_z]$, and $V^\top$ is the transpose vector of $V = [G0 \;\; G1 \;\; G2 \;\; G3 \;\; G4 \;\; G5]$.

6.1.3 Sensor Installation

The last step to install this sensor is the integration with the arm. In order to enable the installation of the sensor in the arm, it is required to create two special components. Firstly, using a CAD software, the prototype is created and secondly a CNC machine is used to contrive the support. Figure 6.4 presents the CAD prototype developed using CATIA V5 software (mounting side adapter, Figure 6.4a and tool side adapter, Figure 6.4b). The entire sheet of the developed components is presented in apendix A.



(a) Mounting side adapter. (b) Tool side adapter.

Figure 6.4: Support blueprint to integrate the sensor with the arm.

Figure 6.5 illustrates the real assembly of the sensor in the cyton gamma 1500 manipulator. This adapter concerns the robot global axes, which means the sensor axes are align with the robot ones.

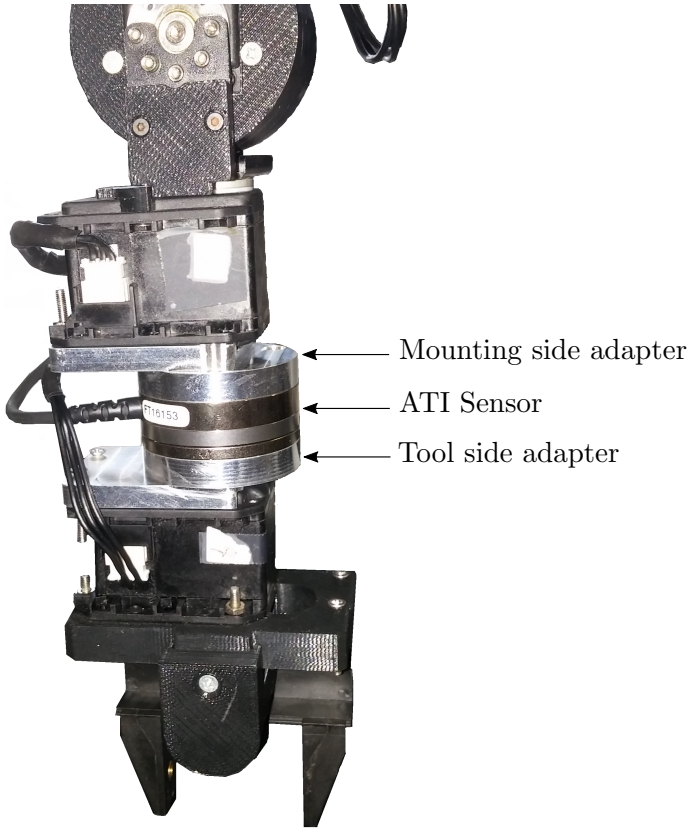


Figure 6.5: ATI Sensor assembly into the robot arm.

6.1.4 Signal Overview

Figure 6.6 presents the global flow of signal processing. The transducer is powered up through the supply box and receives the forces and torques. These signals, treated by the dedicate circuit, follow to the arduino that sends them through TCP/IP to the client computer. Finally, the created node in the client PC converts the signals into forces and torques.

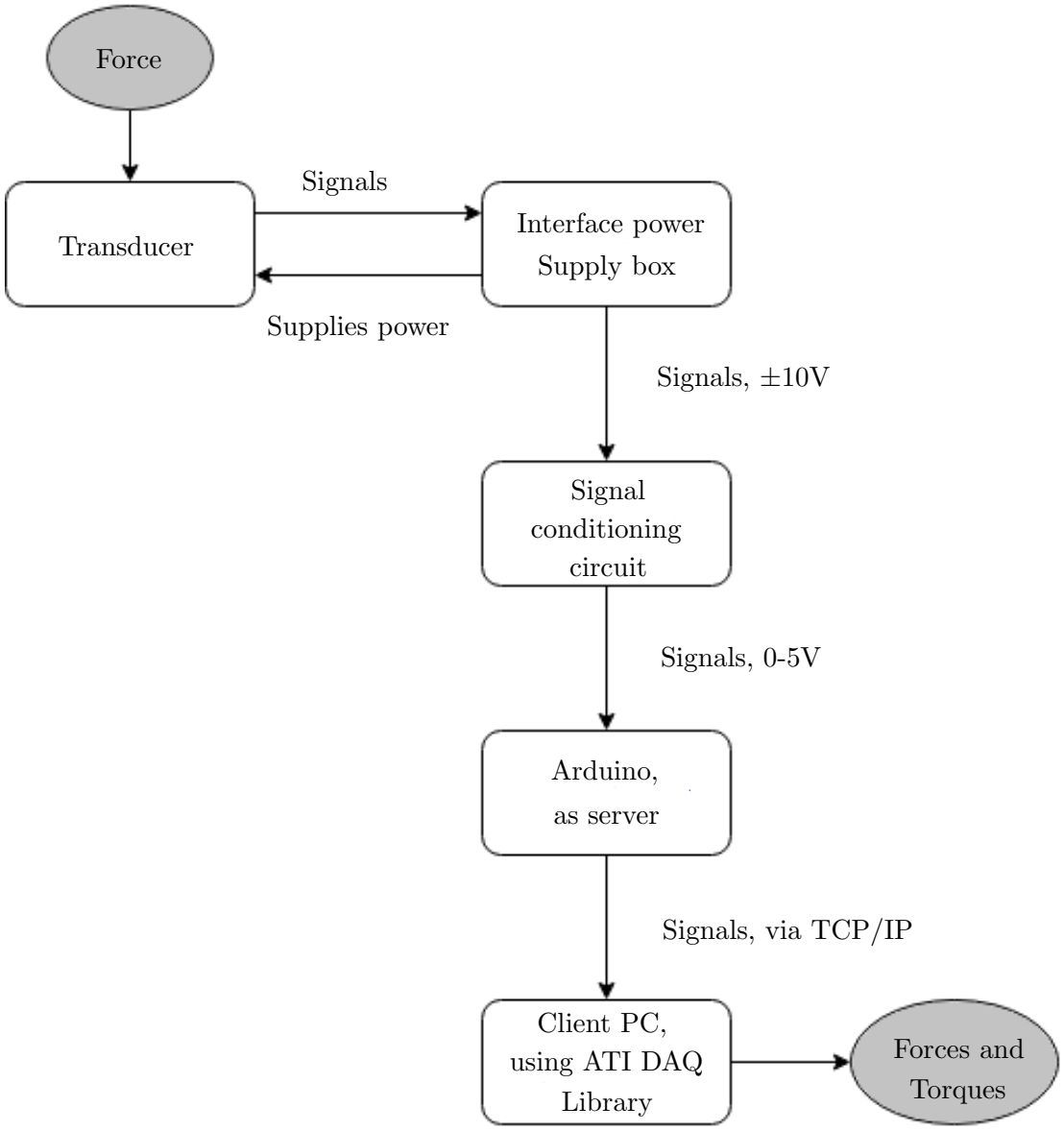


Figure 6.6: Overall signal flow from transducer to force and torque values.

Signal Reading Example

As mentioned above, the sensor got damaged during the installation, leading to the impossibility to get readings while transferring objects with the robot. Despite this misfortune, Figure 6.7 illustrates a qualitative reading example where it is feasible to detect two phases:

Phase A - In this phase a non-constant force is applied matched with the Z-axis of the sensor. It can be seen that this generated almost only force and no torque.

Phase B - Here a random force is applied. It can be seen that the applied force was not matching with any axis, since it generated some force and torque.

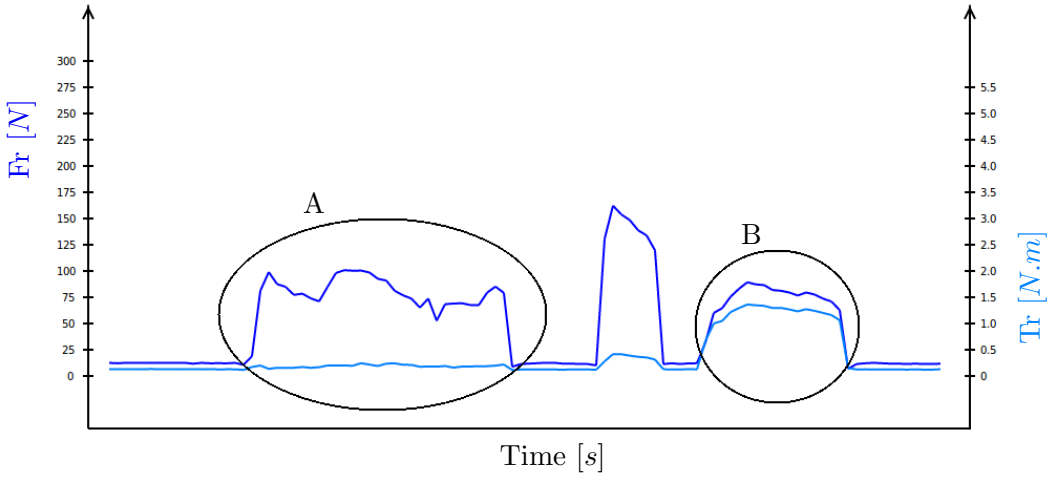


Figure 6.7: Example of resulting force and torque reading.

6.2 FSR Integration

Although the ATI transducer got damaged throughout the work, it is noticeable that the F/T sensor would not be enough, since there is a need to feel the object between gripper's fingers. Additionally, it is also necessary to detect the starting and ending phases of the object transference. To fulfil these needs, two FSR are installed in Cyton's gripper. The idea is to observe FSR's feedback and use it to dictate gripper actions. Using an Arduino with an implementation similar to the one already described in a previous section (only by changing the message to be sent to the client) and two voltage dividers it is possible to do the readings.

Figure 6.8 presents a finite-state machine (FSM) for the object transfer task. This FSM has 4 states (nodes) and 5 conditions that affect these states. The FSM starts through the "Opened Waiting" state, where the gripper is fully opened and waiting for a user to start transfering an object. When the user starts touching the gripper installed FSR with the object, the gripper starts closing and the state changes ("Closing" state). If the sensors reach a similar force, it means that the robot is holding the object and the state machine moves into the "Closed Waiting" state. On the other hand, if the transference fails, the gripper will reach its minimum position (fully enclosed) and the state machine goes to the "Opening" state instead of the "Closed Waiting" state. To move from the "Closed Waiting" state to the next state it is necessary that the user starts extracting the object from the robot gripper. This action results in a force disparity (the force feedback from each sensor is not equal) that leads the state machine to the "Opening" state. Once in the "Opening" state, when the robot gripper reaches its maximum position, the state machine moves to the "Opened Waiting" state.

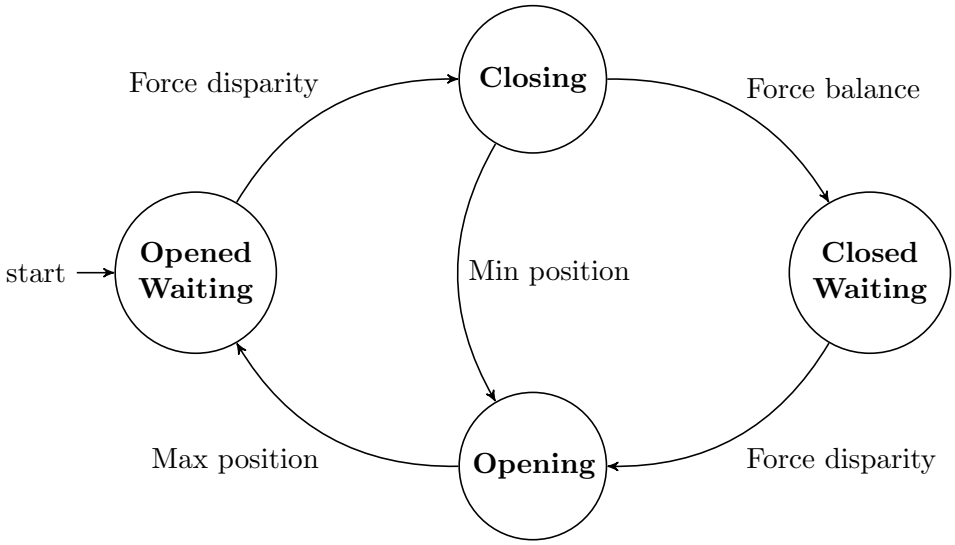


Figure 6.8: Diagram with states and transitions modeling the objet transfer between a human and the robot.

6.2.1 FSR's Overall Assembly

Figure 6.9 presents the overall assembly of the force sensitivity resistors. The FSR's are glued in the center of the gripper's fingers. These are connected to a box (illustrated in the right side of the figure), containing the Arduino plus shields and the voltage dividers, through flex cable. The Arduino is connected through Ethernet to the client computer.

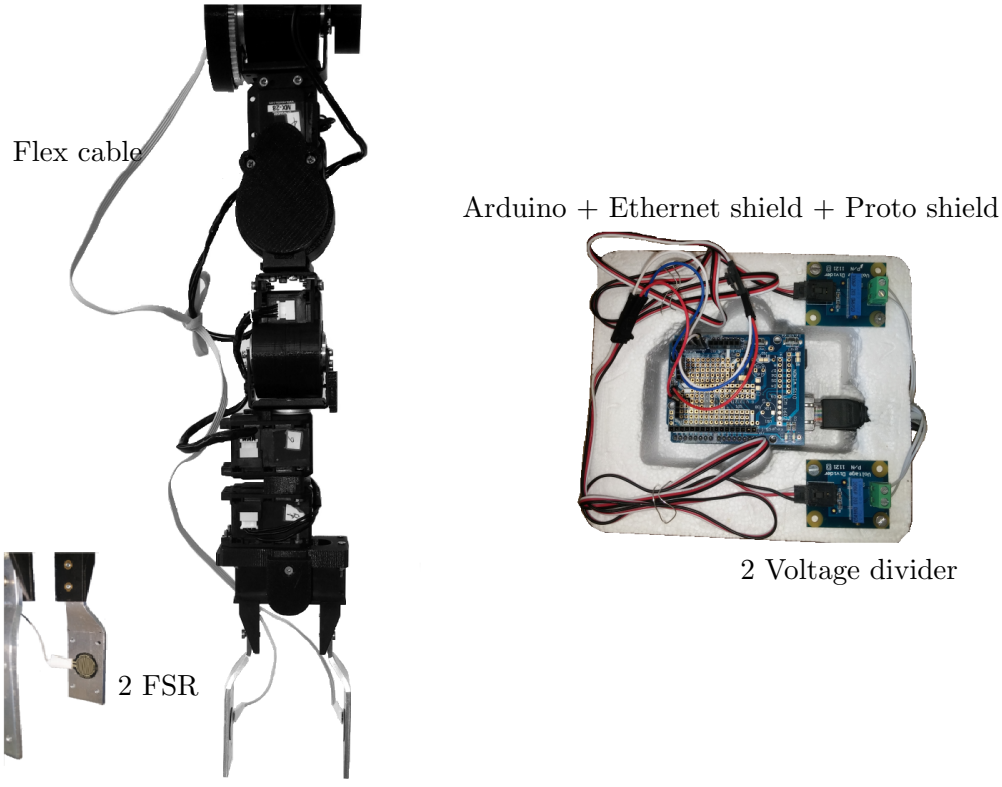


Figure 6.9: Overall assembly of the force sensitivity resistors.

6.2.2 Algorithm Implementation

The algorithm presented in this subsection results from many experiments leading to the optimization of the transference action. The algorithm foresees a transference where the user object gets in touch with one of the sensors and the robotic arm gripper is forced to close to its last position. If both the gripper installed sensors reach a pre-defined force, within a range of 15% difference between them, the gripper stops closing and stays in the current position. This force must be enough to hold the object weight. The friction of the objects is not considered in this work. The gripper velocity is directly proportional to the force applied by the user to the gripper, as described by the following equation:

$$\omega = f \times 0.2, \tag{6.3}$$

where ω is the gripper velocity in rad/s, and f is a dimensionless variable with magnitude equal to the average force (last 50 reads, force range magnitude [0-20]).

This implementation facilitates the user's job, since it also considers the possibility of the user not making the object touching one or both the sensors during all the transference. In this approach, the force readings are based on the last 50 readings.

To streamline the process, avoid conflicts and further the control, a node is developed to control the system phases. This system is divided into gripper closure and opening phases:

Gripper closure phase: The gripper is wide-open and there is no object between it. The user decides to transfer an object, in this case the gripper will often end holding an object[1].

Gripper opening phase: The robot is holding an object and waiting for the user to grab it.

This implementation starts in the gripper closure phase. The following two pseudo codes (algorithm 2 and 3) shows how the gripper is controlled in each phase for the approach explained before:

Algorithm 2 Closure phase

```
while gripper closure phase do
  if average of the last 50 reads > pre-defined force then
    set speed proportionally to average force;
    send gripper to is last position (fully enclosed);
    if FSR_A & FSR_B force read approximately equal then
      send gripper to its current position;
      gripper opening phase = true;
      gripper closure phase = false;
    end if
  end if
end while
```

[1] If the user fails to transfer the object and the gripper reaches its last position, it moves back to the position where it is fully open. The state here is preserved.

Algorithm 3 Opening phase

```
while gripper opening phase do
  if FSR_A & FSR_B force read outside threshold range then
    set speed;
    send gripper to is 1st position (fully open);
    gripper closure phase = true;
    gripper opening phase = false;
  end if
end while
```

Action Controller ROS Architecture

Figure 6.10 shows the force-based interaction software ROS architecture. The first node Cyton_GA receives the force values from the Arduino server and publishes them into the topic FSR_val. Cyton_FC node subscribes from this topic and from the Fcontrol topic that contains the feedback from another node (Cyton_FC_control) responsible for managing the transfer states. Cyton_FC node uses this information to dictate the gripper actions, publishing the gripper positions and velocities into the gripper_controller/comand topic. Dynamixel_manager node is then responsible to make the gripper move to the desired position.

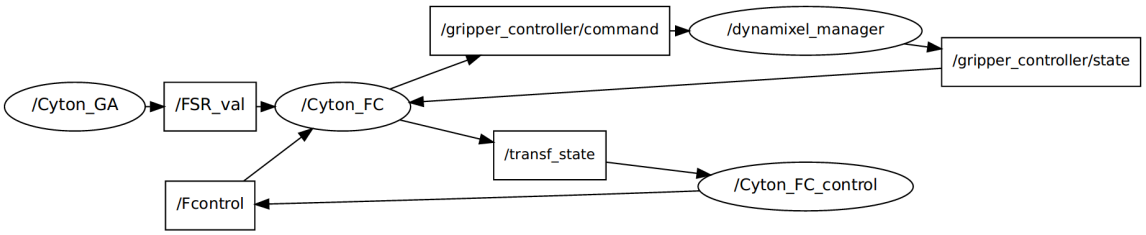
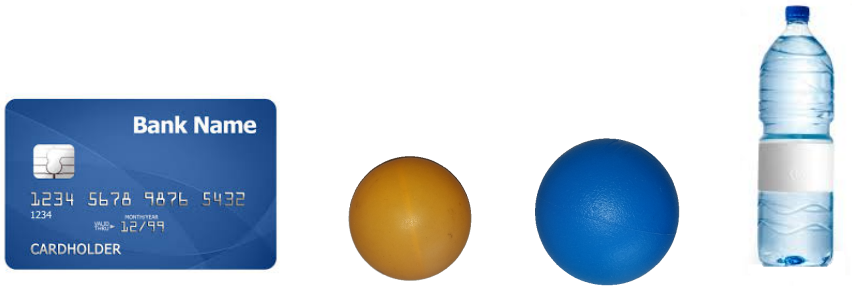


Figure 6.10: Force-based interaction software ROS architecture.

6.3 Evaluations

The object transfer task mainly consists of 4 objects being handed over from the user to the robot gripper, and vice versa. Since the robot arm stiffness is reduced, which affects the force propagation between the object and the sensor, the experimental tests are done with symmetric objects. Spherical geometry also showed as a solution for this issue, since it is easier to make contact between the sensors and the objects. The gripper's fingers are fixed in a pre-selected position for the following experimental tests. The success rate evaluation is independent of the gripper orientation. Despite the low precision of these sensors, it is possible to analyse the differences and the required force to transfer each object.

Figure 6.11 shows the objects in use: a card, a ping pong ball, a soft blue ball and a water bottle.

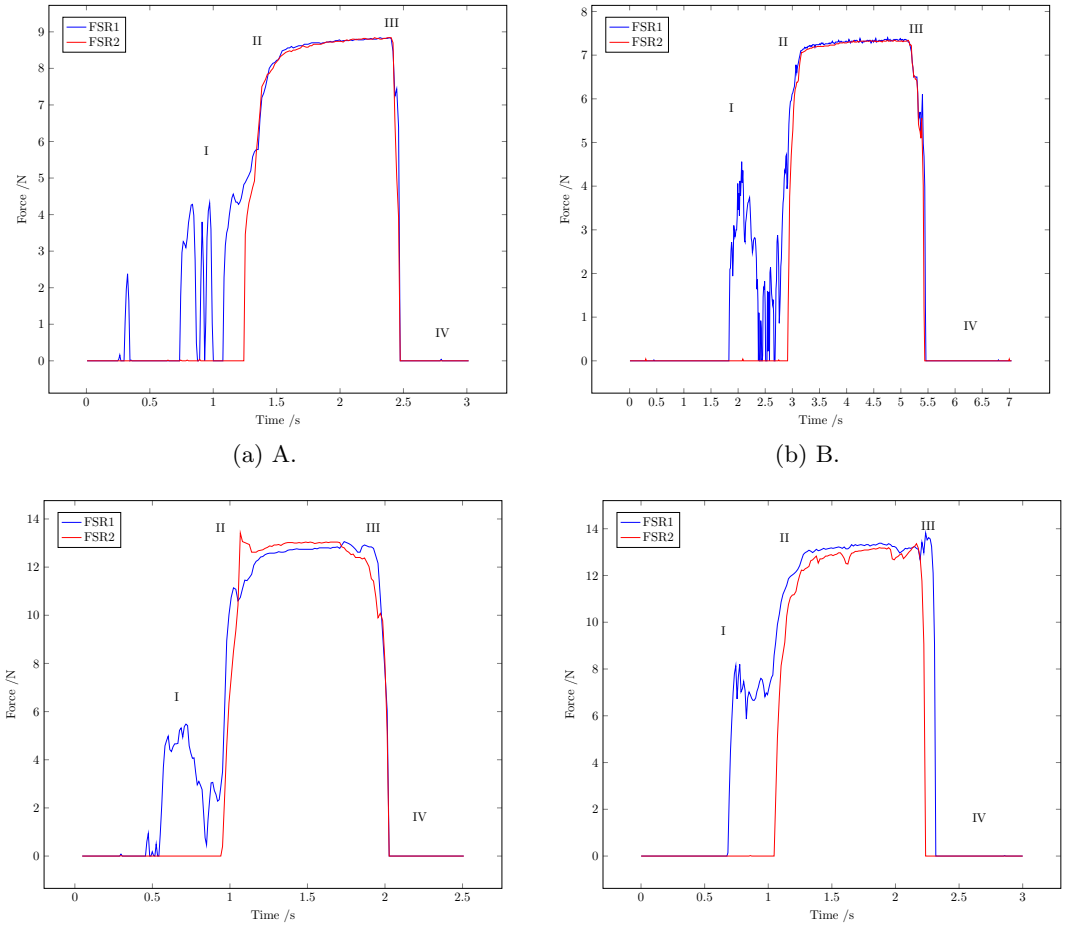


(a) A. (b) B. (c) C. (d) D.

Figure 6.11: Objects used in the transference. Note: The objects are not represented to scale.

6.3.1 Force Evolution in a Object Transfer Task

This subsection illustrates (Figure 6.12) a force evolution graph for each object. Each graph contains both FSR's readings. To start the transference, in order to standardize the results and ease the evolution analyses, the user starts transferring the object, always making contact between the object and the sensor named FSR1.

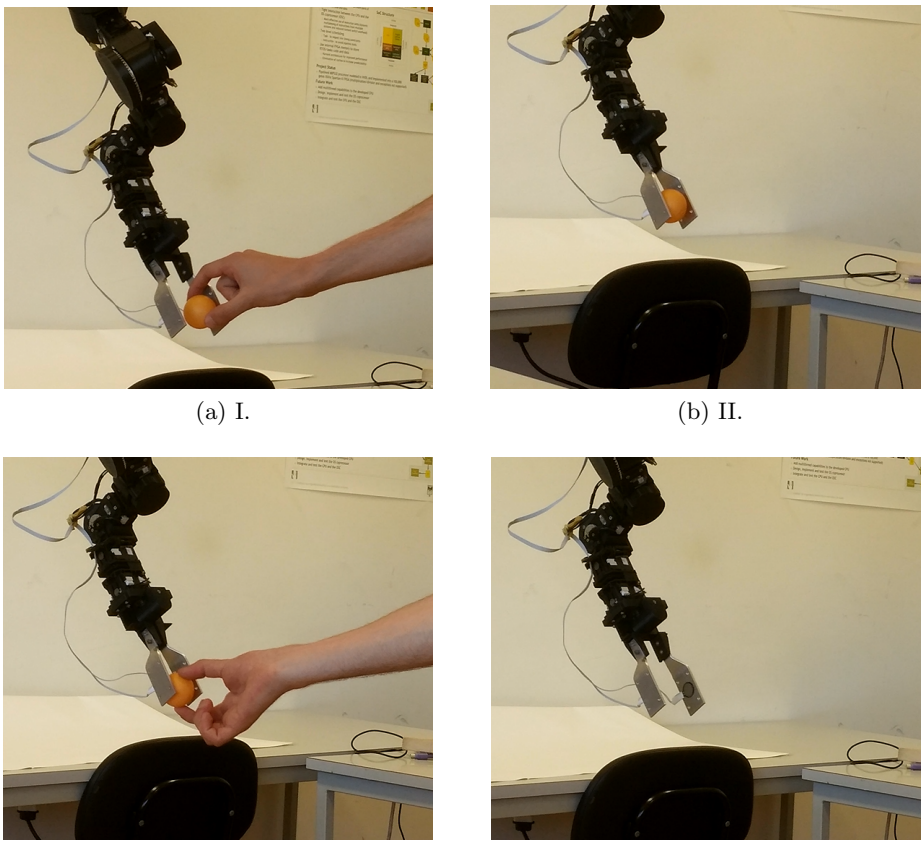


(a) A. (b) B.

(c) C. (d) D.

Figure 6.12: Force evolution for each object in a transference.

The phases I to IV marked on the graphs match the following illustrations (figure 6.13):



(a) I. (b) II.

(c) III. (d) IV.

Figure 6.13: Illustrations, using the object B, corresponding to the phases (I, II, III and IV) marked on the force evolution graphs.

The first distinguished phase (phase I) in all the graphics is the approximation between the object and the sensor where the first contacts are made. This phase is distinct between the objects and, as it can be seen in the next subsection, it is always distinct regardless the object in use. The second distinguished phase (phase II) happens when both sensors readings are the same within a 15% range. The gripper stops in that current position and the object is now being hold by the robot. The decreasing forces visible in the graphics (phase III), result of the user extracting back the object, it is the gripper opening phase. Consequently, the last phase (phase IV) occurs when the gripper is fully opened and there is nothing between its fingers.

It is visible that for different objects, despite the similarity in weights, the force required to hold each object is different. In order to take some accurate conclusions a repeatability test is done next.

6.3.2 Repeatability Analysis

The repeatability experimental tests consists of each object being transferred between the user and the robotic manipulator 10 times. Next figures (6.14, 6.15, 6.16 and 6.17) show those 10 transferences of each object.

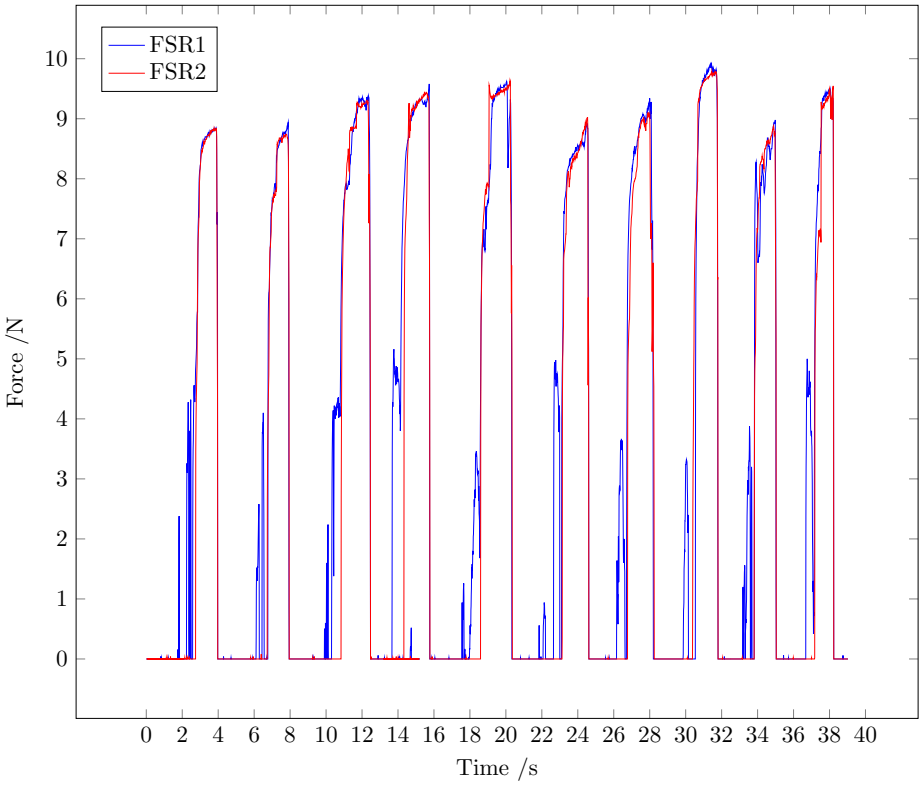


Figure 6.14: Repeatability test using object A.

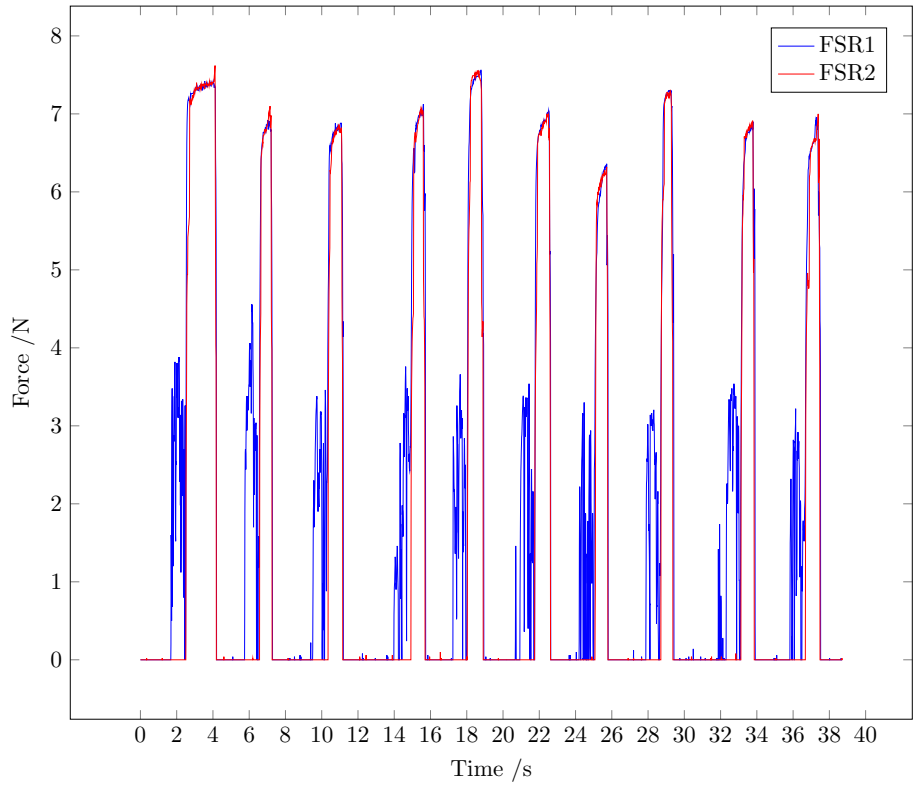


Figure 6.15: Repeatability test using object B.

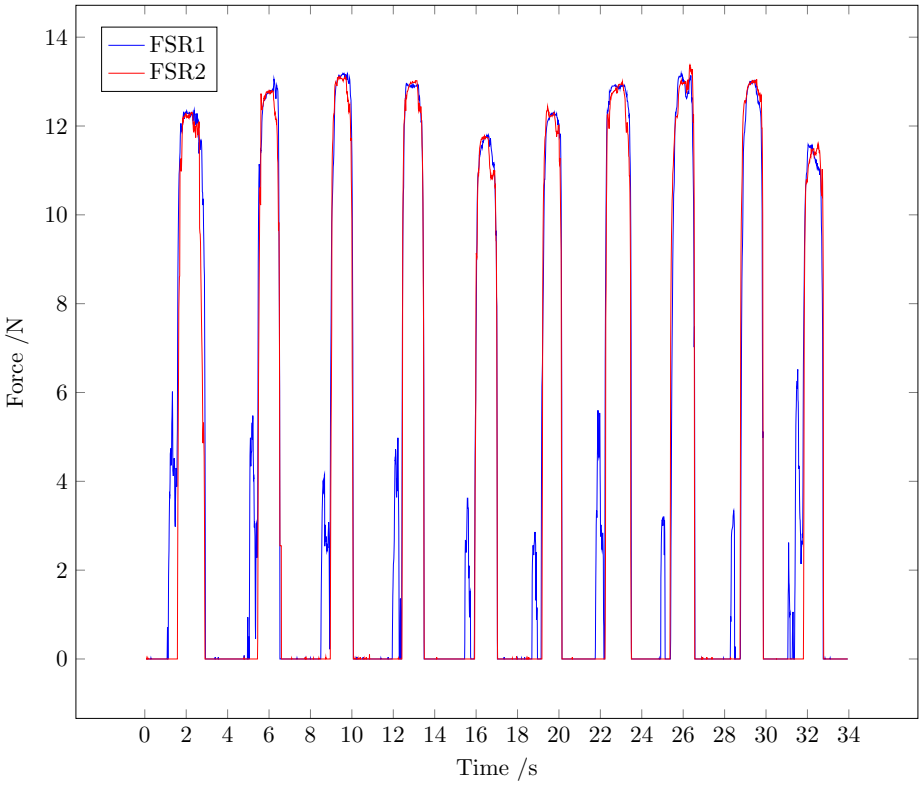


Figure 6.16: Repeatability test using object C.

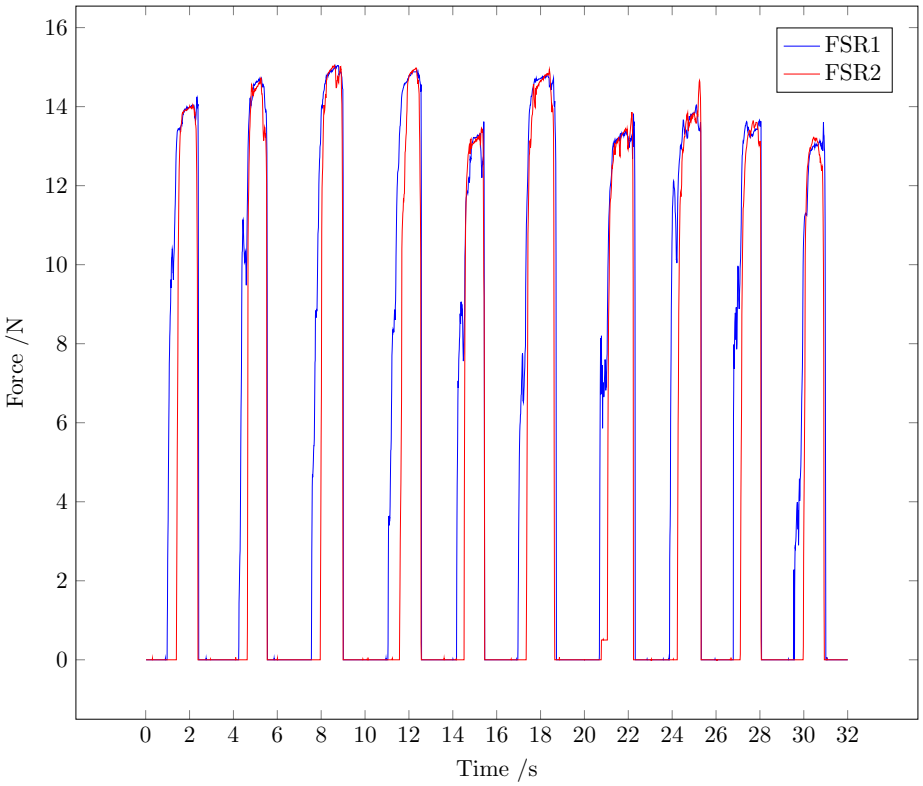


Figure 6.17: Repeatability test using object D.

First, regardless the initial phase stated above, the forces required to hold the objects are approximately 9, 7, 13 and 14 Newton respectively from object A to object D. These force differences are mainly caused by the objects different properties and geometries. These differences affect the way the contact occurs between object and sensor. It is visible that for a stiffer object (e.g. object A and B when compared with object C and D), the contact arises with some facility, resulting in a lower required force. On the other hand, the force required to hold the blue ball and the empty water bottle is higher. This higher force is noticed when the objects are slightly crushed by the gripper's fingers. In fact, the gripper is more closed than what would be normally required to hold these objects, because it is when the FSR feel the object. In this step, these objects are deforming more the resistors, which leads to a lower resistance and consequently a higher force.

6.3.3 Transfer Algorithm Success Rate

Each object is transferred 100 times in order to analyse the success rate. One successful transfer corresponds to the fulfillment of the four phases described on section 6.3. Table 6.1 shows the results of the transference success rate.

Table 6.1: Transference success rate.

Success rate [%]			
A	B	C	D
98	99	90	78

The success rate for the objects A and B is quite good, 98% and 99% respectively, given the facility to feel the contact between those objects and the sensors. The soft blue ball properties affect the success rate, which is 90%. The water bottle object is the one presenting the lower success rate, 78%, because not only its properties affect the transference, but its surface irregularities makes the contact even more difficult.

Given these objects, the results are the expected ones and quite successful; however the use of only 2 FSR limits the diversity of objects that can be handed with the robot (e.g. a non symmetric object may cause more problems to this set up, since one of its sides may not have contact with the sensor).

6.4 Overall System Architecture

The overall system architecture presents a succinct integration of the hardware and software used in this dissertation. Relative to hardware, this includes one Cyton gamma 1500 arm, two force-sensitive resistors, one Arduino with an Ethernet shield, and a Kinect sensor. Cyton arm and kinect are connected with the main computer through USB interface. On the other hand, the Arduino establishes a communication through Ethernet. The USB controller that binds the arm with the computer is set on TTL mode with a baudrate of 1 Mbps. The software architecture is based on the Indigo version of the ROS framework under ubuntu 14.04. The necessary packages to reach the objectives use C/C++ and python programming languages. Figure 6.18 illustrates the integrated system architecture.

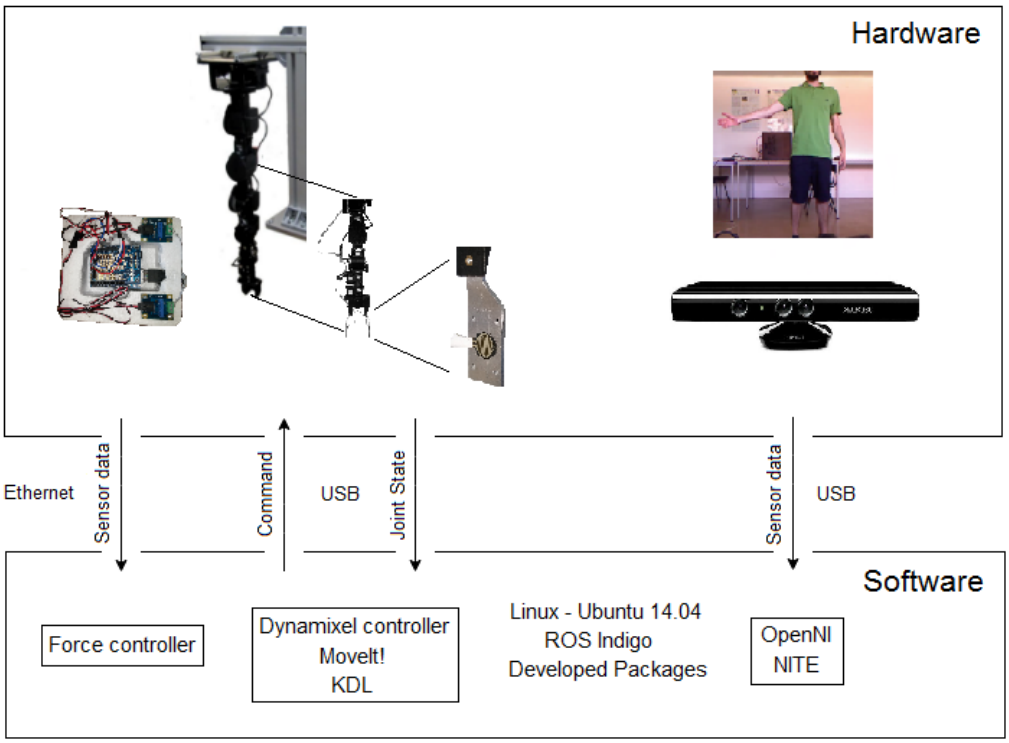


Figure 6.18: Integrated system architecture.

6.5 Overall ROS Architecture

The final overall ROS architecture illustrated in figure 6.19 shows a brief overview of all the developed software. The main nodes and topics related to vision are represented with the colour red. A node to manage the user hand position to be sent to the robot with the appropriate frame transformation, and a node to detect the object that works as a signal to start the object transfer task. The corresponding nodes and topics adorned with the colour green are related to the force readings and gripper actions based on those reads. There is a node to manage the opening and closing phases and a node to control gripper actions inside each phase. The manipulator control developed nodes and topics are represented with the colour blue. One node contains all the information about the servos and there is also a node for Cyton foward kinematics and one for Cyton inverse kinematics.

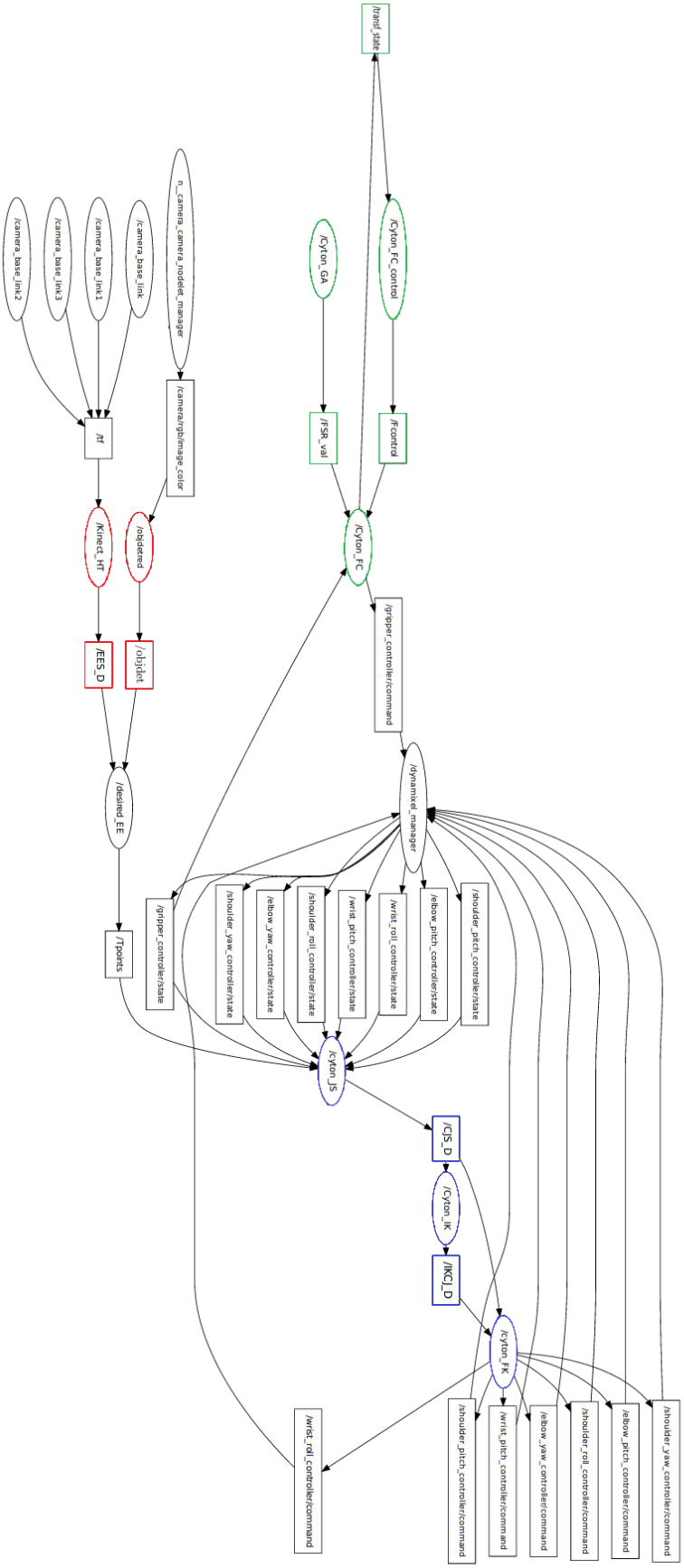


Figure 6.19: Final overall ROS structure.

Chapter 7

Conclusions and Future Work

This chapter presents the final conclusions of the work and perspectives of future developments.

7.1 Conclusions

In the field of Human-Robot Interaction, in endless future joint-action scenarios humans and robots will have to interact physically in the interest of a profitable collaboration. This dissertation makes a contribution to this topic by accomplishing a human-robot object transfer task. The transfer of objects between humans and robots is a major way to coordinate activity and cooperatively perform advantageous work. The main objective of this work is to develop algorithms to control the Cyton Gamma 1500 robotic arm to achieve this task. This work is divided into three fundamental parts: low level control of the robotic arm, pre-interaction approach using vision and interaction by contact using force sensors. The low level control of the robotic arm is accomplished by making use of the dynamixel packages and the MoveIt! software incorporated with a kinematic and dynamic library. The pre-interaction approach using vision is successfully achieved using a 3D sensor. This sensor allows a user's hand to be detected as well as an object to be used as a signal for the transition between pre-interaction and interaction phases. The 3D sensor accuracy has proved to be sufficient, and the obtained small error is compensated by the user adaptation. Concerning the interaction by contact approach, despite the misfortune occurred with the ATI sensor mini 40, the implementation of force sensitive resistors made it possible to achieve the proposed goal. Although all the chosen objects have been transferred, the object properties and geometries affected the success rate.

At software levels, ROS proved to be very powerful and useful, being the link between different thematics. It highly supports in the implementation, control and management of several processes. Although the hardware limitations and the lack of maintenance of the robot led to various difficulties throughout the work, the goals proposed are successfully accomplished, even if some refinements, improvements and extensions are required to improve the system's performance and ability to cover a wider range of similar tasks.

7.2 Future Work

Throughout the execution of this project it was noticed that the system could be improved by addressing new features or improving the implemented ones. Given the current state of development, perspectives of future work point in the following directions:

1. A better robotic manipulator should be used for this kind of task, because the current arm is not stiff enough, which leads to a bad performance of the force propagation between human, object and force sensor;

2. A torque/force sensor should be implemented in the wrist joint of the manipulator arm to have a better perception of the force propagation;

3. The gripper's fingers should have more force sensitive resistors to improve the diversity of objects that can be handed over, and also to look forward to new possible conclusions, such as objects dimensions, objects geometry reconstruction based on the contacts made and multiple objects handed over at once;

4. A complete force analysis, taking into consideration object friction and weight;

5. A database with different objects properties, so that the 3D sensor can recognize the objects and thus adjust the gripper position, orientation and force required to hold these objects.

Bibliography

[1] Markus Huber et al. "Human-robot interaction in handing-over tasks". In: *Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN* (2008), pp. 107–112. ISSN: 1944-9445. DOI: 10.1109/ROMAN.2008.4600651.

[2] Alexandre Campeau-Lecours et al. "A time-domain vibration observer and controller for physical human-robot interaction". In: *Mechatronics* 36 (2016), pp. 45–53. ISSN: 09574158. DOI: 10.1016/j.mechatronics.2016.04.006. URL: http://linkinghub.elsevier.com/retrieve/pii/S0957415816300277.

[3] Aaron Edsinger and Charles C. Kemp. "Human-robot interaction for cooperative manipulation: Handing objects to one another". In: *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication* (2007), pp. 1167–1172. ISSN: 1944-9445. DOI: 10.1109/ROMAN.2007.4415256.

[4] Justin W Hart et al. "Predictions of Human Task Performance and Handover Trajectories for Human-Robot Interaction [Extended Abstract]". In: *HRI Workshop on Timing in Human-Robot Teaming* (2015), p. 4.

[5] Michael A. Goodrich and Alan C. Schultz. "Human-Robot Interaction: A Survey". In: *Foundations and Trends® in Human-Computer Interaction* 1.3 (2007), pp. 203–275. ISSN: 1551-3955. DOI: 10.1561/1100000005. arXiv: arXiv:1011.1669v3. URL: http://www.nowpublishers.com/article/Details/HCI-005.

[6] Gabriel J. Garcia et al. "Survey of visual and force/tactile control of robots for physical interaction in Spain". In: *Sensors* 9.12 (2009), pp. 9689–9733. ISSN: 14248220. DOI: 10.3390/s91209689.

[7] Tiago Moura. "Development of a Dual-Arm Robotic System for Gesture Imitation". Master Thesis. University of Aveiro, 2015.

[8] Faraj Alhwarin, Alexander Ferrein, and Ingrid Scholl. "IR stereo kinect: Improving depth images by combining structured light with IR stereo". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8862 (2014), pp. 409–421. ISSN: 16113349. DOI: 10.1007/978-3-319-13560-1.

[9] C. S. Lovchik and M. A. Diftler. "The Robonaut hand: a dexterous robot hand for space". In: 2 (1999), 907–912 vol.2. ISSN: 1050-4729. DOI: 10.1109/ROBOT.1999.772420.

[10] H. Kawasaki, T. Komatsu, and K. Uchiyama. "Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand II". In: *IEEE/ASME Transactions on Mechatronics* 7.3 (2002), pp. 296–303. ISSN: 1083-4435. DOI: 10.1109/TMECH.2002.802720.

[11] Aaron Ladd Edsinger. "Robot Manipulation in Human Environments". In: *Organization* 1994 (2007), pp. 102–109. ISSN: <null>. DOI: 99.2007/edsinger.thesis.

[12] Willow Garage. *PR2 Overview*. URL: http://www.willowgarage.com/pages/pr2/overview (visited on 05/21/2016).

[13] Jim Mainprice et al. "Sharing effort in planning human-robot handover tasks". In: *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication* (2012), pp. 764–770. ISSN: 1944-9445. DOI: 10.1109/ROMAN.2012.6343844.

[14] *e-Tech Gadget*. URL: http://www.e-techgadget.com/tag/pr2 (visited on 06/10/2016).

[15] Aaron Edsinger and Charles C. Kemp. "Manipulation in human environments". In: *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS* (2006), pp. 102–109. DOI: 10.1109/ICHR.2006.321370.

[16] Aaron Edsinger. *Domo Robot*. URL: http://people.csail.mit.edu/edsinger/domo.htm (visited on 06/11/2016).

[17] Aldebaran Robotics. *Robot NAO*. URL: https://www.ald.softbankrobotics.com/en/cool-robots/nao/find-out-more-about-nao (visited on 06/12/2016).

[18] Florian Röhrbein, Germano Veiga, and Natale Ciro. "Gearing Up and Accelerating Cross-Fertilization between Academic and Industrial Robotics Research in Europe: Technology Transfer Experiments from the ECHORD Project ABC". In: *Springer Tracts in Advanced Robotics* 94 (2014), pp. 177–195. ISSN: 1610742X. DOI: 10.1007/978-3-319-02934-4.

[19] Aldebaran Robotics. *Robohub*. URL: http://robohub.org/robots-the-nao-humanoid/ (visited on 06/12/2016).

[20] N.P. Papanikolopoulos and P.K. Khosla. "Shared and traded telerobotic visual control". In: *IEEE International Conference of Robotics and Automation* April (1992), pp. 878–885.

[21] S. Shibata et al. "An Analysis of the Process of Handing Over An Object and Its Appliction to Robot Motions". In: *Proc. of the International Conference on Systems, Man and Cybernetics* (1997).

[22] Maya Cakmak et al. "Using spatial and temporal contrast for fluent robot-human handovers". In: *Proceedings of the 6th international conference on Human-robot interaction - HRI '11* (2011), p. 489. DOI: 10.1145/1957656.1957823. URL: http://portal.acm.org/citation.cfm?doid=1957656.1957823.

[23] T. Sato et al. "Robot Imitation of Human Motion based on Qualitative Description from Multiple Measurement of Human and Environmental Data". In: *Proc. of the International Conference on Intelligent Robots and Systems* (2003).

[24] K. Nagata et al. "Delivery by Hand between Human and Robot based on Fingertip Force-Torque Information". In: *Proc. of the International Conference on Intelligent Robots and Systems* (1998).

[25] Patrizia Basili et al. "Investigating Human-Human Approach and Hand-Over". In: *Human Centered Robot Systems: Cognition, Interaction, Technology*. Ed. by Helge Ritter et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 151–160. ISBN: 978-3-642-10403-9. DOI: 10.1007/978-3-642-10403-9_16. URL: http://dx.doi.org/10.1007/978-3-642-10403-9_16.

[26] Robai. "Cyton Gamma 1500 Arm Specifications". In: (2015).

[27] Robotis. *Characteristics of Dynamixel*. URL: http://www.robotis.com/xe/dynamixel_en (visited on 05/28/2016).

[28] ROBOTIS. *ROBOTIS e-Manual v1.27.00*. URL: http://support.robotis.com/en/product/dynamixel/mx_series/mx-64.htm (visited on 05/28/2016).

[29] Jungong Han et al. "Enhanced computer vision with Microsoft Kinect sensor: A review". In: *IEEE Transactions on Cybernetics* 43.5 (2013), pp. 1318–1334. ISSN: 21682267. DOI: 10.1109/TCYB.2013.2265378.

[30] ATI Industrial Automation. *F/T sensor: Mini40*. URL: http://www.ati-ia.com/products/ft/ft_models.aspx?id=Mini40 (visited on 03/12/2016).

[31] Automation and Robotics. *Tactile sensing*. URL: http://www.southampton.ac.uk/~rmc1/robotics/artactile.htm (visited on 03/18/2016).

[32] Phidgets. *1121 User Guide*. URL: http://www.phidgets.com/docs/1121_User_Guide (visited on 03/19/2016).

[33] A. Koubaa. *Robot Operating System (ROS): The Complete Reference*. Studies in Computational Intelligence vol. 1. Springer International Publishing, 2016. URL: https://books.google.pt/books?id=wY2RCwAAQBAJ.

[34] Simon Puligny and Mondada Francesco. "ROS interface and URDF parser for Webots". In: *Master@EPFL* February (2014).

[35] A. Sucan Ioan and Chitta Sachin. *MoveIt!* URL: http://moveit.ros.org (visited on 05/10/2016).

[36] R. Smits. *KDL: Kinematics and Dynamics Library*. URL: http://www.orocos.org/kdl (visited on 04/11/2016).

[37] OpenNI organization. *User Guide*. URL: http://www.openni.org/documentation (visited on 05/18/2016).

[38] Marcus Liebhardt. *openni tracker*. URL: http://wiki.ros.org/openni_tracker (visited on 05/19/2016).

[39] F. Han et al. "Space-Time Representation of People Based on 3D Skeletal Data: A Review". In: (2016), pp. 1–21. arXiv: arXiv:1601.01006v1. URL: http://arxiv.org/pdf/1601.01006.pdf.

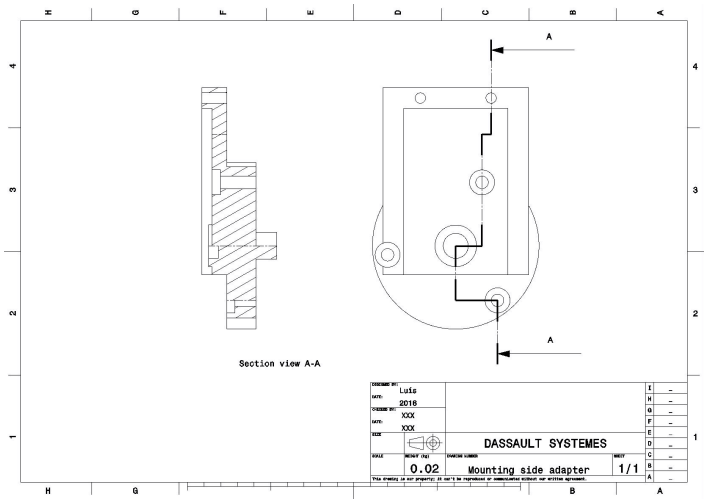
[40] Lentin Joseph. *Mastering ROS for Robotics Programming*. 2015, p. 451. DOI: 10.1007/s13398-014-0173-7.2. arXiv: arXiv:1011.1669v3.

[41] Akansel Cosgun, B Martin, and Henrik I Christensen. "Accuracy Analysis of Skeleton Trackers for Safety in HRI". In: (2013), p. 2013.

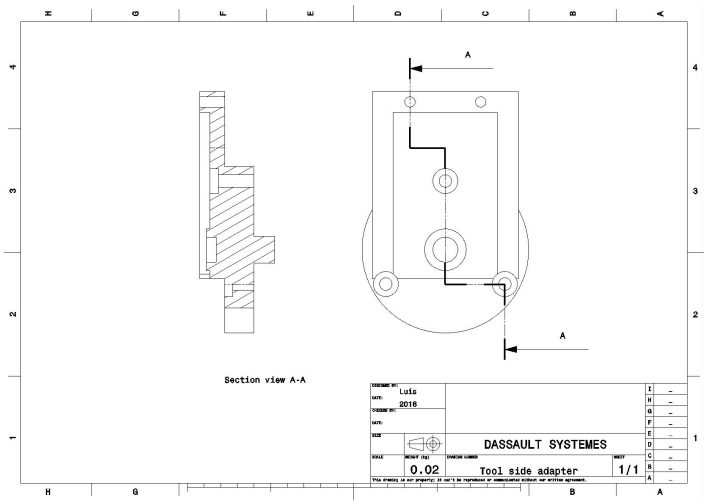
Appendix A

Developed Components to Install the Sensor in the Arm

Figure A.1 presents the entire sheet of the developed components.



(a) Mounting side adapter.



(b) Tool side adapter.

Figure A.1: Support blueprint to integrate the sensor with the arm.

Appendix B

User Guide

In this appendix is outlined the procedure required to use the packages developed throughout this work. The packages have been used and developed under Ubuntu 14.04 and ROS indigo.

1. Download the packages: The packages can be downloaded from the following repository: `https://github.com/Luis93A/cyton1500.git`;
2. Compile the packages;
3. Launch the packages and run the nodes as explained in the "readme" file.