FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Human-Robot interaction system in non-planar environments using Projection Mapping

Tiago Ferreira da Cunha

Master in Electrical and Computers Engineering

Supervisor: Prof. Dr. Armando Jorge Miranda de Sousa

Co-Supervisor: Dr. Germano Manuel Correia dos Santos Veiga

July 29, 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | **2014/2015**

A Dissertação intitulada

"Human-Robot Interaction System in Non-planar Environments Using Projection Mapping"

foi aprovada em provas realizadas em 23-07-2015

o júri

Presidente Professora Doutora Maria Margarida de Amorim Ferreira
Professora Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Artur José Carneiro Pereira
Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Professor Doutor Armando Jorge Miranda de Sousa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Tiago Ferreira da Cunha

Autor - Tiago Ferreira da Cunha

Faculdade de Engenharia da Universidade do Porto

Resumo

Human-Robot Interaction (HRI) é uma área de estudo que tem recebido atenção considerável na comunidade académica e na indústria. Os robots estão a preencher cada vez mais papéis na sociedade de hoje, realizando tarefas que são cada vez mais complexas e desafiantes a nível social. Consequentemente, é importante perceber como funciona esta interação, como medir a sua eficiência, onde melhorar, e como desenhar um sistema capaz de funcionar ao mais alto nível cumprindo todas as necessidades interativas que a tarefa precise e que cumpra todas as restrições impostas pelo ambiente.

Esta tese propõe alguns princípios aplicáveis numa aplicação genérica de HRI, como também discute uma implementação num *setup* de validação (ABB IRB140) com o objetivo de integrar o sistema produzido no projeto CLARiSSA. O sistema de validação foi modelado com auxílio do Eyeshot e ao Gazebo usando ficheiros CADs (STLs, IGSs). Também foram realizadas calibrações de câmara e projetor.

Uma interface foi desenvolvida em Eyeshot de forma a permitir *feedback* do operador, quer na forma de *projection mapping* ou com uma *Graphical User Interface* (GUI) que permite ao utilizador(es) interpretar num computador informação proveniente do ambiente real utilizando o Eyeshot como ferramenta de modelação. O sistema desenvolvido tem diversos módulos e um dos mais importantes é a possibilidade de adicionar novos objetos dinamicamente em *run-time*. Juntando isto a outro sistema que consiga identificar objetos e informar a interface da sua pose aumentamos a sua capacidade de adaptação a vários cenários.

O sistema desenvolvido apresenta uma interface intuitiva para uso por operadores especializados, reduzindo a necessidade de mudança de contexto, ou seja, reduz a necessidade de ter o operador a memorizar o estado do sistema e a reproduzi-lo depois. Também reduz erros de operação e permite cooperação entre diversos utilizadores e robôs. Em relação ao projecto CLARiSSA, onde esta tese foi aplicada, a interface promove uma abordagem interessante com uso de projection mapping. Embora existam erros de projeção, devido às calibrações, o sistema projeta imagens com 2 cm de erro a meio metro de distância da superficie. Finalmente, a empresa SARKKIS pode usar o sistema proposto adaptando o programa em Eyeshot porque a versão final do programa usou Gazebo e recorreu ao ROS para as comunicações, que são funcionalidades experimentais em Windows e que não são trivialmente convertidas para Eyeshot e C#.

Abstract

Human-Robot Interaction (HRI) is a field of study that has received considerable attention in the academic community and in the industry. Robots are filling more and more roles in present society, performing tasks that are more complex and more socially challenging. It is therefore important to define this interaction, how it works, how to measure it, how to improve it and how to design a system capable of fully performing all the interactive necessities required by the tasks or restrained by the environment.

This thesis proposes generic architecture principles for an HRI application, as well as an implementation on a validation setup (ABB IRB140), with the aim of further implementing the system produced in project CLARiSSA. The validation system was modeled in Eyeshot and Gazebo through the use of CADs files (STLs, IGSs). Camera and projector calibrations were also performed.

An interface was developed in Eyeshot to enable user feedback, either in the form of projection mapping or with a Graphical User Interface (GUI) that allows the user(s) to visually interpret the real scene in a computer resorting to Eyeshot. The system has different modules, and one of the most important is the ability to dynamically load new objects into the world. Coupling this with a system that identifies the objects and informs the interface of what the object is and its position increasing the system's flexibility in handling various scenarios.

The system design presented provides an intuitive interface for industry operators by reducing the need to context-switch and hence reduce errors. It also allows for cooperation between several users and robots. As for Project CLARiSSA, the provided interface promotes an interesting approach to the projection mapping paradigm. While the existing errors are all due to the calibrations, the system accurately projects images with 2 cm error at half a meter distance from the target. Lastly, SARKKIS can use the proposed system design up to an extent. This means that, some future work has to be done in order to be able to integrate the system's features in Eyeshot, because the final version was produced in Gazebo and resorting to ROS, which are features that are not trivially transformed to Eyeshot and C#.

Keywords: *Human-Robot Interaction*, *Augmented Reality*, *Projection Mapping*, *Generic Architecture*, *Robotics*, *Project CLARiSSA*, *ABB*

Acknowledgments

First and foremost, I would like to thank my supervisor, Professor Dr. Armando Jorge Sousa, for sharing expertise and his sincere and valuable guidance, and for all the encouragement extended to me.
I place on record, my sincere thank you to Dr. Germano Veiga, co-supervisor of the dissertation, for the continuous encouragement and help provided. I am also grateful to Luis Rocha from the robotics lab and João Silva from Sarkkis for their availability and kindness.
I am extremely thankful and indebted to the Lab 001 family - Hugo Costa, Peter Cebola, Ricardo Carvalho, Susana Neves and Valter Costa - for their friendship and support. It's thanks to them I've gotten this far.
To my parents and sister, my sincere thanks for the unceasing encouragement, support and attention during the duration of this dissertation.
To Denise Neves Gameiro, I am grateful for the companionship, the friendship and the encouragement you gave me (I wouldn't have made it this far without you).
I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand to make this work come alive.

Tiago Ferreira da Cunha

"If the only tool you have is a hammer,
you tend to see every problem as a nail."

Abraham Maslow.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

2D	Two-dimensional Space
3D	Three-dimensional Space
AR	Augmented Reality
CAD	Computer Assisted Design
COLLADA	Collaborative design activity
DOF	Degrees of Freedom
DXF	Data Exchange Format
FOP	Field of projection
FOV	Field of view
GUI	Graphical User Interface
HRI	Human-Robot Interaction
HSL	Hue-Saturation-Lightness
HSV	Hue-Saturation-Value
IDE	Integrated Development Environment
ISO	International Organization for Standardization
LOA	Level of autonomy
LUT	Look-up table
RIA	Robot Institute of America
ROS	Robot Operative System
SAR	Spatial augmented reality
SDF	Standard Data file
SRDF	Semantic Robot Description Format
URDF	Unified Robot Description Format
YAML	Yet Another Markup Language

Chapter 1

Introduction

The term robotics refers to the study of robots and it first appeared when Isaac Asimov, scientist and science fiction writer, employed it in one of his works to describe the technology used to make robots [1]. In the mid 20th century, the field of robotics developed even further causing diverse changes in the way work was done. The robots are faster, do not need to stop to rest, and rarely fail. In fact, the human element is considered the weak link in the operation of automatized systems.

Nowadays, co-operation between elements of the world (humans and robots) is of increased priority because these machines are being used in a growing range of applications. The demand for high flexible robotic systems, which must have the capability of adapting themselves to their surrounding, are rapidly increasing. In industrial environments, where the robot's work is well-known and structured settings, that need decreases, but if the world becomes too complex, a human will always need to be present to interact with it.

There is also an increasing trend of robots being moved into environments originally designed for human use. This has, during the past few years, led to increased interest for the field of dual-arm manipulation. "Robot manipulation in its basic forms is a well studied field that has seen remarkable developments in the last 50 years, but the added complexity of dual or multi-arm manipulation presents many challenges that may not be present in the single manipulator case" [2]. Even if that robot is considered fully autonomous the human might be supervising its goals or results, changing them, approving them or even adding new ones without physically interfering with him [3]. This higher complexity means that dual arm manipulation requires more advanced system integration, high level planning and reasoning, while also solving the problems of how to create an intuitive interaction.

This interaction is not only important for the humans safety, but it also boosts the team's productivity, leaving the hazardous, tedious chores to the robot and letting the human perform other high-level cognitive tasks. More will be detailed about this topic in subsequent chapters.

1.1 Motivation

Industrial environments often have non-optimal working conditions, which are considerably harsher than white-collar jobs, and may pose a threat to the health of workers. In an industrial environment, humans are exposed to extreme conditions, which, depending on the job, can be very severe (workers are exposed to extreme environments, such as temperature, radiation, and others) [4].

Autonomous machines are good candidates to replace human workforce at high-risk jobs, both due to ethic concerns and the machine's imperviousness to harsh conditions. Additionally, the flexibility, adaptability and precisions of robotic manipulators can avoid common industrial problems.

In order to integrate the robot in a human team, one must first solve questions related to their interaction. In order to increase the performance of any complex system, in many domains like transportation, assembly, maintenance and others, it is necessary to understand the nature of interactions between the human and machine components [5].

1.2 Objectives

The objective of the current work is the development of advanced human robot interaction systems, with mixed initiative and intuitive interaction even for the worker unfamiliar with robotics. Tasks will include the development of interaction devices, such as augmented reality for operator feedback and teaching systems for part positioning and correction.

The augmented reality must be projected on any sort of display, as it is to apply in an industrial environment. This means that a solution for non-planar projection and acquisition will also be presented. This system will also be integrated in a simulator with a file serialization to model the aggregate scene and manipulator. All of the above, decreases the need to perform context-switches because all important information will be in the human operator line-of-sight.

1.3 Document Structure

This chapter presented a brief introduction and motivation to the Dissertation, and ends with the proposed objectives. Chapter 2 asserts the necessary tools and familiarity required for the reading of this document, such as literature reviews and related work. Chapter 3 introduces statements regarding the problem at hand and its application in project CLARiSSA. The next chapter (chapter 4) describes the main premise behind this work which is the identification of a generic Human-Robot interaction architecture applied to beam weld applications. Next, chapter 5 derives the previous architecture into the solution implemented, while discussing the problems and obstacles encountered. Finally, chapter 6 discusses conclusions withdrawn from the work presented and elaborates on future work that can be used to improve upon the architecture implemented.

Chapter 2

State-of-Art

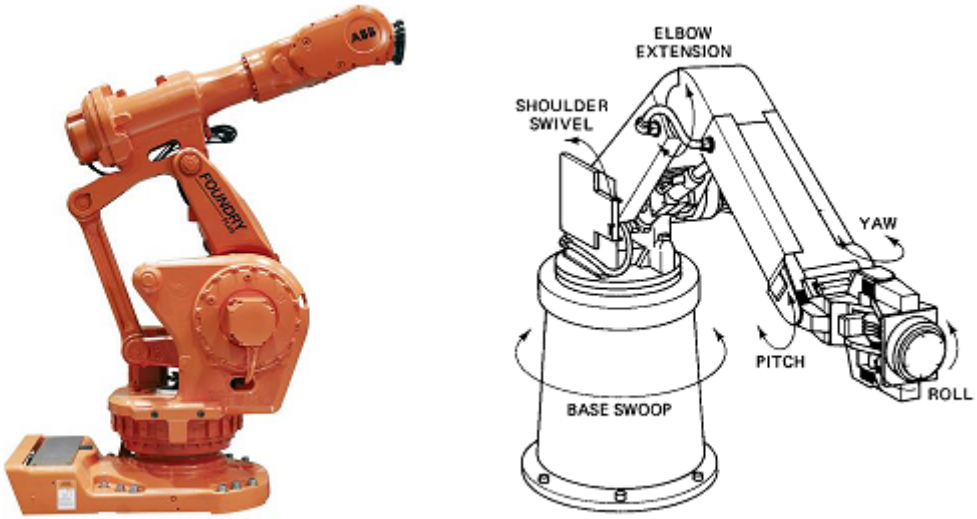
2.1 Robotic Manipulators

Robot Institute of America's (RIA) definition of robot manipulator is:

Definition 1. *A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed functions for the performance of a variety of tasks.*

Through this definition, it is possible to see the interest it brings to the modern industry because it can reduce labor cost and improve human working conditions [6]. Usual applications of robot manipulators are in welding, painting, assembly and pick and place routines. This technology first appeared in 1954 with the first programmable robot by George Devol, as an union of two technologies (teleoperators and milling machines). A literature review of robotic manipulators constitution will be presented, and several key concepts that might prove interesting when dealing with these manipulators will be studied in the following sections.

According to ISO 8373 standard, a manipulator is a machine in which the mechanism usually consists of a series of segments, jointed or sliding relative to one another usually in several degrees of freedom (DOF). A manipulator has links connected by joints to form a kinematic chain. Each joint represents the connection between two links. There are physical aspects and concepts regarding industrial manipulators such as the way joints are implemented, accuracy, repeatability and the way the tool is attached but those will not be addressed in this work because they are not interesting in this scope.



(a) Real robotic arm from ABB (b) Robotic arm representation

Figure 2.1: Robotic Arm parts and representation.

Looking at Figure 2.1, it is not intuitive to understand that the mechanical arm is just one component in a robotic system. Section 2.1.3 resumes common kinematics arrangements and introduces concepts that will be used in section 2.2.

2.1.1 Dual-arm manipulators

Single arm robots can't do their roles in tasks that involve two end-effectors (for instance assembling parts). It's for these specific tasks that a dual-arm robotic should be used [7].

Figure 2.2: Dual Arm industrial manipulator.

2.1.2 End-Effector

It is the end-effector or tool that actually performs the task. A great deal of research is devoted to study new tools design [8]. The joints between the arm and the end-effector are referred to as the wrist and in most applications they are usually revolute.

2.1.3 Mathematical Modeling of Robotic Manipulators

Mathematical modeling of robotic manipulators is a very extensive topic. Therefore only a brief overview of some topics will be covered, in order to contextualize later concepts. To accomplish a generic task some topics must be solved first:

- Forward kinematics: Calculation of the position and orientation of the end-effector in terms of the joint variables [9].
- Inverse kinematics: Refers to the use of the equations of a robot to determine the joint parameters that provide a desired position of the end-effector [9].
- Path planning: Defining a path in task space to move a robot to goal position while avoiding obstacles is called path planning for the robot joints[6].
- Independent joint control: Once the path is planned and reference trajectories are specified, the manipulator must track them by controlling each joint to follow a specific path, velocity or force. That operation is called Independent Joint Control.
- Sensorization model: Cameras are reliable and cheap sensors used in many robotic applications. They can be used to measure robot localization and find other objects position. We can also control the motion of the manipulator relative to the final destination [6, 10]. In section 2.2 this will be elaborated.

There are more problems than those referred here, such as velocity kinematics, dynamics, multivariable/force control and geometric nonlinear control.

2.1.3.1 Homogeneous Transformations and Quaternions

Another set of tools relevant to future topics are the homogeneous transformations. Transformations matrices like equation 2.1 are a representation of a rigid motion combining rotation and translation.

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

where R $\in$ SO(3), d $\in \mathbb{R}^3$ (SO(n) is the special orthogonal group of order n).

These matrices are used to perform coordinate transformations, analogous to rotational transformations [6]. In order to calculate 3D rotations quaternions are also needed. Quaternions are

a number system and can be defined as the quotient of two directed lines in a three-dimensional space or equivalently as the quotient of two vectors [11]. A quaternion is a quad-tuple that defines an element in $\mathbb{R}^4$ where:

$$q = (q_0, q_1, q_2, q_3) \tag{2.2}$$

where q_0, q_1, q_2, q_3 are real numbers.

Basic quaternion properties are as follows:

- Complex Conjugate: The complex conjugate of q is denoted q^* and $(p.q)^* = q^* p^*$.
- Length: The length/norm of a quaternion is given by $N(q) = \sqrt{q^* q}$.
- Unit Quaternion: A unit quaternion has norm equal to 1.
- Inverse: The quaternion inverse is given by the expression $q^{-1} = \frac{q^*}{N(q)^2}$
- Rotation Operator Geometry: The quaternion rotation operator $v \rightarrow w$, where the mapping from quaternion to rotation matrix is done by:

$$w = qvq^* = (q_0^2 - N(q)^2)v + 2(q.v)q + 2q_0(qxv) \tag{2.3}$$

2.2 Sensory Systems

To create a system capable of interacting with its surroundings it must first be able to perceive the environment. In robotic manipulators (and robots in general) this is done by setting up a camera on top of said robot. In order to simplify the equations in this section, the coordinates of the objects are relative to a camera centered coordinate frame as shown in Figure 2.3

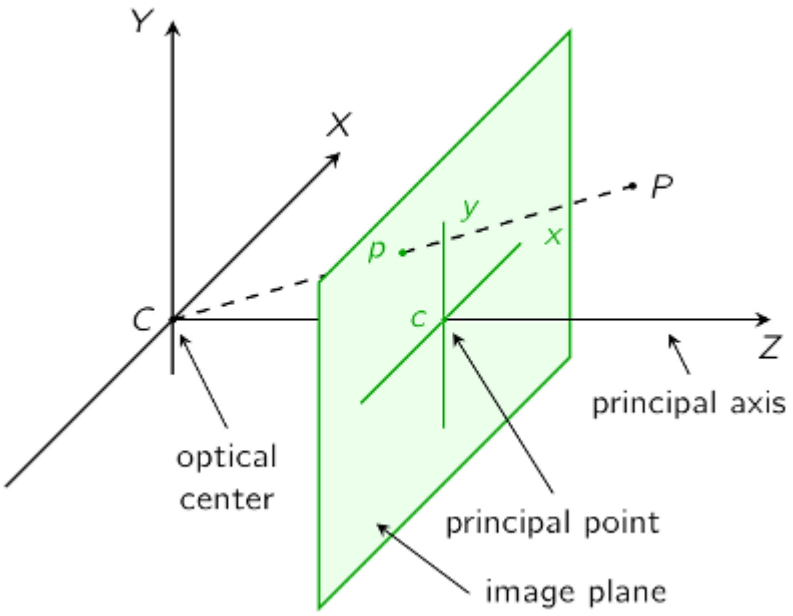


Figure 2.3: Camera coordinate frame [12].

With this assumption for the camera frame, a point in the image plane will have coordinates (u, v, λ). We can now use *(u, v)* as the image plane coordinates. Using the same notation as referred by W. Spong [6] where P is a point in the world with coordinates *(x, y, z)* relative to the camera frame and p is the projection of P onto the image plane and has coordinates *(u, v, λ)*. The pinhole theorem states that the points P and p will be collinear (can be seen in Figure 2.3):

$$k * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ \lambda \end{bmatrix} \tag{2.4}$$

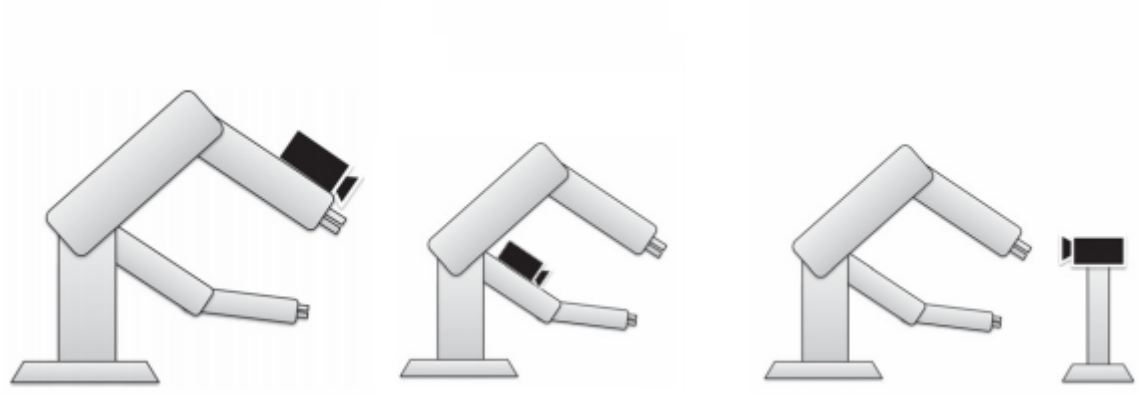
that represents the following equation system.

$$\begin{cases} kx = u \\ ky = v \\ kz = \lambda \end{cases} \tag{2.5}$$

This gives $k = \frac{\lambda}{z}$, $u = \lambda * \frac{x}{z}$ and $v = \lambda * \frac{y}{z}$ which are the equations for perspective projection. [13]. In order to relate pixel coordinates *(r, c)* relation to the real world coordinates (u, v) a relationship must be found first. Usually the origin of the pixels are at a corner rather than the center [14]. The relationship between image plane coordinates and pixel array coordinates is $-\frac{u}{s_x} = (r - o_r)$ and $-\frac{v}{s_y} = (c - o_c)$, where s_x and s_y are pixel coordinates and o_r and o_c are the center coordinates. The result of s_x and s_y must be truncated or rounded up because pixel values are integers.

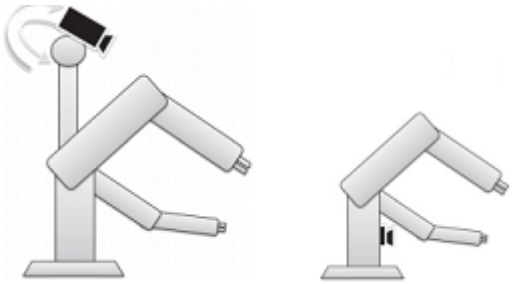
2.2.1 Camera systems applied to robotic manipulators

Visual systems can be characterized according to where their point of view is placed. Figure 2.4 shows typical configurations. It is also possible to employ a combination of the configurations [15] (Figure 2.4).



(a) End-Effector or active arm

(b) Fixed in the workspace



(c) Active head or fixed torso

Figure 2.4: Types of camera configurations for dual-arm manipulators [15].

- End-effector: This configuration can also be called "eye-in-hand". The camera is setup in the robot's end-effector. This configuration provides a predefined geometric relationship between the position and orientation of the camera with respect to the arm.
- Fixed in the workspace or in the torso: This configuration provides a stable place for the vision system to observe the scene.
- Active head or active arm: This configuration provides the vision system with a limited flexibility with respect to the arm's reference system.

The most commonly employed cameras in these kind of systems are RGB-Ds (RGB-Depth cameras) [16], because they provide additional depth information.

2.2.2 Camera Calibration

Another important problem in autonomous manipulation is synchronizing the robot's internal representation of the world with the real world by calibrating its sensors [17]. The objective of camera calibration is to determine all of the parameters that are necessary to relate the pixel coordinates *(r, c)* to the *(x, y, z)* world coordinates of a point in the camera's field of view (FOV). When all of the parameters are known, an accurate prediction is possible relating the image pixel coordinates to the projection of *P(r, c)*.

In robotics applications, tasks are expressed in terms of the world coordinate frame [6]. Knowing the position and orientation of the camera frame relative to the world coordinate frame, the following expression is valid:

$$x^w = R_c^w * x^+ O_c^w \tag{2.6}$$

or:

$$x^c = R_w^c * (x^w - O_c^w) \tag{2.7}$$

Defining $R = R_w^c$ and $T = -R_w^c O_c^w$,

$$x^c = Rx^w + T \tag{2.8}$$

where R and T are called the extrinsic camera parameters.

Using perspective projection equations, the 3D mapping to pixel coordinates is obtained.

$$r = -\frac{\lambda * x}{s_x * z} + o_r \tag{2.9a}$$

$$c = -\frac{\lambda * y}{s_y * z} + o_c \tag{2.9b}$$

To determine (r, c) values of f_x, o_r, f_y, o_c must be known, where

$$f_x = \frac{\lambda}{s_x} \tag{2.10a}$$

$$f_y = \frac{\lambda}{s_y} \tag{2.10b}$$

The parameters o_r, o_c, f_x, f_y are constant for a given camera and do not change when the camera moves [6]. The first parameter to be determined is the image center which can be achieved through the vanishing points concept. This concept represents the intersection of three mutually orthogonal parallel set of lines. The image center is given by the center of the triangle formed by the three (as seen in Figure 2.5).

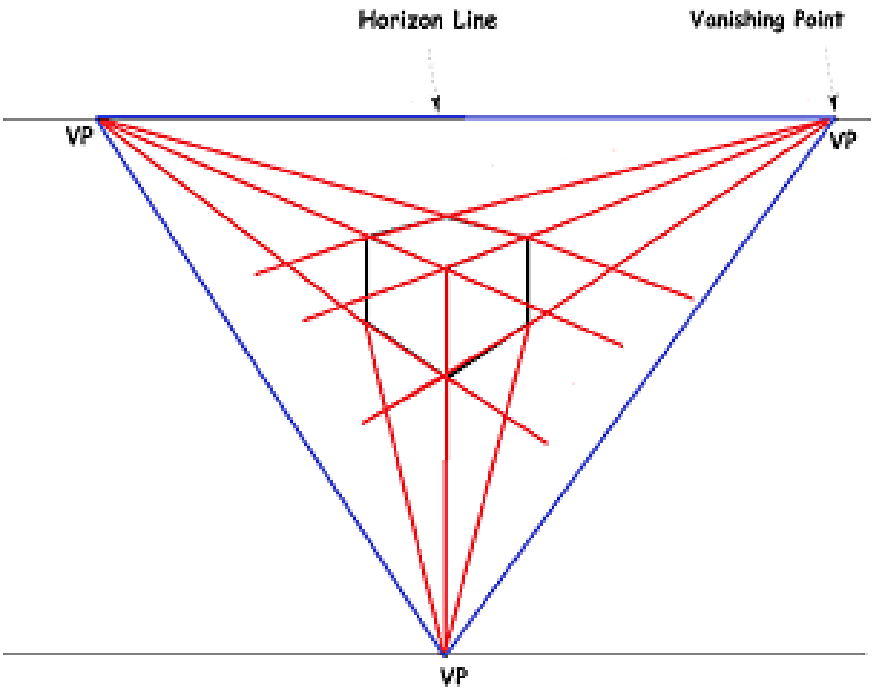


Figure 2.5: Vanishing point illustration. Adapted from [18].

Once the center is found, determing the remaining parameters is straightforward. The extrinsic parameters of the camera are given by:

$$R = \begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix}, T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{2.11}$$

The coordinates of a point in the world with respect to the camera frame are given by:

$$\begin{cases} x^c = & r_{11}x + r_{12}y + r_{13}z \\ y^c = & r_{21}x + r_{22}y + r_{23}z \\ y^c = & r_{31}x + r_{32}y + r_{33}z \end{cases} \tag{2.12}$$

Combining these equations with the intrinsic parameters equation to obtain:

$$\begin{cases} r - o_r = & -f_x \frac{x^c}{z^c} = & -f_x * \frac{r_{11}x + r_{12}y + r_{13}z + T_x}{r_{31}x + r_{32}y + r_{33}z + T_z} \\ c - o_c = & -f_y \frac{y^c}{z^c} = & -f_y * \frac{r_{21}x + r_{22}y + r_{23}z + T_y}{r_{31}x + r_{32}y + r_{33}z + T_z} \end{cases} \tag{2.13}$$

Since the coordinates of o_r and o_c are known, two transformations to simplify the current equation system can be applied. The first one is $r \leftarrow r - o_r$ and the second is $c \leftarrow c - o_c$. Now, for the points r_i, c_i, x_i, y_i, z_i:

$$r_i f_i (r_{21}x_i + r_{22}y_i + r_{23}z_i + T_y) = c_i f_x (r_{11}x_i + r_{12}y_i + r_{13}z_i + T_x) \tag{2.14}$$

Combining the N equations into the matrix equation $Ax = 0$:

$$A = \begin{bmatrix} r_1x_1 & r_1y_1 & r_1z_1 & r_1 & -c_1x_1 & -c_1y_1 & -c_1z_1 & -c_1 \\ r_2x_2 & r_2y_2 & r_2z_2 & r_2 & -c_2x_2 & -c_2y_2 & -c_2z_2 & -c_2 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ r_Nx_N & r_Ny_N & r_Nz_N & r_N & -c_Nx_N & -c_Ny_N & -c_Nz_N & -c_N \end{bmatrix} \tag{2.15}$$

and

$$x = \begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \\ T_y \\ \frac{f_x}{f_y} r_{11} \\ \frac{f_x}{f_y} r_{12} \\ \frac{f_x}{f_y} r_{13} \\ \frac{f_x}{f_y} T_x \end{bmatrix} \tag{2.16}$$

If $\bar{x} = \begin{bmatrix} \bar{x_1} & \bar{x_2} & \bar{x_3} & \bar{x_4} & \bar{x_5} & \bar{x_6} & \bar{x_7} & \bar{x_8} \end{bmatrix}$ is a solution for $Ax = 0$,

$$\bar{x} = k \begin{bmatrix} r_{21} & r_{22} & r_{23} & T_y & \frac{f_x}{f_y} r_{11} & \frac{f_x}{f_y} r_{12} & \frac{f_x}{f_y} r_{13} & \frac{f_x}{f_y} T_x \end{bmatrix}^T \tag{2.17}$$

k being and unknown scalar. To solve for the camera parameters W. Spong [6] demonstrates that by exploiting the fact that R is a rotation matrix (see 2.18 and 2.19):

$$\sqrt{\bar{x_1}^2 + \bar{x_2}^2 + \bar{x_3}^2} = \sqrt{k^2(r_{21}^2 + r_{22}^2 + r_{23}^2)} = |k| \tag{2.18}$$

$$\sqrt{\bar{x_5}^2 + \bar{x_6}^2 + \bar{x_7}^2)} = \sqrt{\frac{f_x}{f_y}k^2(r_{21}^2 + r_{22}^2 + r_{23}^2)} = |\frac{f_x}{f_y}k| \tag{2.19}$$

Therefore, choosing k such that $r(r_{11}x + r_{12}y + r_{13}z + T_x) < 0$. After that, the values of $k, \frac{f_x}{f_y}, r_{21}, r_{23}, r_{11}, r_{12}, r_{13}, T_x, T_y$. All that remains is to determine T_z, f_x, f_y, and the third column of R. R can be determined as the vector cross product of its first two columns. Returning to the projection equations

$$r = -f_x\frac{x^c}{z^c} = -f_x\frac{r_{11}x + r_{12}y + r_{13}z + T_x}{r_{31}x + r_{32}y + r_{33}z + T_z} \tag{2.20}$$

Using a similar approach as for the first eight parameters:

$$r(r_{31}x + r_{32}y + r_{33}y + r_{33}z + T_z) = -f_x(r_{11}x + r_{12}y + r_{13}z + T_x) \tag{2.21}$$

and solve for the remaining two parameters (T_z and f_x).

A pan tilt head can also exist. Usually it has two DOF: a rotation about the world z axis and a rotation about the head's x axis. Basically, it allows looking up and down, left and right. The rotation matrix is given by 2.22 where θ is the pan angle and θ is the tilt angle [14].

$$R = R_{z,\theta}R_{x,\alpha} \tag{2.22}$$

2.2.3 Camera and Projector Systems

A projector and camera work together as a stereo system, with the advantage of being able to choose an adequate pattern to find point correspondences. However, the calibration procedure must be adapted to the projector [**?**].

Augmented Reality (AR) is the technology of combining real word images and video with computer-generated information and imagery. A projector and a camera system is an example of AR, more specifically of a spatial augmented reality (SAR). AR has vast uses in medical, manufacturing and repair, military, and robotics [3]. The camera is used to detect objects and that information is then mapped to the projector.

2.3 Human-Robot Interaction

Human-Robot interaction (HRI) is a field of study dedicated to understand, design and evaluate robotic systems for use with or by humans [3, 5]. We can categorize the interaction in two categories:

- **Remote Interaction** The human and the robots are not co-located in the same physical or temporal space;
- **Proximate Interaction** The humans and the robots are located in the same space for example service robots.

This field of study appeared in the beginning of the twenty-first century. In 1992 the IEEE International Symposium on Robot Human Interactive Communication (RoMan) was first held and continues annually [19]. This field has been propelled by two major worldwide competitions: the *AAAI Robotics Competition and Exhibition* [20] and the RoboCup Search and Rescue. From these competitions the three main fields of application emerged. Those are robot-assisted search and rescue, assistive robots and space exploration. Service robots assist human beings in various tasks, from household chores to other dull, repetitive or difficult tasks. Industrial robots are programmed to carry out repetitive actions with high level of precision. With HRI, a designer can affect five attributes that influence the interactions between humans and robots [3]:

- Level and behavior of autonomy;
- Nature of information exchange;
- Structure of the team;
- Adaptation, learning, and training of people and the robot;
- Shape of the task.

These attributes are important because it is fundamental to understand the environment and how structured it is. A simple deviation from the world model can cause the system to fail. A possible compromise to avoid the domain restriction is to find the right balance between robot autonomy and human-robot interaction [5]. These attributes will be detailed in a later section.

The main idea behind HRI is to compensate uncertainty in order to avoid the robot to break down by using an appropriate mixture of knowledge about the environment, robot autonomy and user input. Communications factors, such as *delay*, *jitter* and *bandwidth* have profound effects on human performance. As such, HRI quality depends strongly on the communication channel's capacity to carry information [21]. Timing factors in the robot may time-oriented HRI metrics too, like in robots that do not interact at human rates.

Lastly, the human's role also affects the effectiveness of HRI. There are 5 roles [3] a human can possess: Supervisor, Operator, Mechanic, Peer and Bystander. The way the interface supports the interaction is a measure of its performance.

2.3.1 Task Metrics

In order to establish a functional communication channel, two requirements have to be fulfilled. The task definition can be divided into definition, which is the action of providing parameters as

inputs for task execution, and description. This last one results in the appointment of instructions to the robot about how to solve a specific task. The final requirement is the dynamic function allocation which encompasses the process dynamics.

The transmission of information from one peer to another occurs through a predefined protocol. In a human-robot team that transmission works differently. The sensors must not only be used for task execution, but also to recognize situations of possible interactions with other agents. It is important to avoid interpretation errors to safeguard human agents from hazardous situations [**?**]. Many problems cannot be solved by single Human/Robot agents working in isolation, because they do not have all the necessary resources or information [5].

2.3.2 Design attributes

Designing autonomy consists of mapping inputs from the environment into actuator movements, schemas or other acts. In the words of [3] "Autonomy is not an end in itself in the field of HRI, but rather a means to supporting productive interaction" . Table 2.1 shows a scale of levels of autonomy (LOA) for machines introduced by Sheridan and Verplank in 1978.

Table 2.1: Table of Levels of autonomy, defined by Sheridan and Verplank.

1	Human does it all;
2	Robot offers alternatives;
3	Robot narrows alternatives down to a few;
4	Robot suggests a recommended alternative;
5	Robot executes alternative if human approves;
6	Robot executes alternative; human can veto;
7	Robot executes alternative and informs human;
8	Robot executes selected alternative and informs human only if asked;
9	Robot executes selected alternative and informs human only if it decides to;
10	Robot acts entirely autonomously;

The scale introduced by Sheridan and Verplank suffered minor changes in order to adapt it to run in every task instead of the global process [22]. The autonomy scale of HRI can go from teleoperated operation (minimum autonomy) to peer-to-peer collaboration (full autonomy at appropriate times).

A second component is the manner in which information is exchanged between the human and the robot. Metrics for efficiency of exchanges include the interaction time required for intent and the time for the instructions to be communicated to the robot, the workload of the interaction, the amount of situation awareness produced by the cooperation and the extent of shared understanding between all parties [3, 21, **?**]. By analyzing the way information is exchanged, and the format of the medium, HRI can be divided in the structure shown in Table 2.2.

One of the objective of HRI is to produce systems that do not require significant training. For long term interactions, it is a must that the robot is able to adapt to its environment and learn from the participants.

Table 2.2: Types of HRI information exchange

Type	**Uses**
Visual Displays	GUIs or Augmented Reality interfaces
Gestures	Hands, facial movements and movement-based signaling of intent
Speech and natural language	Auditory speech, text-based responses
Physical Interaction and haptics	Used remotely in augmented reality or in teleoperation

To apply HRI to robotic manipulators, it is common to add either a camera, a projector or a combination of both. Projectors are used as HRI systems to produce augmented reality. The use of a projector makes it so the users don't need to use an accessory to see the augmented reality. Projector-based AR is called SAR. Projector installations can be fixed or portable [23]. A fixed projector installation is sold by Prodevco [24] and shown in Figure 2.6.

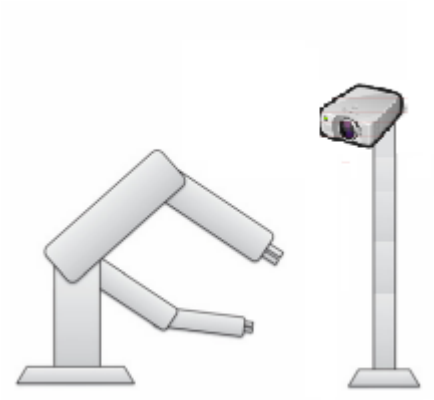


Figure 2.6: Fixed projector installation on dual-arm scenario.

Common accessories for HRI include a variety of devices worn by an operator, such as colored markers [25], instrumented gloves [26], motion tracking sensors or even pocket lasers which will be discussed in the next section.

2.3.3 Sensorizing non-planar environments

Sensorizing non-planar environments is a particularly challenging undertaking. In cases where HRI relies on a laser accessory, significant problems occur determining pose, angle of incision and direction. However, there are two different methods to solve this. The first, requires the laser to be adapted to emit three/four beams and then calculating the projection they make with the surface. The other requires the use of a laser beam that projects an arrow (giving us the sense of direction). On this last approach, determining the angle of incidence wouldn't be trivial, in which case, a 45^{o} degree assumption would have to be made.

2.3.4 Projection Mapping

"Projection Mapping uses everyday video projectors, but instead of projecting on a flat screen (e.g. to display a PowerPoint), light is mapped onto any surface, turning common objects of any 3D shape into interactive displays" [27]. Projection mapping can be used for advertising, live entertainment, gaming, and recently it has been used in engineering. Profitter [24] is an interesting example of this, because it allows the operator a more precise method of welding, less context-switching and faster operation times (the operator doesn't need to be sure that the welding is in the right place). Some commercially available software are presented in table 2.3.

Table 2.3: Projection Mapping software commercially available

Software Name	**Operative System**
DynaMapper	IOS
LPMT	Linux / Mac OSX / Windows
Avolites	Mac OSX / Windows
Arkaos GrandVJ XT	Mac OSX / Windows
MadMapper	Mac OSX
Blendy Dome VJ	Mac OSX
Resolume Arena	Mac OSX / Windows
Visution Mapio	Mac OSX / Windows
HeavyM	Mac OSX / Windows
VPT	Mac OSX
MXWendler	Mac OSX / Windows
Millumin	Mac OSX
MWM – Multi Window Mapper	Mac OSX
Mesh Warp Server	Mac OSX / Windows
Painting With Light	Mac OSX / Windows
Vioso	Windows
Coolux	Windows
d3 Technologies	Windows

With projection mapping it is possible to project an image or video allowing the creation of augmented reality scenes. "If we project an image in a non flat surface it will only look right from only one point of view and distorted from all other" [**?**].
Analyzing Figure 2.7 and discovering the geometric relationship between the user T, the projector P and the display surface D for any arbitrary three dimensional point V, where the projector and the user are defined by a camera model, it is possible to determine the point of view of the lens of the projector.

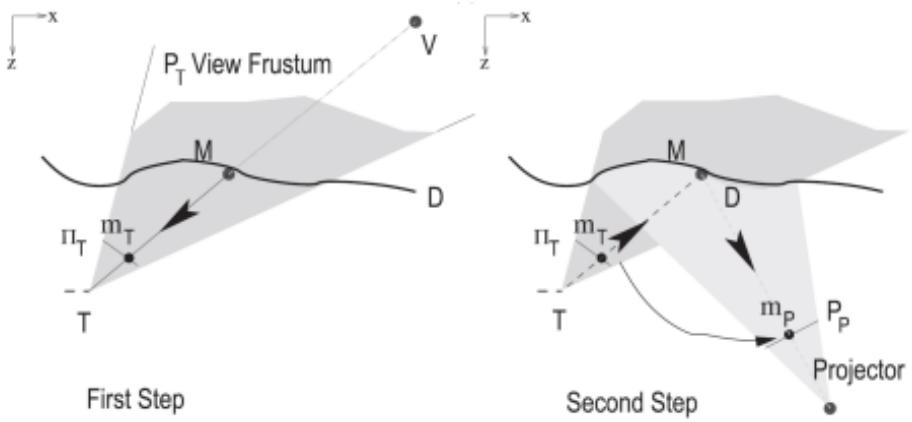


Figure 2.7: Scene rendering process [**?**].

The transformation process can be split into two tasks. The first is the calculation of the projected image on the display surface from the user point of view (forward mapping). The second is the calculation of the projector's view of the first projection (backwards mapping).

To project in a flat surface, a virtual representation of the world is necessary, and the result image of this process is related by an homography [**?**].

A transformation of the projective space is a mapping M: $P^3 -> P^3$ defined as:

$$\begin{bmatrix} s \\ u \\ v \\ w \end{bmatrix} \mapsto \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ m41 & m42 & m43 & m44 \end{bmatrix} \begin{bmatrix} s \\ u \\ v \\ w \end{bmatrix} \tag{2.23}$$

The (4,4) matrix is called the homogeneous transformation matrix of M. The (3,3) matrix consisted of m11,m12,m13,m21,m22,m32,m31,m32,m33 is the Rotation matrix while the last column represents the translation. Homogeneous transformations allow the conversion of a point's coordinates into another point coordinates.

$$PointA = H^{pointA}_{pointB} * pointB \tag{2.24}$$

2.4 Discussion

In this chapter, a state-of-art was presented for industrial manipulators, HRI and its applications in industry. Regarding manipulator modeling, the next chapter will present the test setup used for the dissertation and will build from the concepts studied in section 2.1. Lastly, the generic HRI architecture that will be developed and evaluated by the metrics presented in section 2.3 and will use projection mapping as a primary communication tool. This concepts are essential to the understanding of following chapters, since the work was built on top of them. The projection mapping and homography concepts are of particular interest to model the system.

Chapter 3

Context

3.1 Project CLARiSSA

The CLARiSSA demonstrator makes part of the SMERobotics project and is focused on a collaborative dual-arm robot for assembly applications in steel fabrication. It was first presented at Automatica trade fair 2014 by SARKKIS [28] and INESC Porto (Figure 3.1, 3.2). The SMERobotics consortium (European Robotics Initiative for Strengthening the Competitiveness of SMEs in Manufacturing) aim to create SME-suitable robots that assist in managing uncertainty by symbiotic Human-Robot-Interaction and embedded cognition going beyond flexibility by semantic integration [29]. SARKKIS is a partner of the SMERobotics consortium and focuses on innovation mechatronics software with a portfolio of products for off-line programming of robots for structural steel fabrication.

(a) CLARiSSA welding figure

(b) Benefits and innovation

Figure 3.1: Poster presented at Automatica 2014.

Welding is the process of joining two or more materials by heating, by applied pressure, or both [30]. The surfaces of the materials are fused to form a single unit. Welding dates back to the earliest days of metalworking, and continues to be widely applied today due to its cost effectiveness, reliability, and safety. When compared with other joining methods, such as riveting and bolting, welded structures tend to be stronger, lighter-weight, and cheaper to produce [30]. Automatic welding improves quality, productivity and reduces overall cost from the welding process. Clarissa aims to answer the needs of this market.

CLARiSSA is a dual-arm manipulator with automatic path planning, CAD based interpreter, parts recognition and localization, and HRI for quality control and completing missing information. CLARiSSA takes inputs from a CAD file, and performs welding operations on steel beams. However, it's not known whether the CAD files are correct or not, and therefore an HRI system could be a useful addition. The next section will detail this problem further.

Figure 3.2: CLARiSSA demonstrator in Automatica 2014.

Robotics is a highly interdisciplinary field of engineering, demanding knowledge of Operative Systems, Communications, Image Processing, Software Engineering, Task Planning, Machine Learning, among others. Therefore, it is a field that usually requires multidisciplinary teams and efforts. With that in mind, it's easy to see that tools that abstract certain aspects of reality (simulators) behind it, make the robotic programmer's life easier. It lets you focus on a specific problem, develop and test new ideas. Industrially, it saves time and therefore money. It also allows for quick experiments before implementation into real life manipulators. This is particularly important in HRI, in scenarios where robots physically interact with humans, such as medical robotics, HRI and serviceable environments. In these cases, since the environment isn't well structured (unpredictable patterns, too many variables to consider), it is imperative to test the safety, and robustness of any new algorithms by means of realistic and reliable simulations [31]. In light of this, a growing number of commercial and open-source simulators have been designed and implemented.

"There is an increasing trend of robots being moved into environments originally designed for human use" [2]. Dual-arm robots can outperform their single armed counterparts because they are

able to perform roles in which operators do the job with their two arms, for example in assembly applications. There are several factors that motivate the use of dual arm setups:

- Similarity to operator: as referred above, transferring the operator's bi-manual skills allows robots to operate in roles that require two hands;
- Manipulability: The second manipulator allows the ability to control both parts;
- Cognitive motivation: "Human-like dual-arm setups have been used to explore how human-like physical interaction relates to cognition. Likewise, in an HRI context, it has been argued that since humans have an intuitive understanding of bi-manual manipulation, the actions of a dual arm robot are easier for an observing human to understand and relate to" [2];
- Human form factor: Dual-arm systems occupy less space, and they are interchangeable with their human peers, which removes the need to redesign the workspace.

3.1.1 Eyeshot

Eyeshot is a 3D graphics and CAD control for .NET Framework which combines multiple data sources, input devices and CAD entity types. Eyeshot can be used as a simulator for robotics because of its ability to:

- Do heavy modeling operations asynchronously;
- Import/export standard CAD formats;
- Select between a number of preconfigured viewport styles and configurations at design-time;
- The Eyeshot control can be easily configured as 2D or as a 3D viewport;

The items above address the features Eyeshot is useful as a robotics simulator tool. It lacks however, usual features such as direct kinematics and inverse kinematics calculator, but they can be implemented. For example, Figure 3.3 shows the complex model of a motherboard taken from the website image gallery [32].

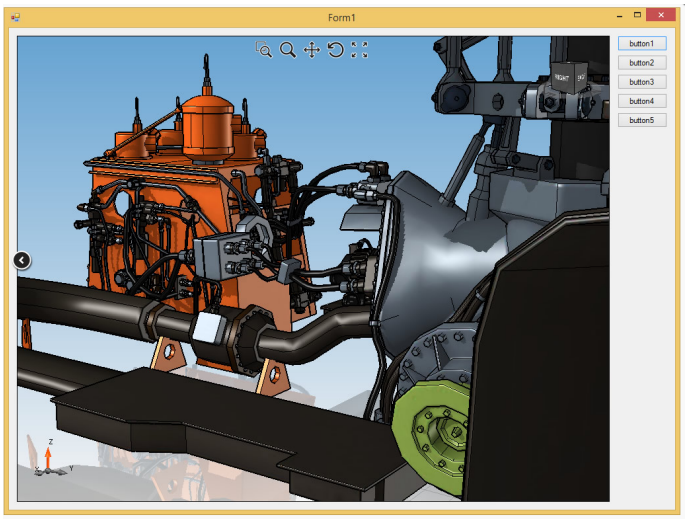


Figure 3.3: Eyeshot gallery image [**?**]

3.1.2 Gazebo

Gazebo is another simulator that offers the ability to simulate complex 2D and 3D environments (Figure 3.4). In alternative to Eyeshot, it is a cross-platform interface. Gazebo was used in the scope of this work to produce the final version of the prototype, because Eyeshot did not accurately represent the transformations imposed. This will be detailed further in section 5.2.

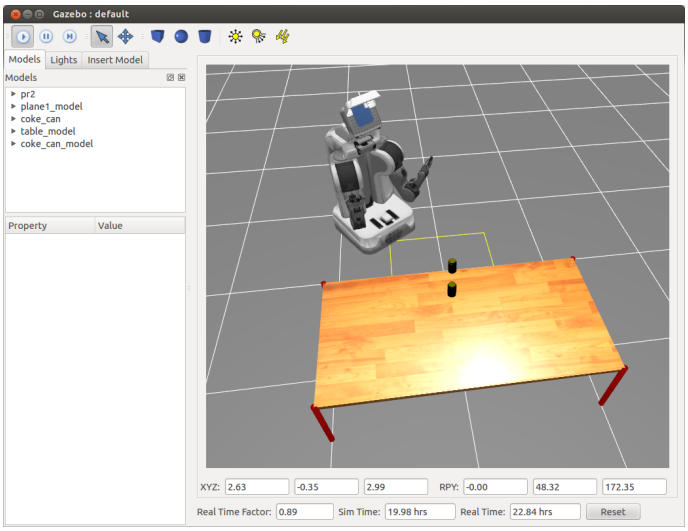


Figure 3.4: Gazebo simulating a robot [33].

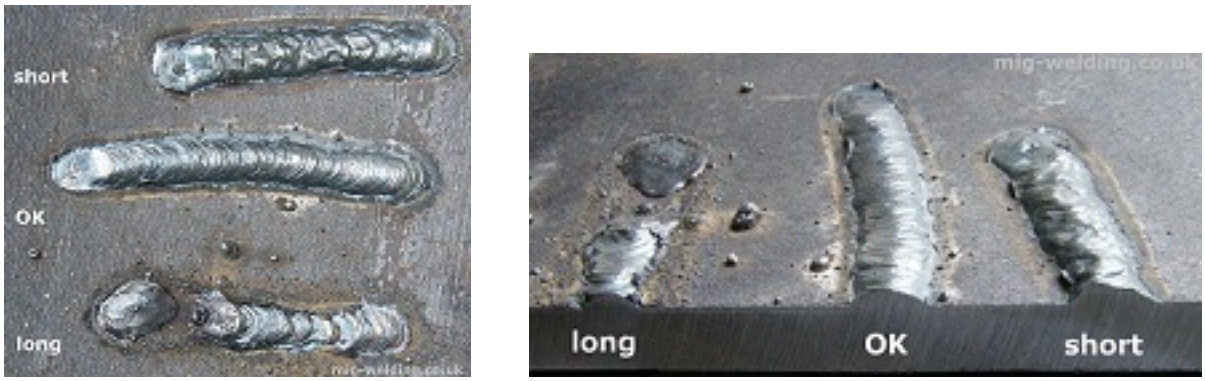
The scene modeling is done similarly to Eyeshot, but only .STL files are used for the scene.

3.1.3 Computer Assisted Design

Computer Assisted Design (CAD) systems came to combat traditional industrial robot programming, using the teach pendant which was a time-consuming task and required technical expertise [34]. CAD packages are becoming more powerful and widespread extracting motion information from a CAD data exchange format (DXF) files and converting it into robot commands. As CAD systems are a very powerful tool, they are widely used. For example, [35] use CAD specification with and open-source robotic grasping simulator and [36] where CAD is also used for cell specification. Eyeshot and Gazebo can import standard CAD formats (IGES, STEP, STL, others). The import in Eyeshot was performed to allow all of the standard CADs (using native functions), while in Gazebo only STLs were imported. The main difference between IGES/STEP and STL files is the amount of space the output files occupy and consequently the modeling speed. For example, while modeling a circle of 1 cm diameter the output STL file has up to 15 Megabytes while in Eyeshot it has 75 Kilobytes.

3.2 The Human-Robot interaction problem in welding applications

Up to this date, the vast majority of the effort employed has been spent on developing hardware and software that could expand the robots' capabilities instead of developing intuitive controls and displays. In welding applications, where the environment is harsh and the probability of the user making a mistake is high, a human-robot interaction display could prove beneficial.
Project CLARiSSA aims to be able to automatically weld with the help of the user. HRI facilitates this interaction, making user activity less skill intensive and more error tolerant. A display of the welds position on top of the beams, can improve the users performance, reducing the need for context-switch and the need to memorize or mark specific positions. It also vastly reduces the need for highly trained operators.
To particularize, an HRI in a welding application would be a proximate interaction, where the human's role would adapt according to the current necessity. In the ideal case, the user would be a supervisor, or a bystander (if the robot would be fully automated), instructing the robot, but would also work as a peer or operator when needed (scenarios of uncertainty or places the robot can't reach). The robot would always be dependent of the operator's orders asking when in doubt, (5 on the Sheridan, Verplank table 2.1) either through a visual display (simulator) or through augmented reality/projection mapping.
The need to withdraw the human operator from welding operations is because they are jobs in great supply but that demand a vast expertise. For example, in Figure 3.5 it's clear that a weld has to be perfectly executed and at the correct range. Moreover, replacing welds is extremely tricky and inconvenient.



(a) Arc Length Faults Up View

(b) Arc Length Faults Side View

Figure 3.5: Arc length faults on welding processes.

Some common welding errors are presented, and some causes and solutions are discussed in table 3.1 as well as a brief exposure of how HRI can solve them is shown.

Table 3.1: Possible welding defects

Defects	Probable cause	Traditional solution	How Human-Robot interaction can help solving the defect
	a) Short arc	Hold longer arc. Use proper electrode.	The arc will always be projected on the surface to weld.
1.	b) Insufficient puddling time	Allow sufficient time for gases to escape.	Not applicable.
Porous welds	c) Impaired base metal	Remove impurities in base metal.	Not applicable.
	d) Incorrect current	Use proper current.	Not applicable.
	e) Improper welding technique	Use weaving motion to eliminate pin holes.	Not applicable.
	a) Speed too fast	Weld slowly enough to get good root penetration.	Not applicable.
2.	b) Electrode too large	Select electrode according to welding groove size.	Selecting the electrode can be done with the projection in front of the operator.
Incomplete penetration	c) Current too low	Use sufficient current.	Not applicable.
	d) Faulty preparation	Calculate electrode penetration properly. Leave proper free space at bottom of weld.	Electrode penetration can be done with the projection in front of the operator, if operator is welding.
	a) Shrinkage of weld metal	Use intermittent welds. Control cooling.	Not applicable.
3.	b) Faulty clamping of parts	Clamp parts properly.	Not applicable.
Warping	c) Faulty preparation	Peen joint edges before welding. Space parts properly.	Not applicable.
	d) Overheating at joints	Increase travel speed. Use high speed, moderate penetration electrodes.	Not applicable.
	a) Incorrect speed	Use correct speed.	Not applicable.
4.	b) Current improperly adjusted	Use proper current to allow deposition penetration.	Not applicable.
Poor fusion	c) Faulty preparation	Use proper cleaning, edge preparation, and positioning.	Faulty preparation can be solved by eliminating the need of context switch.
	d) Improper electrode size	Select proper electrode.	Selecting the electrode can be done with the projective in front of the operator.
	e) Improper welding technique	Weave must be sufficient to meld sides of joint. Prevent weld metal from curling away from plates.	Not applicable.

It is proven therefore, that there is a need for highly intuitive welding interfaces to ease the operator's workload and that at the same time allows inter-cooperation between operators and machines (exchanging information both ways) exists.

3.3 Validation Setup

As CLARiSSA's robotic arm was unavailable during the duration of the dissertation, a validation setup was necessary to recreate the circumstances and industrial environment. The setup drawing is represented in Figure 3.6 and consists of the ABB_IRB140T robot manipulator, a table where the beam is placed and where the projection will occur, and a cardboard floor to fixate the origin of the scene. The origin of the scene is on the top left corner of the floor.

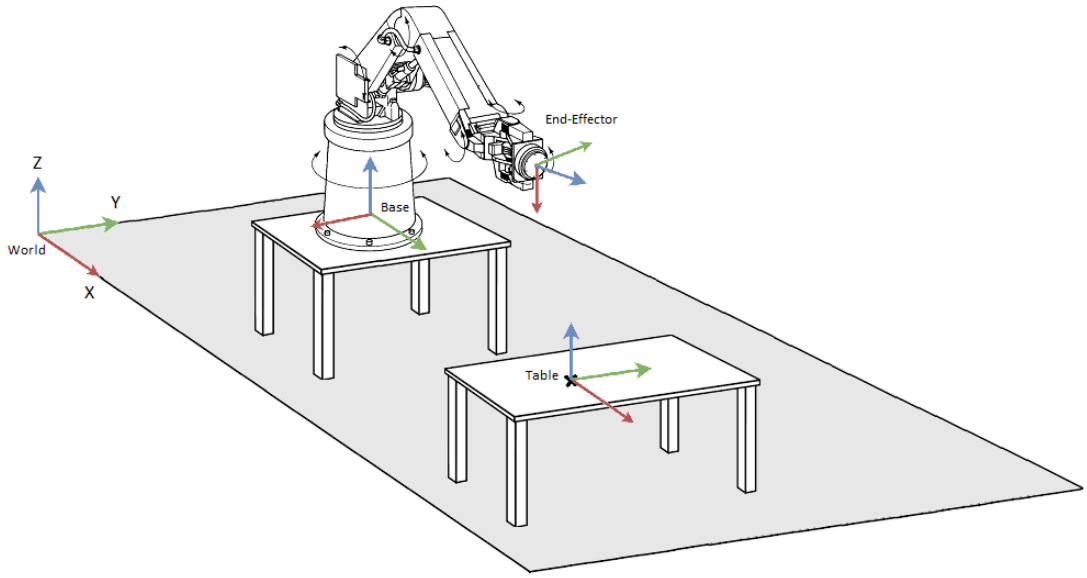
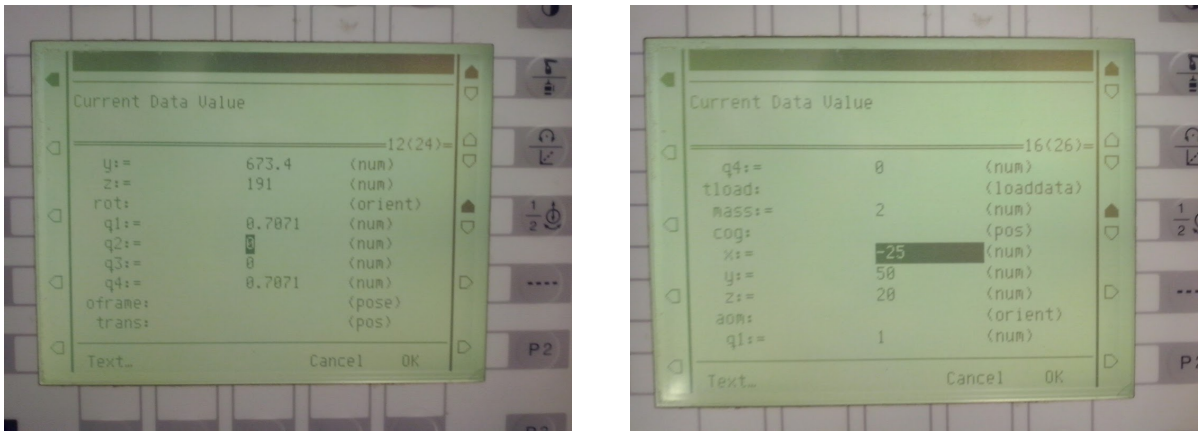


Figure 3.6: Validation setup drawing.

3.3.1 Setup Calibration

The first step when handling the ABB robotic arm is to calibrate the work referential and the tool. This can be done using the ABB's console. The point chosen to be the table referential (in cm referent to the origin) was x = 45, y = 673.4, and z = 191. Next, to avoid trajectory collisions with the projector handle, the center of gravity and weight of the tool also has to be defined (Figure 3.7).



(a) Work object referential definition.

(b) Weight of the tool and center of gravity.

Figure 3.7: ABB console definitions.

When the tool and work referential are set, a workspace needs to be programmed. As the validation setup required the projection to be displayed in 2 faces of the beam (see Figure 3.8), a workspace that could sweep both the upper part of the beam and the right part was programmed.

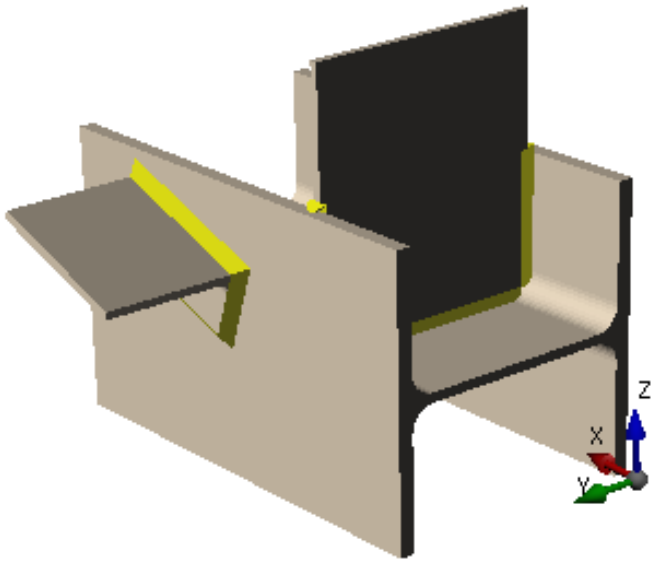


Figure 3.8: Beam Model in 3D software Eyeshot.

The serial port had the following configurations on both ends (ABB and computer):

```
> Baud Rate = 9600;
> Data Bits = 8
> Stop Bits = one;
> Parity    = None;
> Handshake = None;
```

The serial client and server's operation methods are detailed below in pseudo-code. In order to allow parallel execution between the robot's move order and the serial operations, a trap routine was configured to run with each 0.25 seconds of operation. "When an interrupt occurs, control is immediately transferred to the associated trap routine (if any)" [37].

ABB:
> Read Tool Position (x,y,z);
> Read Tool Quaternions (qw,qx,qy,qz);
> Read Joint Position;
> Structure the serial package;
> Send serial package;
> Loop;

Computer:
> Read data package from serial;
> Divide string first level;
> Divide strings second level;
> Update simulator variables;
> Loop;

It is to note that in the simulator, the packet must be handled carefully, and if it is empty or incomplete the simulator must not parse that packet (which introduces communications delays). After this, the camera and the projector have to be calibrated. Camera calibration is a necessary procedure in 3D computer vision in order to extract metric information from 2D images. It has been studied extensively in computer vision and photometry. Camera calibration describes the process of finding the camera parameters which facilitates the mapping between image and world coordinates. It captures the properties of a camera's lens such that it becomes possible to work with your images like they had been captured by a perfect pinhole camera.

The intrinsic camera parameters are related to the camera internal characteristics while the extrinsic relate transformations to determine other relevant transformations (in the case of the setup shown, transformations from the end-effector to the camera and then from the camera to the projector). Ideally, the central pixel would be half of the resolution of the image (in the x and y component) and the focal distances would be the same. However since the camera center is not parallel to the image plane the parameters are determined by:

$$f_x = f_y = \frac{C_x}{tan(f_{ov}/2)} \tag{3.1}$$

where f_x and f_y are the focal lengths in pixels, C_x and C_y are the optical center coordinates and f_{ov} is the field of view.

Then using a calibration toolbox [38] we calculate the extrinsic and intrinsic camera parameters and the associated error. The calibration procedure starts by obtaining several images from a chess pattern, with know dimensions in various perspectives (Figure 3.9).

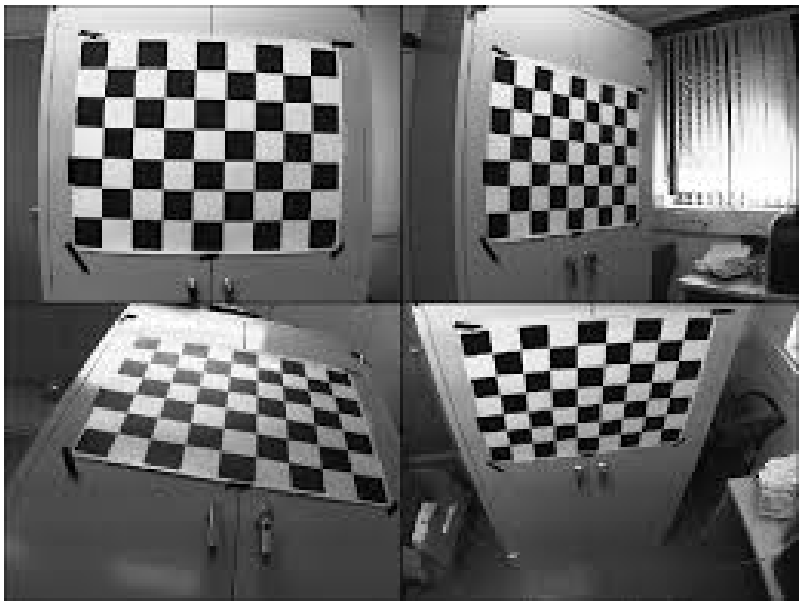


Figure 3.9: Chess pattern for camera calibration.

Identifying the chessboard corners and calculating the error associated (error between the position of the point identified and its projection) indicates a good estimative of the parameters found. The error was less than 1 pixel which further indicates that the estimative was good.

The following results were obtained:

Table 3.2: Intrinisic parameters for camera calibration

	Values	**Error**
Focal Length (fc)	[1525.72360, 1523.58147] px	[4.65717, 4.27532] px
Central Point (Cx, Cy)	[664.94735, 361.32823] px	[4.60629, 3.96935] px
Distortion (kc)	[-0.20860, 0.23070, 0.00220, 0.00027, 0.00000]	[0.01334, 0.10986, 0.00052, 0.00058,, 0.00000]
Skew	[0.00000] - 90.0º	[0.00000] - 0.0º

In the same way as the camera, projectors also need to be calibrated. The projector parameters were obtained resorting to the calibration software [39, 40]. This system uses the projector to project structured light patterns and the camera to capture this images. This process is repeated for diverse pattern perspectives. The methodology implemented uses local homographies to obtain sub-pixel precision. The following results were achieved for intrinsic parameters:

Table 3.3: Intrinsic parameters errors

	Values	Error
Focal Length (fc)	[407.8939, 861.1495] px	[0.12342, 0.12342] px
Central Point (Cx, Cy)	[642.7833, 418.3529] px	[0.12342, 0.12342] px
Distortion (kc)	[-0.00134, -0.10030, -0.00071, -0.00341, 0.00000]	[0.01334, 0.10986, 0.00052,0.00058, 0.00000]

For the extrinsic parameters:

$$R = \begin{bmatrix} 0.97663 & -0.01396 & -0.21445 \\ 0.01416 & 0.99989 & -0.00062 \\ 0.21443 & -0.00243 & 0.97673 \end{bmatrix} \tag{3.2}$$

$$T = \begin{bmatrix} 0.29298 & -0.01450 & -0.06229 \end{bmatrix}^T \pm 0.00019 \tag{3.3}$$

$$H_{Cameraprojector} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tag{3.4}$$

To calculate the homography between the support and the camera, the ABB IRB_140T was used. A camera and projector were fixed on a support like shown in Figure 3.10. Using the robotic arm it is possible to determine with high precision the position of the end-effector relating the robot base. The following strategy was used:

- Calculate the homography between the pattern and the robot base;

  Calibrate the tool;

  With the tool identify three corners of the calibration pattern (superior left, right and inferior left) and save the translations values and quaternions for each position;

  Calculate the homography;

- Calculate the homography between the camera and the calibration pattern and store the translation values and the quaternions in each position;

- Calculate the remaining homography.

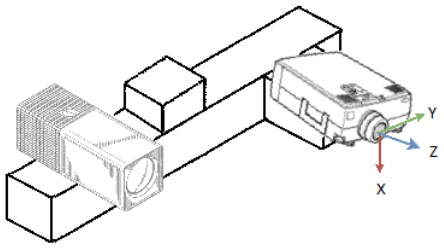


Figure 3.10: Projector setup drawing.

To detail the transformation between the calibration pattern and the robot base, a tool must be used (pointed tip). This calibration allows the robotic manipulator to determine the dimensions of the tool and returns the value of the position and orientation. To do this calibration procedure, the tip of the tool is placed in the same position in different angles preferably in distinct quadrants to obtain higher robustness.

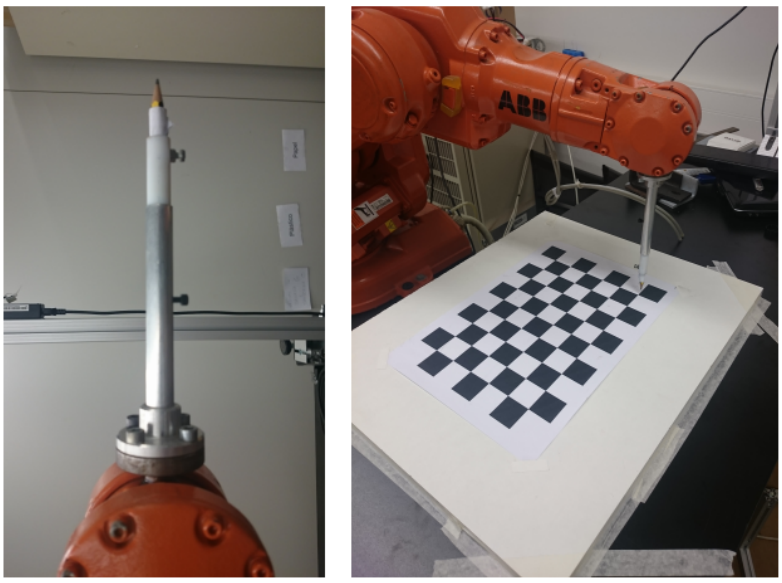


Figure 3.11: Tool calibration procedure.

Through direct kinematics the 3D localization of each of the points is given in relation to the robot base frame. In Table 3.4 these points are shown:

The points acquired represent the referential frame of the calibration pattern (origin, X axis, Y axis) in the robot frame. The following homography is obtained:

$$H_{ChessPattern} = \begin{bmatrix} 0.0406 & -0.9992 & -0.0054 & 0.7424 \\ 0.9992 & 0.0406 & 0.0013 & 0.1527 \\ -0.0011 & -0.0054 & 1.0000 & -0.1956 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

The homography between the camera and the calibration pattern is equal to the calibration of the extrinsic parameters, which means, its position relating to the world.

$$H_{ChessPattern}^{Camera} = \begin{bmatrix} -0.033678 & 0.999400 & 0.008081 & -0.16503801 \\ 0.998053 & 0.033205 & 0.052798 & -0.99002173 \\ 0.052498 & 0.009844 & -0.998572 & 0.52561520 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$H_{End\ Effector}^{RobotBase} = \begin{bmatrix} 0.0637 & -0.9980 & -0.0028 & -0.0229 \\ -0.9814 & -0.0632 & 0.1812 & 0.4613 \\ -0.1810 & -0.0088 & -0.9835 & 0.7860 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{3.7}$$

The missing transformation is obtained from:

$$H_{Camera}^{endEffector} = H_{RobotBase}^{ChessPattern} * H_{Camera}^{endEffector} * H_{ChessPattern}^{Camera} = I \tag{3.8}$$

Table 3.4: Three corner positions from the pattern.

	Origin	Down	Right
X (m)	-0.1829	-0.1758	0.1158
Y (m)	0.7345	0.5482	0.7474
Z (m)	0.1994	0.1984	0.1998
qw	0.00019	0.00023	0.00022
qx	0.70160	0.70156	0.70159
qy	-0.71255	-0.71259	-0.71257
qz	0.00464	0.00465	0.00468

resulting in:

$$H_{Camera}^{End\ Effector} = \begin{bmatrix} 0.0218 & 0.9996 & 0.0176 & -0.0834 \\ -0.9776 & 0.0177 & 0.2095 & -0.1357 \\ 0.2091 & -0.0217 & 0.9776 & 0.0409 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \tag{3.9}$$

After the calibrations routines are done, the end effector's position is given in accordance to the table referential. To have that position in the scene coordinates, a transformation that relates the table coordinate frame to the scene frame is needed. That transformation is fixed if the table doesn't move spatially. To combat this problem, tape markers were placed on both the table legs and on the table referential origin. After this transformation is obtained, multiplying it by the transformation that relates the scene origin to the robot base and the robot base to the end effector coordinates gives the end effector coordinates in the scene coordinate frame. Lastly, a transformation that relates the end effector coordinate frame to the projector is necessary. The equations that dictate the above operations are presented below:

$$H_{tableToProjector} = H_{tableToWorld} * H_{WorldToBase} * H_{BaseToEffector} * H_{EffectorToProjector} \tag{3.10}$$

$$P_{projector} = H_{tableToProjector} * P_{endEffectorInTableCoordinates} \tag{3.11}$$

The $H_{tableToWorld}$ is created by measuring the physical space with a measuring tape. As the table was aligned and marked with the origin, no rotation is needed. The translation that details the transformation is:

$$P_{tableToWorld} = H_{tableToWorld} * P_{table} \tag{3.12}$$

where $H_{tableToWorld} = \begin{bmatrix} 1 & 0 & 0 & -1000 \\ 0 & 1 & 0 & -570 \\ 0 & 0 & 1 & -920 \\ 0 & 0 & 0 & 1 \end{bmatrix}$,

In the same way, the matrices that describe the necessary transformations are shown below:

$$H_{WorldToBase} = \begin{bmatrix} 1 & 0 & 0 & 270 \\ 0 & 1 & 0 & 580 \\ 0 & 0 & 1 & 720 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_{BaseToEffector} = \begin{bmatrix} 1-2*(q2^2+q3^2) & 2*(q1*q2-q0*q3) & 2*(q0*q2+q1*q3) & \text{ABB x} \\ 2*(q1*q2+q0*q3) & 1-2*(q1^2+q3^2) & 2*(q2*q3-q0*q1) & \text{ABB y} \\ 2*(q1*q3-q0*q2) & 2*(q0*q1+q2*q3) & 1-2*(q1^2+q2^2) & \text{ABB z} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where q0, q1, q2, q3, ABB x, ABB y, and ABB z are values that come through the ABB serial port.

$$H_{EffectorToProjector} = \begin{bmatrix} 0.0464 & 0.9985 & 0.0276 & -89.1 \\ -0.9515 & 0.0368 & 0.2258 & -43.49 \\ 0.2245 & -0.0409 & 0.9519 & 3.83 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3.4 Discussion

In this chapter, a case study of project CLARiSSA is presented, and discussion on its functionalities and requirements are shown. To meet the demand, a validation setup was recreated since the CLARiSSA robot was unavailable. This setup was used to experiment the projection mapping. Camera and projector calibrations are also discussed for the validation setup, as well as general calibrations for the ABB IRB_140T, calibrating the work referential, the tool weight and center of gravity, and the serial communication between the robot and the computer. In the next chapter several concepts for a generic HRI are presented and discussed, as well as a generic architecture for Projection Mapping. These concepts will be examined and studied and an implementation based of them will be produced in Chapter 5.

Chapter 4

Generic Architecture for HRI Applications

The concept of a generic architecture for HRI interactions considers an environment (scene), which is composed of several devices such as a robot (can be a mobile robot, a robotic arm or other automated machinery such as a track beam), the surrounding objects (such as tables, obstacles, other machinery, etc.) and a projector to perform projection mapping. Exploring the concept presented, it's possible to divide the architecture in three main components:

- User Interface: Manages user interface events from an input device. That device can be either a voice recognition module, a laser pointer, a mouse, a keypad, a joystick, a gesture recognition module, a generic controller (Wii remote, Playstation controller), a marker or instrumented gloves selected and configured with the proper environment variables defined in the configuration system (ECS), according to each end-user.
- Software: Transforms user/interface events into actions/outputs, transmitting the information to different physical items. It is also responsible for updating the graphical interface, managing errors, supervising the communication module, and managing the multi-threaded synchronization of every software unit.
- Peripherals: The group of physical components aggregated to the system. It consists of a robot (mobile or robotic arm), and a projector to perform projection mapping. It deals with the specific characteristics of each peripheral allowing two-way communication between each component (through Wi-Fi, Ethernet, Serial RS232, CAN, infra red, radio protocol, and others).

An example of this can be explained from leap motion [41]. This concept is illustrated in figure 4.1.

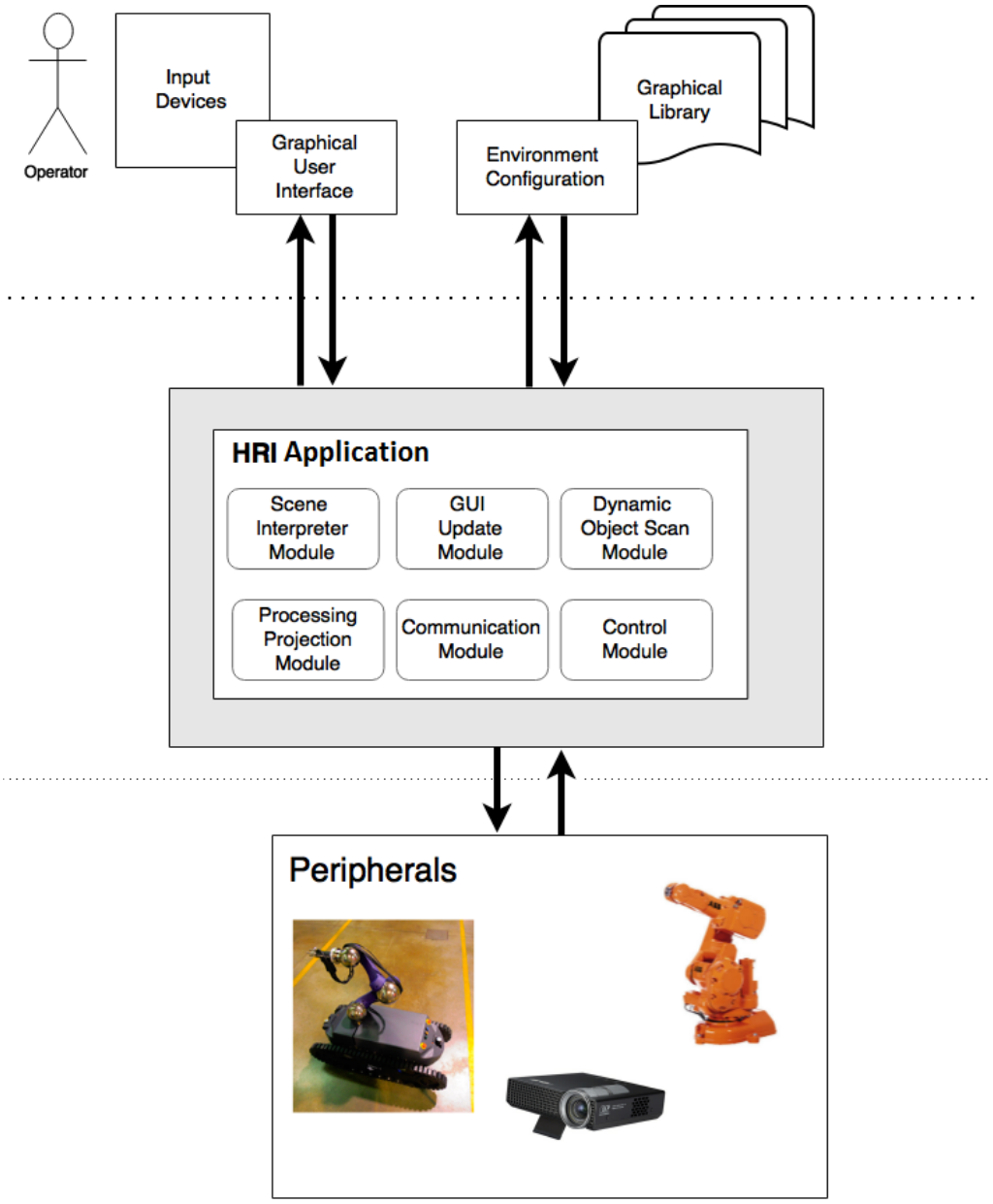


Figure 4.1: Software architecture for a generic HRI interface.

The HRI represents the main core of the architecture and plays the role of supervision, synchronization and coordination between other components. It includes the following modules:

- Scene interpreter;
- Dynamic object scan;
- GUI update;
- Control.
- Projection update;

- Communication.

In the following sections the previous modules will be discussed in greater detail.

4.1 Modules Description

4.1.1 Scene interpreter

One can argue that the understanding of reality contents and restraints is of great importance to apply correct projection mapping techniques. A general-purpose scene analysis system can be achieved with some success by limiting the task domain [42]. Scene analysis has also been approached by restricting the complexity of the scene (for example in outdoors) [43]. One way to combat the problems depicted in scene interpretation is to generate prior knowledge of the scene, which means, to have up to some extent the 3D model of the scene beforehand and to dynamically insert or remove objects from it.
To model the robot, several tools already exist [44]. To some extent, these tools can be used to model a scene using CADs. So, the scene interpreter can be modeled as an extension of a CAD interpreter. Using a tree approach to load CADs we have the potentiality of referencing objects/meshes to one another. The tree decomposition groups objects into Block References, which consequently creates a relationship between the current block to its parent and allows the block to be displayed as it is defined in its content fields. Each instance of the block reference includes parameters that define the content encased in the block and a reference to another block. A collection of parent and referenced blocks constitute a block reference hierarchy (Figure 4.2).

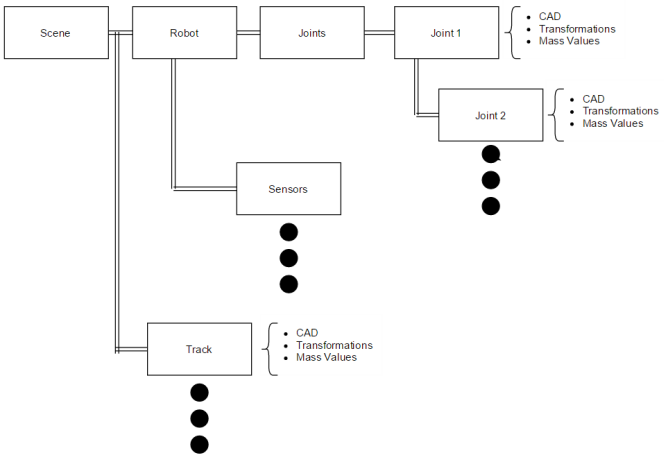


Figure 4.2: Example of a block reference hierarchy for a scene with a robotic manipulator and track.

A referenced block can contain as many blocks as possible and therefore reference lower-level blocks to any depth. The top block is the topmost level in the hierarchy of referenced blocks. Only one top block can exist, making it the parent block of the hierarchy. To prevent cyclic inheritance,

a block cannot refer directly or indirectly to a block that is superior to it in the block reference hierarchy (Figure 4.3).

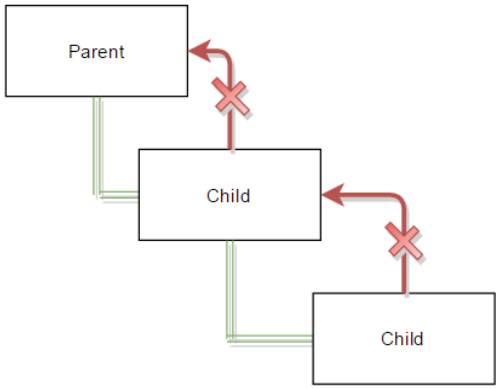


Figure 4.3: Block reference cyclic inheritance example.

A parent block can reference multiple child blocks. Different blocks can reference the same child block, by creating a copy of that block's data. If a block does not depend on data that is available only from a parent block, it can be used as a standalone block. This approach is advantageous because it allows block references to be used like subsystems, and organizes them hierarchically. Besides, it is possible to develop a block reference independently from the models that use it (modularity). It is also possible to reference a block multiple times without having to make redundant copies of it, and multiple blocks can reference the same block.

4.1.2 Dynamic object scan and GUI update Modules

The graphical user interface can be used in several applications, and therefore should be considered when designing a generic architecture for HRI. Visual information is precious to operators to understand the scene, and to be able to debug on a computer. Therefore, the graphical user interface should be updated as often as possible with new information. In the same way, the dynamic object scan module deals with entity operations (insert / remove / transform), transforming the current virtual scene entities to match the new information received. It should also modify the scene file to the current virtual environment settings, as well as couple necessary functions/-functionalities to each new entity. This operations should preferably be performed on the interface low-times (times where normal operation is not active).

4.1.3 Processing Projection Module

This module is responsible for performing projection mapping on the environment. Given the projector position and parameters (such as focal length, angle of view, width and height) and the

environment scene description, the module is able to output the correct image to project (Figure 4.4). In a generic architecture, it should also allow to tune the image according to several environment configurations (such as contrast, lighting, and others).

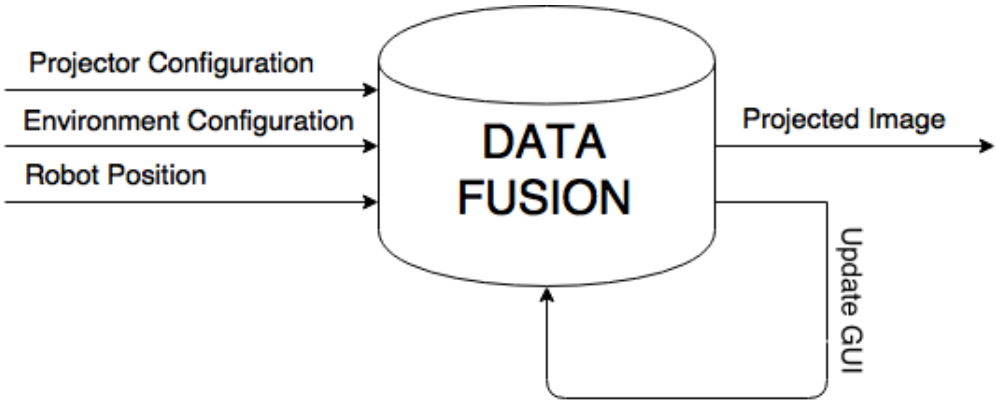


Figure 4.4: System representation of data fusion converting the configurations and the robot position into an updated projected image and a GUI update.

4.1.4 Communication Module

The communication module represents the low-level controller having the role of an intermediate between the HRI software and the physical peripherals. It ensures the possibility of intercommunication between the HRI supervisor and different environment peripherals. In fact, the diversity of common peripherals implies a wide range of networking protocols necessary to manage the whole network.

4.2 Key HRI features and functionalities

The HRI interface turns user and robot inputs into actions and commit a projected image as an output to the projector. The interface is a configurable software that connects the user to the environment. The HRI manages the actions of the user and the events which come from the system or physical peripherals. Some key features include:

- Modularity: The system must be as modular as possible, using as many or as little parts as needed;
- Extensibility: Adding/removing services or functionalities without modifying the operation of the system;
- Precision: Reduced projection errors on any scenario, indoors or outdoors, invariant to lighting as possible.

Another feature of the generic HRI architectures is the ability to project dialog boxes into a scenario. Common examples of dialogs (Figure 4.5)include:

- Error Dialog: This dialog is shown when the operation is not supported or an error has occured;
- Warning Dialog: This is shown to advertise the user that an operation did not go as expected. Usually followed by an acknowledge message;
- Acknowledge Dialog: This dialog is shown when a new command is performed or a new command was chosen after an warning message.

(a) Error Dialog.

(b) Acknowledge Dialog.

(c) Acknowledge Dialog.

Figure 4.5: Different types of dialogs.

4.3 Discussion

In this chapter, some principles for a generic HRI application were discussed. These should be applied when designing an interface for human robot interaction. Some considerations about the peripherals the user and a way to create abstract communications between them and the projection mapping system were studied and explained. A scene interpreter module is discussed, and the key features of the HRI and projection mapping are presented. The following chapter will build on the concepts presented and present the HRI architecture implemented with the help of Eyeshot and Gazebo in the validation setup presented in Chapter 3.

Chapter 5

Proposed Solution

5.1 Eyeshot implementation

Starting from the generic idea presented in the previous chapter, and focusing on the requirements presented by CLARiSSA project, such as scene adaptation, projection mapping on beams, and operator communication and visual feedback, we can outline and particularize the architecture presented in a way that it only contains relevant components. To implement the HRI and projection mapping system, Eyeshot was used. The software makes working with graphics easy and it is able to build complex scenes/environments easily. The CAD interpreter followed the same generic principles as the one presented in chapter 4. In order to perform projection mapping some concept introductions have to be explained, such as, the way Eyeshot models a camera, how it handles projections and how it deals with color spaces. It would be possible to manually perform projection mapping using only image transformations, but since Eyeshot already provides functions related to camera and scene modeling making it easier for the end-user. However, to compare the manual implementation to the Eyeshot implementation, it was necessary to implement both solutions. It is also important that for the proof-of-concept the validation setup mentioned in chapter 3 was used with the same calibrations procedures presented.

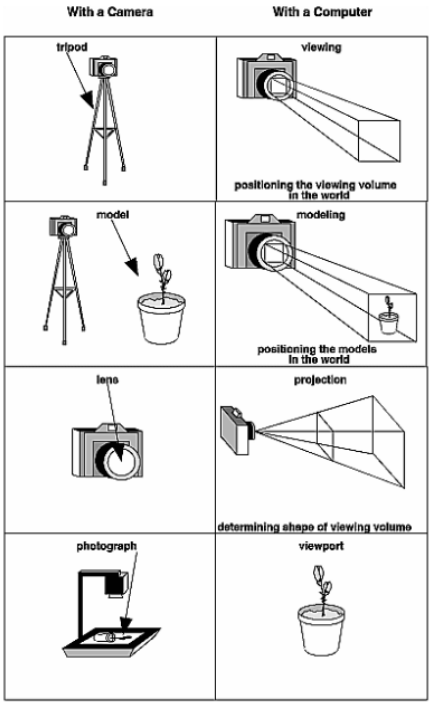
5.1.1 Eyeshot camera model

Transforming an image to produce an output for correct viewing is the same procedure one must make when taking a photograph with a camera. The steps are as follow:

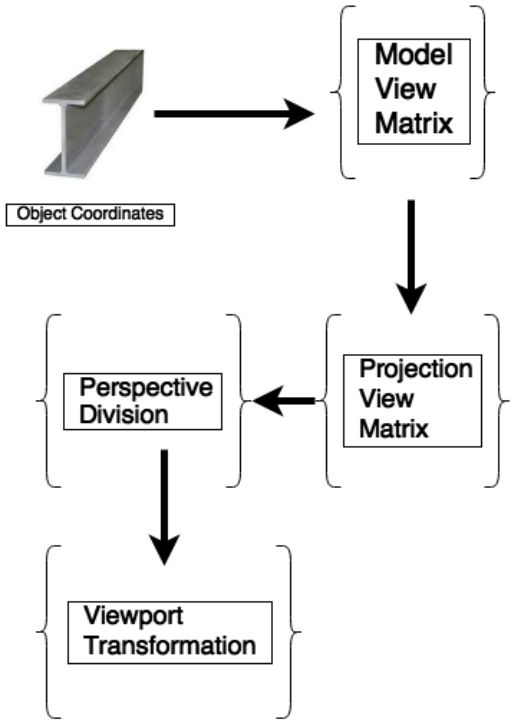
- Viewing Transform: Position the camera on the scene;
- Modeling Transform: Arrange the scene to be photographed at the correct position and rotation;
- Projection Transform: Choose the camera parameters, such as, the lens (focal length/angle of view) and adjust the zoom;

- Viewport Transformation: Perform size transformations to the image, enlarging or reducing the view so that it matches the expected output;
- Draw Scene: After all the transformations are performed, the scene can be drawn.

The figure 5.1 represents a simple camera analogy to explain how the viewport modeling works when compared to a real-life camera. The figure on the right exemplifies a vertex transformation from object coordinates to the viewport coordinates.



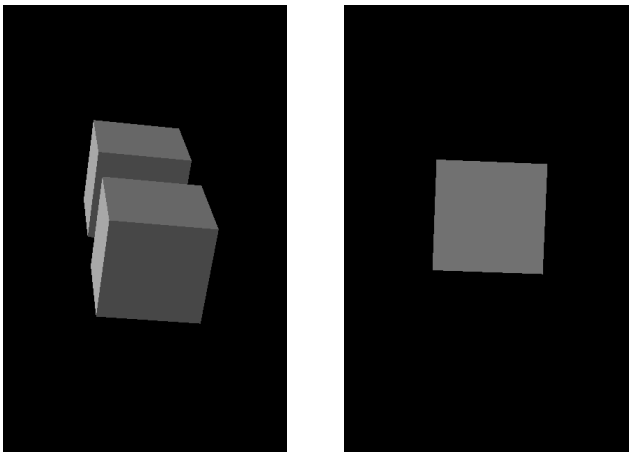
(a) Camera Analogy: Photograph vs Computer Model.



(b) Steps to perform the vertex transformation using a computer.

Figure 5.1: Basic concepts of camera modeling done with the use of computers [45].

To specify viewing, modeling, and projection transformations, 4x4 matrices are constructed and then multiplied in succession and with the coordinates of each vertex. The viewing and modeling transformations represent the model view matrix (eye coordinates). All transformations are applies to the z coordinates as well. This way, it can correctly reflect the depth of a coordinate. This is very important because the scene can delete certain invisible objects. (Example: Two objects with the same x and y coordinates but different z values). In figure 5.2 the z plane obstruction is explained. The image on the left has the camera position in the way that both cubes are visible. On the right, the view is placed facing frontwards towards the cubes, allowing the graphical library not to draw the back cube.



(a) Cube modeling without z plane obstruction

(b) Z plane obscuration.

Figure 5.2: Modeling example using Eyeshot [45].

The projection transform works exactly like choosing a lens for a camera.

While modeling the camera one must look at three different parameters (figure 5.3): The first, the focal length of a lens which is the distance (fixed) from the optical center of the lens to the sensor when the lens is focused on an object at infinity. "The angle of view is the angle of subject area that is projected onto the camera's sensor by the lens" [46]. The angle of view depends of the focal length, and the size of the camera's sensor. Lastly, the field of view is another way of representing the angle of view, but is expressed as a measurement of the subject area.

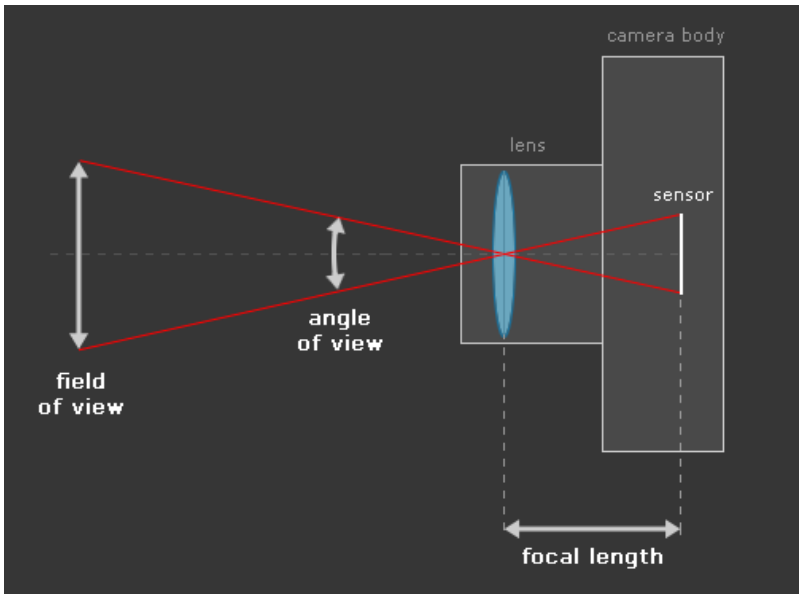


Figure 5.3: Top view of a camera [46].

In addition to the field-of-view considerations, the projection transformation determines how objects are projected onto the screen. Two types of projections are provided by Eyeshot:

- Perspective projection: Matches real-life view. Makes objects farther away appear smaller;
- Orthographic projection: Maps objects directly onto the screen without affecting their relative size.

In this work only perspective projections were used. The viewing volume for a perspective projection is a *frustum* of a pyramid (a truncated pyramid whose top has been cut off by a plane parallel to its base). Objects that fall within the viewing volume are projected toward the *apex* of the pyramid, where the camera or viewpoint is (Figure 5.4).

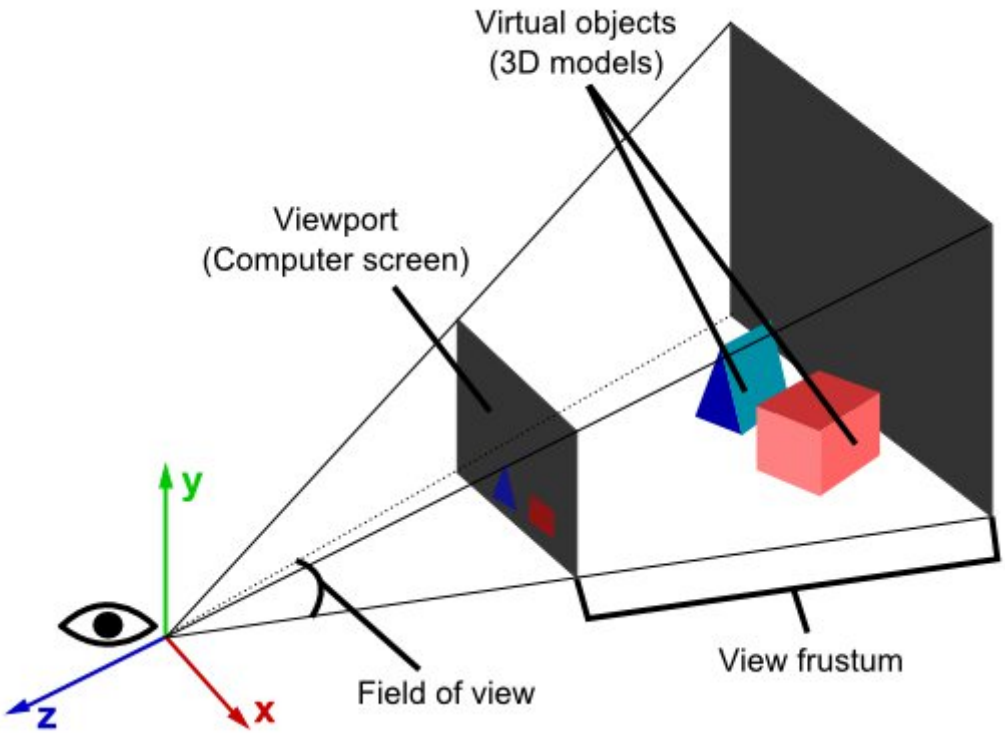


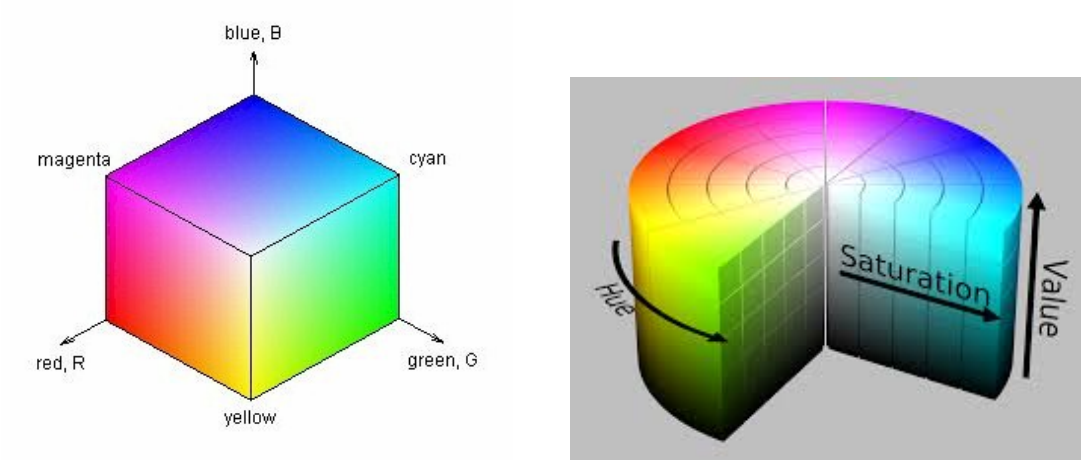
Figure 5.4: Perspective Transformation view.

Windows is responsible for placing windows on screen. By default the viewport is set to the entire pixel rectangle of the window that's opened. Eyeshot fixes this by intercepting windows events and rescaling the viewport accordingly.

5.1.2 Eyeshot color representation

In Eyeshot, visible colors are emulated by lighting pixels with a combination of red, green and blue light. To display a particular color, the monitor sends the right amounts of red, green, and blue light. The R, G, and B values can range from 0.0 (none) to 1.0 (full intensity). When R, G and B are 0.0 the color displayed is black (absence of color), and if they are 1.0 the color displayed is white (full color intensity). Another way to represent this colors is to use the HSV or HSL color

space. The idea is to rearrange the geometry of RGB in an attempt to be more intuitive by mapping the values into a cylinder (figure 5.5).



(a) RGB Cube (b) HSV Cylinder

Figure 5.5: Color spaces (RGB and HSV) [47] .

Eyeshot provides two color display modes. The first, the RGBA mode, uses a fourth variable (the alpha) to provide transparency and blending. The number of distinct colors that can be displayed at a single pixel depends on the number of bitplanes and the capacity of the hardware to interpret those bitplanes. "The number of distinct colors can't exceed 2^n, where n is the number of bitplanes. Thus, a machine with 24 bitplanes for RGB can display up to 16.77 million distinct colors" [47]. The second one, the color-index mode is basically a LUT (Look-Up table). "(...) the number of simultaneously available colors is limited by the size of the color map and the number of bitplanes available. The size of the color map is determined by the amount of hardware dedicated to it" [47].

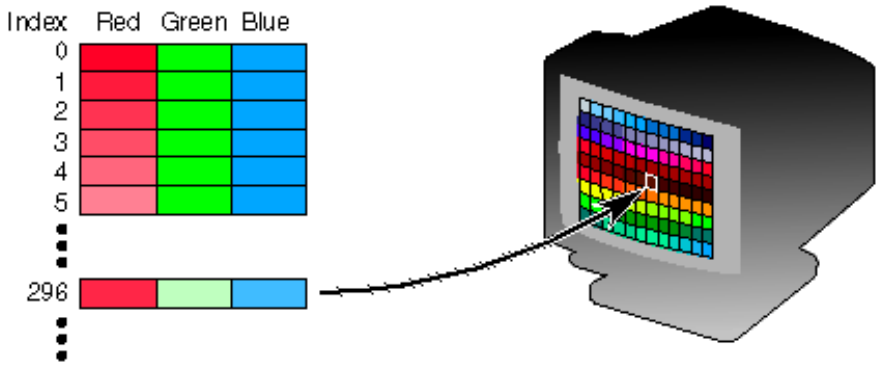


Figure 5.6: Color space representation.

Eyeshot uses both these display modes whenever it feels appropriate to (unless forced otherwise).

5.1.3 Manual implementation

Before explaining how the implementation was performed in Eyeshot, a brief mention on how the manual implementation is done is necessary. First, an image must be created with the size of the projector's lens (width and height). Next, a perspective view of the object has to be taken (Figure 5.7) and then, depending on the camera location, the horizon has to be determined (using the vanishing points technique). After the horizon is determined, it is possible to determine the remain angles. Then, in whatever scene is present, screen shot the weld in the appropriate camera position. The camera must be moved to the correct position first (to have in accordance the depth of the beam). When the screen shot of the weld is taken, an homography must be calculated to achieve the expected result, followed by an perspective transform to map the beam welds into the real-life scenario.

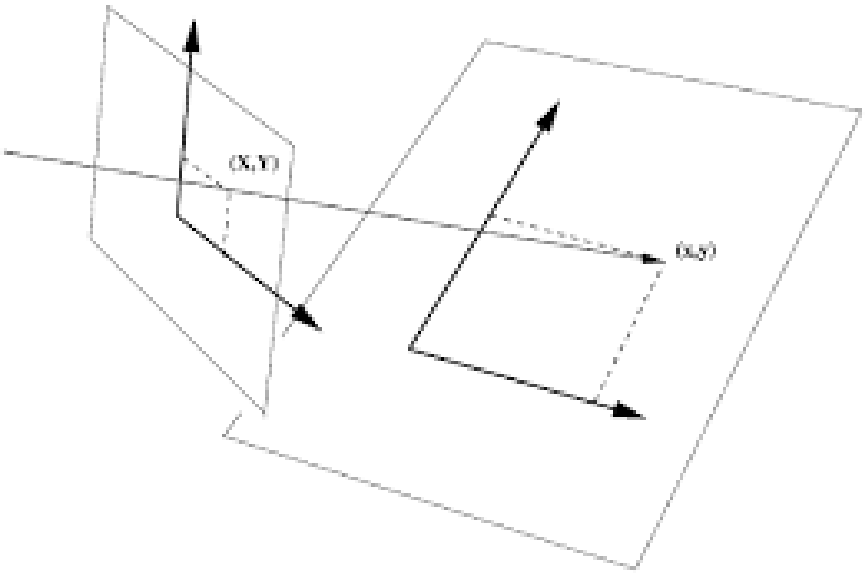


Figure 5.7: Perspective Projection referential.

5.1.4 Eyeshot implementation

To perform projection mapping with Eyeshot, and taking from the principles enunciated above, one must first create a viewport. The viewport doesn't need to have the projector width or height (since this will be forced later through EMGU_CV). This viewport will hold several properties, and if it is the reference for the projection (the viewport whose screen shot will be used as the image source for projection) a suitable background color must be chosen (or a way to automatically contrast the background must be provided). In the tests performed, in order to ensure contrast with the beam welds, the background of the viewport was painted black RGB(0,0,0). As mentioned in the previous subsections, the projection type has to be perspective in order to work with it to match real-life view. When the projection starts, the viewport image is resized and adjusted to the projector width and height, modifying the viewport camera's and then on a new window the

image source control is updated (each process has a different thread attached to it, so in order to update the source control a delegate must be created). To ensure modularity, every information pertained to the environment (ABB position, scene description file, Beam CAD, beam position and projected image) must be abstracted from the interface through a communication node. The nodes implemented were a serial node for the ABB end-effector position, a local scene file (.scene) containing the scene definition plus the projector position in the world, a local CAD standard file (.igs). Through TCP/IP sockets the beam position, rotation and scale is given, and the projected image is saved locally. This is for the cases when there's another application managing the projection (the same application in case the same computer is used for the projection). In those cases the images is saved for the other application to open it. When that doesn't happen the image control is instantly updated.

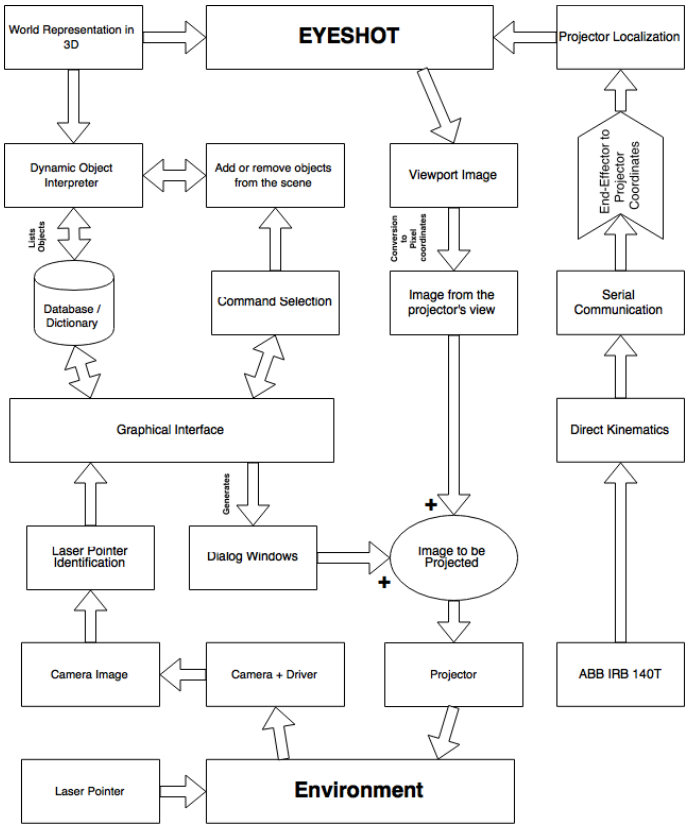


Figure 5.8: Detailed architecture for the projection mapping software in ABB IRB140.

In Figure 5.8 a schematic of the implemented architecture (both in Eyeshot and Gazebo is shown). The image shows some HRI elements passable of being implemented such as a laser pointer (which can be any other input device). The graphical interface is linked to a database or a dictionary to allow fast object recognition.

5.2 Discussion and Results

At the end of this work, an interface for projection mapping was obtained.

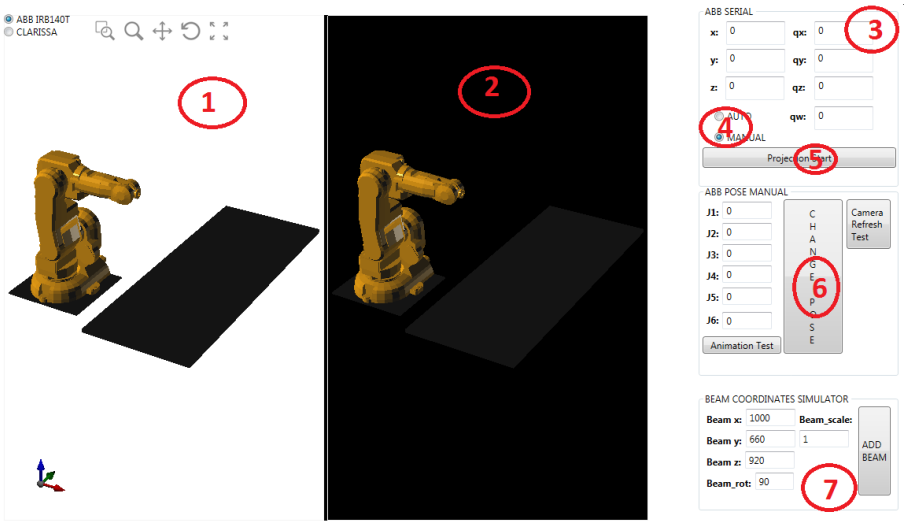


Figure 5.9: Eyeshot Interface explained.

The interface developed has several key points such as:

1. The main viewport, that can be maneuvered (zoom, translate, etc.) by the user;
2. The secondary viewport represents the projector image. When the projector or the ABB moves, this image is updated;
3. A serial communication is implemented to receive values from the end effector of the ABB;
4. An automatic mode is available as well as a manual mode. The manual mode is present for tests;
5. The projection start button initializes the timed events to refresh the source control. It also detaches the viewport into a new window and sourcing it with the projected image;
6. This button changes the pose of the robot. If the communication was bi-directional, one might order the robot to perform those movements;
7. In this menu, the beams coordinates are received and updated (the user can update them by hand).

The need for highly modular and intuitive interfaces is partly solved with this application because it can perform projection mapping everywhere as long as a scene exists and is modeled accordingly, and the projector position is known. The scene can change and the system will adapt accordingly because it can dynamically adjust itself by rendering after an object place event is detected.

However, some limitations are also present. The first one is that all the communications nodes are not abstract and have to be selected (at the expense of having to have them implemented).

Reading from the serial or TCP/IP or others and handling the packets is also time-consuming and creates unmeasurable delay to the projection. This can be avoided by using ROS (Robotic Operative System) to abstract communications and manage a more constant rate of data. This approach is particularly useful because when dealing with serial events the new data string can only be used if an entire packet is received (there might be times when the serial port read is scheduled over another instance of itself and the packet is scrambled and unusable). As ROS deals with this, it makes the system less prone to error. ROS also allows the system to be modular. In Eyeshot, to add extra viewport views, extra viewports have to be added. If object manipulation is required in the different windows, it becomes even more tricky as all actions performed in one of the viewports would have to be copied to the other viewport.

For example, as only one viewport camera can exist, extra viewports are necessary to display visual feedback.

One way to delete communication errors would be to remove from the architecture the need to create an image and source the viewport view to the image. This is be accomplished by creating two viewports and allowing one of them to detach itself from the control window. However, Windows does not allow this, unless they are different viewports all together.

Another limitation of this system is that its not cross-platform, and requires Eyeshot to work. This eliminates the possibility of transferring the code to an embedded platform. Windows O.S also induces lag in the current architecture because it is responsible for managing windows. This means that it will only update when it has time to do so, since refreshing the screen is not a priority task which means that, the Windows scheduler never refreshes the window, and the objects of the scene are never visible to the camera. This is fixed in later versions of Eyeshot, but a quick work-around was to render the viewport to bitmap every iteration (lag inducing).

Another limitation is when performing kinematics of the robot for visual feedback. When the robot is a simple case-study like the ABB_IRB140T, the Denavit-Hartenberg (DH) and the inverse kinematics (ikfast) are found online and can be encapsulated in a Windows .dll file and included in the Visual Studio solution (vastly reducing the ability to cross-platform the interface). For other robots, DH and OpenRave's ikfast tool would have to be used first to add kinematics to the robot and later encapsulated in a .dll file and added to the solution. This ensues even less modularity ability.

As for the precision of projection mapping, it's normal to have errors ranging from 2 degrees to 6 degrees (image 5.10). This is due to bad calibration of the workspace. As a 1:1 relation is considered between the modeled scene and the real-world (in millimeters) a slight error on either the projector position, the ABB end-effector data, or the table referential will originate small errors on the modelview matrix, and consequently on the projection. This is particularly important when the objects are small and the height of the projector is elevated (greater projection volume).

(a) Front View. (b) Back View.

Figure 5.10: Initial tests performed with the ABB.

A way to reduce these errors is to force calibrations routines on the end-operator. It is however impossible to fully eliminate these errors.
Another problem Eyeshot has is the lack of a module that can show what's the current work referential. This is a great addition and it's present in RVIZ (ROS module) and can help debug problems. A video of the system test can be found at [48].

It describes the interaction between the end-user and the robot to project the beam welds onto the beam currently present at the lab (small size). A manual mode for scene operations is present, to enable unit tests and for the calibration model (the manual mode saves the image, while the automatic mode skips the save step to achieve faster times).

However, the video filmed the system running the application with Gazebo simulator and not with Eyeshot. Although functionalities are similar, the current version of Eyeshot, at the time of writing, has a bug when dealing with modelview matrices, and the camera class seems buggy, or it does not update without an update command. It is also extremely slow because the scene needs to be rendered to bitmap each time, which is not needed with Gazebo.

To take numbers into account, the window was refreshed every 1.5 seconds for a new projection compared to 50 ms using Gazebo. This was because the scene had to render to bitmap in every iteration causing it to become slow (in Eyeshot version 8 this is no longer a problem). Other causes for the delay include the one incurred by the serial string buffer (which would only get an entire package every 200 ms).

With Gazebo, at half a meter from the beam, an error of 2 cm was measured for the projected image. For Eyeshot at the same distance, an error of 10 cm was measured (Figure 5.11). While the implementations were the same in both softwares, Eyeshot's camera functions did not interpret the final projection matrix for the projector's view well, causing it to have that slight error.

(a) Projection using Eyeshot.

(b) Projection using Gazebo.

Figure 5.11: Initial tests performed with the ABB.

To summarize, Table 5.1 shows the main limitations of each implementation.

Table 5.1: Limitations and comparison of Eyeshot and Gazebo

	EYESHOT	GAZEBO
Projection Error	10 cm @ 0.5 meters	2 cm @ 0.5 meters
Projection Refresh Rate	1.5 seconds	50 milliseconds
CPU USAGE	Can reach 100%	Irrelevant
Communications	Needs either a serial or TCP/IP layer.	Uses ROS
Interface	C# (Slow)	C++ (fast)
Native CAD Controls	YES	NO
Matrix and Quaternion Functions	Native, but buggy!	Yes, through libraries
Documentation	Scarce	Good (Community-driven)
Abstract and Generic	NO	YES

After analyzing the limitations and testing the implementation on the validation setup, an architectural change would be needed, either in the form of operative system/simulator change, or develop modules for Eyeshot to enable effective communication between modules. Lastly, a way to override window's window control would be needed to improve projection time as it is largely dependent on the viewport refresh and rendering. For SARKKIS and Project CLARiSSA three possible solutions arise. The first, is to use Gazebo on a virtual machine and use ROS to handle communication nodes allowing cooperation with the Eyeshot application if the use of Eyeshot is still intended. The second solution is to port the entire code base to Gazebo (losing all Native CAD controls which may not be optimal). The third and final solution is to wait for the update of Eyeshot and hope that the camera methods are corrected.

Chapter 6

Conclusions and future work

This dissertation presented an HRI interface that was built on generic architectural principles for an HRI application. The application was modeled using Eyeshot and used projection mapping and a GUI to provide visual feedback.

This application will be integrated in project CLARiSSA and will serve as a proof of concept for future beam welding applications using robotic manipulators. While some other solutions exist in the market, those require extensive operator training, or are placed in locations where they might not produce favorable results, or they are not adapted for human-robot cooperation scenarios.

While Eyeshot provides an easy interface for 3D modeling, it provided some implementation limitations that were not easy to deal with. Namely, the lack of documentation provided by DevDept, was a big hindrance in the development of the work. Another limitation is that this system is not cross-platform and would require a computer to be present at the scene. In mobile applications, the PC battery would need to be taken into account and would turn into another limitation of the system. Another limitating factor is the fact that both the camera and the projector require cables to operate which might get in the way of the robot's movement. Lastly, as calibrations are required, a fine tuning of the parameters would be needed, and even that would be useless if the projector or camera deviated from their estimated position (physical impact, others). In Eyeshot, the system presented a 4 6 degree error in the camera rotation when the position was 0.5 m from the target, which equates to a projection error of 10 cm (half the test beam length) while projecting each second and a half. In the Gazebo implementation, the error vastly reduced to 2 cm at 0.5 m while projecting at 20 Hz (50 ms).

For the future, one must analyze first whether the idea to maintain the generic HRI principles would be beneficial or not. While some support exists for Windows, Linux has an extensive community supporting Gazebo, and developing software for it is far easier. It also has access to ROS to maintain as much modularity in the interface as possible. However, a step has been made with the use of a 3D simulator to produce accurate projections, because it would be extremely difficult otherwise. If a 1:1 relation is achieved, it is possible to have an accurate projection, and hence this system would work.

The system is still dependent on information of the robot's localization and the world information. To improve this, modules for self-localization, and dynamic world scanning would need to be integrated. This would allow the system to track itself on the scene, to create a 3D rendering of what the world would be and then to project welds anywhere on the scene. Going forward, studying how to improve the information exchange between the several components would also be beneficial. This means, for example, allowing operators to communicate changes effectively, whether through gestures, markers, or other mechanisms. Safety mechanisms should also be researched and implemented.

The thesis research contributed to the understanding of the needs that HRI can fulfill in industry applications, as well as identifying relative importance of visual feedback in these same applications, how they interact with the common operator, and how they can improve their experience in robot cooperation. Lastly, it also contributed to validate robotic manipulators' effectiveness in beam weld applications (and others), through the use of projection mapping.

Bibliography

[1] Open Library, “Isaac Asimov work.” [Online]. Available: https://openlibrary.org/authors/OL34221A/Isaac_Asimov

[2] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, “Dual arm manipulation - A survey,” *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1340–1353, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2012.07.005

[3] M. a. Goodrich and A. C. Schultz, “Human-Robot Interaction: A Survey,” *Foundations and Trends® in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[4] Industrial Environments, “Industrial Environments.” [Online]. Available: http://www.industrialsafetysolution.com/labeling/industrial_environments.php

[5] H. Woern and T. Laengle, “Cooperation Between Human Beings and Robot Systems,” 1994.

[6] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 2006, vol. 141, no. 1. [Online]. Available: http://elib.tu-darmstadt.de/tocs/134922867.pdf

[7] H. Seraji, L. Cresenta, A. Data, R. Cited, and P. E.-a. R. Macdonald, “Dual Arm manipulators with adaptative control,” 1991.

[8] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, 1994, vol. 29.

[9] S. Kucuk and Z. Bingul, *Robot kinematics: forward and inverse kinematics*, 2006, no. December.

[10] P. Corke, “Visual control of robot manipulators - a review,” *Visual Servoing*, vol. 7, pp. 1–31, 1994.

[11] J. B. Kuipers, “Quaternions: "A Classic of Science",” p. 139, Mar. 1933. [Online]. Available: http://www.jstor.org/stable/3909157?origin=crossref

[12] ROS, “ROS.” [Online]. Available: http://www.ros.org/

[13] S. Hedge, “Perspective Projections,” pp. 1–13, 2015.

[14] B. Willinams, "An introduction to robotics," 2011. [Online]. Available: http://books.google.com/books?hl=en&lr=&id=RVlnL_X6FrwC&oi=fnd&pg=PR17&dq=An+Introduction+to+Robotics&ots=WHfQ7sQpVG&sig=WDfGunl2rmwXXSi7-eQUBJNf7Iw$\delimiter"026E30F$nhttp://books.google.com/books?hl=en&lr=&id=RVlnL_X6FrwC&oi=fnd&pg=PR17&dq=An+introduction+to+robotics&ots

[15] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation - A survey," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[16] H. Costa and T. Cunha, "Red Green Blue Depth Technologies," 2015.

[17] R. Diankov, "Automated Construction of Robotic Manipulation Programs," *Architecture*, vol. Ph.D., no. August, pp. 1–263, 2010.

[18] E. W. Weisstein, "Vanishing Point." [Online]. Available: http://mathworld.wolfram.com/VanishingPoint.html

[19] IEE, "RO-MAN '14." [Online]. Available: http://rehabilitationrobotics.net/ro-man14/

[20] AAAI15, "AAAI15." [Online]. Available: http://www.aaai.org/Conferences/AAAI/aaai15.php

[21] A. Steinfeld, T. Fong, M. Field, M. Lewis, J. Scholtz, and A. Schultz, "Common Metrics for Human-Robot Interaction."

[22] C. a. Miller and R. Parasuraman, "Beyond Levels of Automation: An Architecture for More Flexible Human-Automation Collaboration," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 47, no. 1, pp. 182–186, Oct. 2003. [Online]. Available: http://pro.sagepub.com/lookup/doi/10.1177/154193120304700138

[23] a. Y. C. Nee, S. K. Ong, G. Chryssolouris, and D. Mourtzis, "Augmented reality applications in design and manufacturing," *CIRP Annals - Manufacturing Technology*, vol. 61, no. 2, pp. 657–679, 2012.

[24] Prodevco, "Prodevco : ProFitter 3D projection fitting system." [Online]. Available: http://www.prodevcoind.com/profitter/

[25] H. I. Lin and Y. H. Lin, "A novel teaching system for industrial robots," *Sensors (Switzerland)*, vol. 14, pp. 6012–6031, 2014.

[26] M. H. Yun, D. Cannon, A. Freivalds, and G. Thomas, "An instrumented glove for grasp specification in virtual-reality-based point-and-direct telerobotics." *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 27, no. 5, pp. 835–846, 1997.

[27] P. M. Central, "What is projection mapping? - Projection Mapping Central." [Online]. Available: http://projection-mapping.org/whatis/

[28] Sarkkis, "SARKKIS." [Online]. Available: http://www.sarkkis.com/mechatronics/sarkkis_automatica_trade_fair.html

[29] SMERobotics, "SMErobotics." [Online]. Available: http://www.smerobotics.org/

[30] American Welding Society (AWS), "Vision for Welding Industry," p. 41, 2001.

[31] A. Staranowicz and G. L. Mariottini, "A survey and comparison of commercial and open-source robotic simulator software," *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '11*, p. 1, 2011.

[32] DevDept, "Eyeshot Image Gallery - devDept Software S.a.s." [Online]. Available: http://www.devdept.com/Products/Eyeshot/Gallery

[33] ROS Forum, "PR2 drifts over time in Gazebo - ROS Answers: Open Source Q&A Forum." [Online]. Available: http://answers.ros.org/question/42011/pr2-drifts-over-time-in-gazebo/

[34] P. Neto, J. N. Pires, and A. P. Moreira, "CAD-based off-line robot programming," *2010 IEEE Conference on Robotics, Automation and Mechatronics*, pp. 516–521, Jun. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5513141

[35] A. Miller and P. Allen, "GraspIt!" *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, Dec. 2004. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1371616

[36] ROS, "MoveIt!" [Online]. Available: http://moveit.ros.org/documentation/concepts/

[37] R. On-line, "RAPID Reference Manual System Data Types and Routines On-line ABB Flexible Automation."

[38] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab." [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

[39] D. Moreno and G. Taubin, "Simple, Accurate, and Robust Projector-Camera Calibration," *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pp. 464–471, Oct. 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6375029

[40] Brown University, "Projector-Camera Calibration / 3D Scanning Software." [Online]. Available: http://mesh.brown.edu/calibration/

[41] Leap Motion, "Leap Motion | Mac & PC Motion Controller for Games, Design, & More." [Online]. Available: https://www.leapmotion.com/

[42] B. a. Draper, R. T. Collins, J. Brolio, A. R. Hanson, and E. M. Riseman, "The schema system," *International Journal of Computer Vision*, vol. 2, no. 3, pp. 209–250, 1989.

[43] J. Roning, T. Taipale, and M. Pietikainen, "A 3-D scene interpreter for indoor navigation," *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, pp. 695–701, 1990.

[44] R. Diankov and J. Kuffner, "OpenRAVE : A Planning Architecture for Autonomous Robotics," *Robotics*, no. July, pp. —-34, 2008.

[45] OpenGl, "Chapter 3 - OpenGL Programming Guide." [Online]. Available: http://www.glprogramming.com/red/chapter03.html#name1

[46] Martybugs, "Understanding Your Camera: Focal Length, Field of View and Angle of View Defined : martybugs photography blog." [Online]. Available: http://martybugs.net/blog/blog.cgi/learning/Field-Of-View-And-More.html

[47] OpenGl, "Chapter 4 - OpenGL Programming Guide." [Online]. Available: http://www.glprogramming.com/red/chapter04.html

[48] T. Cunha, "tiagocunha | Results," 2015. [Online]. Available: http://ee10203.wix.com/tiagocunha#!results/c93v