Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey
School of Engineering and Sciences

TECNOLÓGICO
DE MONTERREY®

ROBOTIC-COMPUTER VISION SYSTEM FOR 3D WELDING TRAJECTORIES

A Thesis presented by:

Jesús Braian Rodríguez Suárez

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science

in

Engineering Science

Monterrey, NL.

June 2021

Dedication

The good example, honesty and perseverance are a legacy that I owe to my parents. Thanks Dad and Mom for your inspiration and support throughout my career, to you is dedicated this work with which I culminate an important stage of professional development.

Acknowledgements

I would like to thank the members of my thesis committee, my advisor, Dr. Alfonso Gomez, and my synods, Dr. Enrique Cuan, Dr. Arturo Escobedo and Dr. Rick Swenson, whose suggestions and comments guided and enriched the development of this thesis.

I am also grateful for the support of each and every one of my master's classmates, with whom I was able to count on at all times during the course of my studies. A sincere apology for not putting their names here, but I do not want an omission to erase the relevance of their friendship.

I thank to Consejo Nacional de Ciencia y Tecnología CONACyT for the financial support that allowed me to complete this master's degree and finish my thesis.

To Tecnológico de Monterrey for the full scholarship granted to attend the graduate program, as well as to the institution's personnel who gave me all the support I needed during my stay.

ROBOTIC-COMPUTER VISION SYSTEM FOR 3D WELDING TRAJECTORIES

By
Jesus Braian Rodríguez Suarez

Abstract

The necessity for intelligent welding robots that meet the demand in the real industrial production, according to the objectives of Industry 4.0, has been supported thanks to the rapid development of computer vision and the use of new technologies. In order to improve the efficiency in weld location for industrial robots, this work focuses on trajectory extraction based on color features identification over three-dimensional surfaces acquired with a depth-RGB sensor. The system is planned to be used with a low-cost Intel RealSense D435 sensor for the reconstruction of 3D models based on stereo vision and the built-in color sensor to quickly identify the objective trajectory, since the parts to be welded are previously marked with different colors, indicating the locations of the welding trajectories to be followed. This work focuses on the use of point cloud and a color data to obtain a three-dimensional model of the workpiece with which the points of the target trajectory are segmented by color thresholds in the RGB and the HSV color space, finally a spline cubic interpolation algorithm is implemented to obtain a smooth trajectory. Experimental results show that the RMSE error for V-type butt-joint path extraction is under 1.1 mm and below 0.6 mm for a straight butt joint, showing a suitable system for welding bead of various shapes and materials. It is important to note that to demonstrate its application in a robotic environment, the expected results will be presented in virtual environments created on the Robot Operating System (ROS) software.

List of figures

List of tables

Contents

Chapter 1 – Introduction.

1.1 Introduction.

In the era of globalization, manufacturing industries are facing competitive and uncertain markets where the innovation rate and shortened life cycles products create a challenge in industry to become more productive and flexible, so industrial robots equipped with intelligent programming tools represent the best alternatives to achieve this goal. The justification for the automation using robots results in the capabilities that these can offer: First, robot manipulators can execute industrial tasks in the same manner as humans and with comparable quality for longer periods of time. Second, they present the best rate between production cost and volume of production. Finally, they can be adapted to perform very different tasks mainly due to their programmability and flexibility [1].

In manufacturing industries, the welding process is a commonly found task. Nowadays, there are two main categories of robotic programming methods in industrial applications, which are online and offline programming. The latter, the most common in recent research due to its strength in the programming of complex systems and has proven to be more efficient and profitable for large volume production [2].

The time spent programming a new path for a job in high-volume manufacturing industries becomes the main challenges for using welding robots, especially when changes and uncertainties to the geometric shape in products occur. For this reason, the computer vision systems are currently used to capture the surfaces and/or characteristics of the environment and help achieve rapid off-line programming [2]. However, one of the obstacle in the development of an intelligent welding robot is have an the appropriate image acquisition system, which can solve the problem of trajectory planning, seam tracking and control of the welding systems, against errors caused by light and environmental disturbances [1].

Therefore, this project aims to prove that through a low-cost device that incorporates technologies such as stereo vision and a RGB sensor, it is possible to acquire images that allow measurements in 3D depth data and make a good reconstruction, to post process a correct strategy for robotic trajectory planning. In this field of research, recent works show advantages in seam tracking in real-time, combining two system of vision applied at different times in the process [3] or improving the accuracy using neural network control correction for path planning [4]. For this reason, an overview of the approaches in vision systems in robotic welding are presented below, as a comparison parameter to the challenges and scopes that this project encountered.

1.2 Problem Statement.

The need to achieve even greater productivity and less variability in product quality has forced the industry to adopt the use of vision systems in inspection and path guidance for

applications such as: spray painting [5], dispensing adhesives [6], and automated welding [7].

This industrial growth has pushed towards the fourth industrial revolution, known as industry 4.0, which has a strong adoption for automation, information exchange and manufacturing technologies. Within the industry 4.0, on the subject of robotic automation, one of the four pillars of research and implementation are robotic systems based on intelligence and robotic perception [8].

1.3 Overview of Alternative Robot Welding Vision Systems.

Before going deeper into the theoretical foundations and the technique proposed in this thesis, we proceed to show other approaches that are currently being developed using vision systems focused on robotic welding. The purpose of this section is to place the present research compared to other alternatives reported in the subject.

An important remark is to describe a weld join as a gap between workpieces to be joined, however the geometry of the joint can be presented in various ways, as shown in Fig 1: butt, lap, tee, edge and corner joint. Due to the problem of accurately locating each of these geometries, the proposals seek to satisfy only or at least one of these weld joints at a time.

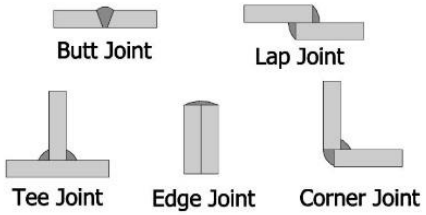


Fig 1 Types of welding Joints.

For example, the seam can be identified without prior knowledge of the geometry or location of the seam from a single image. Kiddee et al. [9] developed a technique to find a T-welding seem, based on the image processing of smoothing the image and extract the edges of the image by using Canny operator algorithm [10], in order to find the initial and ending points of the joint by link the broken edges. In the same way Chen et al. [11] propose a two-step method named coarse to fine where through a Canny detector, the two parallel edges capture in a butt v-joint, are used to fitting the value of the start welding position.

Dinham et al. [12] developed a system that works well even in the presence of imperfections on the surface of the steel. The method developed use Hough transform [13] to detect the outside boundary of the weldment so that the background can be removed.

In weld tracking systems, Ma et al. [14] used two normal charge coupled device (CCD) cameras to capture clear images from two directions; one of them is used to measure the root gap and the other is used to measure the geometric parameters of the weld pool.

In the same field of weld tracking the use of laser vision system can be found, Fernandes et al. [15] developed a low-cost system based on artificial vision in which a laser is set perpendicular to the work piece surface with the webcam on the robot arm oriented towards the projected laser stripe at an angle of 45°. The software tracks the deepest point in the welding seam using algorithms and positions the welding head above it.

Chang et al. [16] developed a robust algorithm for a portable welding robot. The method has its strong points in its ability to detect not only the seam, but also the three-dimensional configuration. The aim of the system is to track the seam for weaving weld path planning using a laser displacement sensor.

Emphasizing the problem of locating weld seams on metallic surfaces as shown in Fig 2, Li et al. [17] propose a robust automatic welding seam identification and tracking method by utilizing structured-light vision. Where the gray distribution of the laser stripe is tracked and the profile of the welding seam is searched in a small area by using the Kalman filter [18], with the aim to avoid some disturbances. Later, the same research Li et al. [19] propose to reduce the influence of noise on extraction of the center line, through double-threshold recursive least square method.

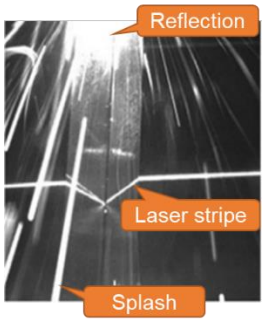


Fig 2 Image acquisition disturbances with laser weld tracking systems.

One example of an experiment already proved in the industrial environment was presented by Liu et al. [7], where an autonomous method to find the initial weld position for a fillet weld seam formed by two steel plates. This method employs an automatic dynamic programming-based laser light inflection point extraction algorithm. The algorithm for this method can support factors induced by natural light which may be present during the processing of laser vision images. An auxiliary algorithm to compensate for a time delay is also presented by the authors.

Recent systems tend to be more robust or complex in terms of the number of tools involved in obtaining images, and filtering data. For example, Zeng et al. [20] proposed a weld position recognition method based on directional light and structured light information fusion during multi-layer/multi-pass welding. On the other hand, Guo et al. [21] propose multifunctional monocular visual sensor based on the combined laser structured lights, which has the functions such as the detection of the welding groove cross-sectional parameters, application for the joint tracking, the detection of the welding torch height, the measuring of the

weld bead appearance, and the monitoring of welding process in real-time. Other approaches to real-time processing were described by Kos et al. [3] to compute the position of the laser beam and the seam in 3D during welding with a camera and illumination laser in order to equalize the brightness of the keyhole and the surrounding area.

Yang et al. [22] proposes a welding detection system based on the 3D reconstruction technology for an arc welding robot. The shape from shading SFS algorithm [23] is used to reconstruct the 3D shapes of the welding seam and the curvature information is extracted as the feature vector of the welds additionally support vector machine algorithm is applied to achieve the detection task of welding quality. To improve the efficiency of the teaching-playback, the authors propose a 3D path teaching method based on stereo structured light vision system using a seam extraction algorithm based on the kernelized correlation filter KCF algorithm [24], which could achieve fast and accurate seam extraction to modify the model of weld seam. Through the coarse extraction and fine positioning of weld seam, their system could well realize a fast and accurate 3D path teaching of a welding robot. Experiment results in Fig 3 show that the measurement resolution is less than 0.7 mm and the proposed method is suitable for V-type butt joint before welding [25].

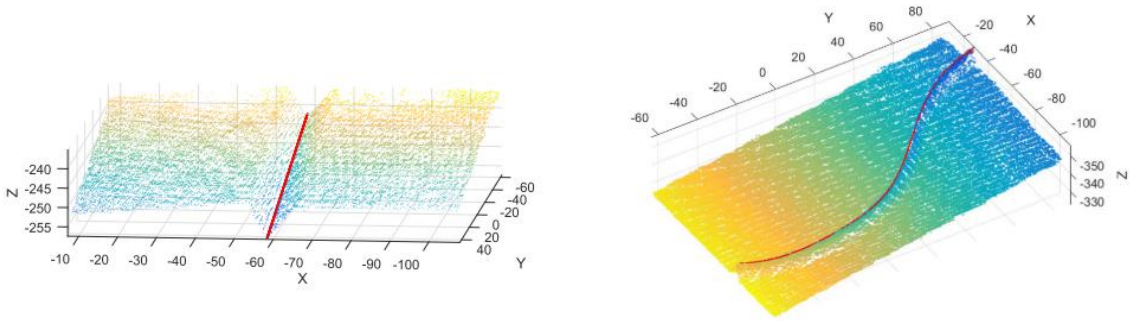


Fig 3 The fitting results of weld seam through 3D reconstruction.

In an effort to cover multiple types of weld seams, Xiao et al. [26] trained a model of a region convolutional neural network R-CNN to distinguish them automatically. In a second step a 3D point cloud data is used to reconstruct welding seams. Finally, through the point cloud and guided by the neural network can obtain the equations and initial points of the weld seam. Guidance test results prove that the extraction error is less than 0.6 mm, meeting actual production demand.

Zhang et al. [27] acquires the 3D information by multiple segment laser scanning. The weld features are extracted by cubic smoothing spline, to detect the characteristic parameters of weld lap joint with a deviation lower than 0.4 mm. Similar approach in structured light systems incorporated an optical filter and LED lighting developed to reduce the effect of noise produced by the arc torch. Where a Fuzzy-PID controller can be obtained the feature point, in horizontal and vertical directions simultaneously [28].

In point clouds acquired with RGB-D sensors, Maiolino et al. [6] use an ASUS Xtion sensor to register and integrate the point cloud with the CAD model to perform an offline programming

system for a sealant dispensing robot. On the other hand Zhou et al. [29] use a RealSense D415, from Intel, to detect and generate the trajectory with an algorithm based on the gradient of the edge intensity in the point cloud, reporting an RMSE error of 0.8 mm at the position from the optimized trajectory vs. the welding path planned by the teach–playback method.

1.4 Hypothesis and objectives.

Justification

By evaluating the literature for reported methods of trajectory planning and tracking for welding seams, we can define that the methodologies fall mostly on two aspects: 2D algorithms for image processing and systems based on weld-path tracking. Therefore, it leaves an opportunity for research on global planning systems based on 3D reconstruction.

Hypothesis

This work aims to prove that through a device that incorporates technologies such as Stereo Vision and a RGB sensor, it is possible to acquire images that allow measuring the features of a three-dimensional objects and achieve the 3D reconstruction, in order to obtain a correct robot trajectory planning.

General objective

Trajectory extraction of weld seams from a three-dimensional workpiece, through a stereo vision with a coupled RGB sensor, enabling the autonomous programming welding robots.

Specific Objectives

- Perform model reconstruction of three-dimensional objects through the point cloud acquisition, by the proposed active stereo vision system.
- Evaluate the error between the aligned point clouds of the CAD model versus the reconstructed point cloud model.
- Evaluate the error between the point cloud generated by the stereo system versus the aligned color point cloud.
- Integrate the data acquisition and processing system with the automated programming and planning robotic system to simulate the weld seam tracking.
- Compare the results from this work to other similar reported in the literature.

1.5 Thesis organization.

Chapter 2 describes the theory related to vision systems, to obtain 3D reconstruction. The Chapter 3 shows the proposed methodology for extract the weld trajectory, through the 3D vision system that incorporates the 3D reconstruction with color information. In Chapter 4 provides a methodology to simulate and implement the algorithm proposed in the vision system of Chapter 3 in the ROS simulation environment. Finally, Chapter 5 shows the results obtained by the vision system and its simulation.

Chapter 2 Theoretical Background.

2.1 Computer vision and depth detection techniques.

Computer vision seeks to understand a given scene using visual information. From a hardware standpoint, vision systems based on transducers that measure light intensity, commonly producing images or image sequences (video) and, in some cases, point clouds. The transducer or camera is an optical, non-contact device that takes advantage of the properties of light to capture environmental information. This light is transformed into a processable physical quantity and displayed a measuring map from the cameras point of view [30].

There are several image acquisition techniques that can provide the (X, Y, Z) coordinates. In order to provide a theoretical framework, this proposal mentions the two main methods for obtaining optical depth images known as depth paradigms: Depth from triangulation and Depth from Time of Flight ToF.

2.1.1 Time of Flight ToF.

This technique measures the time it takes for a signal, usually a photon of light, to travel a certain distance. If the signal is sent from the camera position, it has to travel twice the distance between the camera and the object that reflects the signal as shown in Fig 4. For this reason, the depth z can be calculated directly from half of the time spent travelling, multiplied by the speed of the signal. However, this technique presents problems such as low resolution and high depth distortion [31].

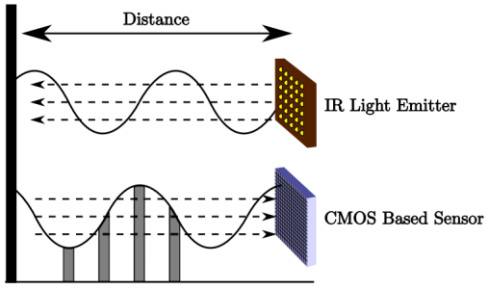


Fig 4 Distance measurement using Time of Flight sensor.

2.1.2 Depth from triangulation.

If an object is viewed from two points of view separated by a distance b, called the baseline, a different angle from the baseline is formed at both positions. This difference in viewing angle results in an offset of the image plane, known as a disparity, from which the depth of the object can be inferred. There are a significant number of techniques based on triangulation, however since this proposal uses a set of cameras, the technique of stereo vision will be explained below [32].

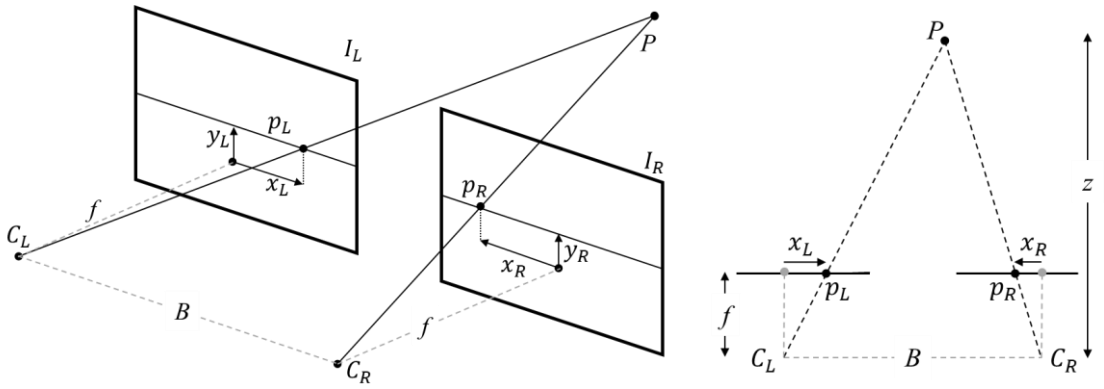
2.1.2.1 Stereo Vision.

Arrangements that consist of two image sensors (cameras) separated by a known distance are known as stereo systems. The principle of stereoscopy is based on the ability of the human brain to estimate the depth of objects present in the images captured by the two eyes.

In the stereoscopic configuration, two cameras are placed close to each other with parallel optical axes. The configuration is shown in Fig 5a. Both cameras, with centers C_L and C_R separated by a distance B called base line, have the same focal length f, so that the images I_L and I_R are in parallel planes. A point in the three-dimensional space P will be projected in different positions, p_L and p_R with coordinates (x_L, y_L, 1) and (x_R, y_R, 1) respectively, of the planes of the images because it's seen from slightly different angles. This difference in position is known as disparity and is mathematically described as disparity in (1) and is used to calculate the z-distance in (2) through the geometric relationship, shown in Fig 5b [33].

$$disparity = x_L - x_R \tag{1}$$

$$z = \frac{B \times f}{\text{disparity}} \tag{2}$$



(a) (b)

Fig 5 a) Stereo camera configuration, b) The geometry of a stereo system.

2.1.2.1 Structured light.

Structured light is an active method to improve depth acquisition by using an external light source that provides additional information to the system. Structured light is based on the use of active illumination of the scene with specially designed 2D spatially varying intensity pattern, where the camera sensor searches for artificially projected features that serve as additional information for triangulation [34].

For the work presented here, the RealSense sensor has an optical projector which uses pseudo-random binary array to produce grid indexing strategy of dots. The array is defined by an $n_1 \times n_2$ array encoded using a pseudo-random sequence, such that every k_1 by k_2 sub window over the entire array is unique [35]. An example of a pattern is presented in Fig 6.

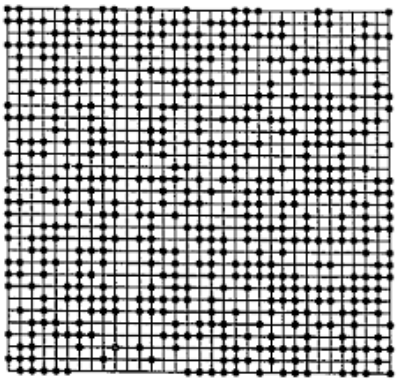


Fig 6 Example of a pseudo-random pattern.

2.2 Point Cloud.

Depth cameras deliver depth images, in other words, images whose intensity values represent the depth of the point (x, y) in the scene. A point cloud is a data structure used to represent points with three dimensions (X, Y, Z) where the depth is represented by the Z coordinate.

Once the depth images are available, it is possible to obtain the point cloud using the intrinsic values of the camera with which the information was acquired. This process is known as deprojection. A point P with coordinates (X, Y, Z) can be obtained according to (3), (4) and (5) from the depth information $D_{x,y}$, being (x, y) the rectified position of the pixel in the sensor. Where the variables c_x, c_y, f_x and f_y are the intrinsic values of the camera used to acquire the information, being (f_x, f_y) the components of the focal length and (c_x, c_y) the image projection center [36].

$$X = \frac{D_{x,y} * (c_x - x)}{f_x} \tag{3}$$

$$Y = \frac{D_{x,y} * (c_y - y)}{f_y} \tag{4}$$

$$Z = D_{x,y} \tag{5}$$

2.3 Illumination.

Illumination is a factor that must be considered for the quality of image acquisition in any machine vision process. We must remember that cameras are devices that allow capturing light information from an area to be analyzed and interpreted. Good lighting of the scene will provide sufficient illumination to distinguish the region of interest, allowing to greatly simplify the processing to be performed on images obtained by cameras, remembering that the vision system translates the colors of the piece in brightness levels that are highly related to the light that is reflected by each pixel [37]. Fig 7 shows the physical configuration of the main lighting systems, and TABLE 1 describes the characteristics as well as the advantages and disadvantages, in addition to the common applications in the industry.

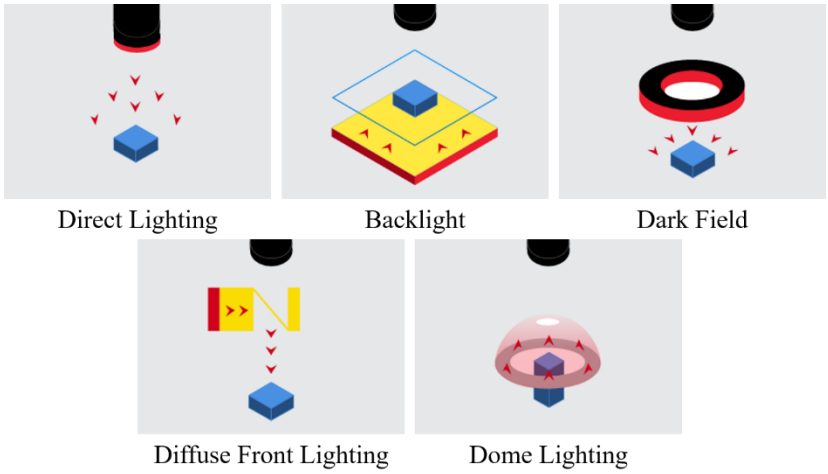


Fig 7 Physical configuration of the main lighting techniques.

TABLE 1 A summary of lighting techniques, advantages, disadvantages, and applications.

Illumination technique	Description	Advantages	Disadvantages	Applications
Direct front lighting	The light is incident directly on the object with a low angle of incidence from the illumination source.	Easily covers the workpiece and is very practical to adjust. Direct illumination provides maximum image contrast.	Three-dimensional pieces cause shadows. Specular reflection in shiny pieces.	Used mainly in the location and recognition of parts and surface inspection.
Backlight	The object is placed between the camera and the light source.	Provides maximum contrast between the part and the background. Simplifies the image by showing the silhouette of the part.	The surface of the part is lost. The object on fasteners is complicated to highlight the effect.	Widely used in the measurement of parts and dimensioning, also in the definition of boreholes.
Dark Field	Achieved either with bars or rings lights with a low angle of incidence (10 to 15 degrees) providing a band of light on the target or along the edge of the target.	Shows surface detail on very low contrast parts.	It is not recommended for light absorbing surfaces or in applications with high requirements.	Especially good for use in surface inspection on shiny, highly reflective targets.
Dome Lighting	Dome lighting, also called "cloudy day" lighting, provides uniform light from various angles.	Provides soft illumination from all directions. Eliminates glare and shadows over the entire surface.	Tends to decrease the contrast slightly. Require close proximity to the target	Metallic or shiny pieces. Three-dimensional pieces where shadows become a problem.
Diffuse Front	Also called coaxial illumination, transmits light perpendicular to the lens by means of a mirror to send the light at a 90-degree angle to the lens.	Highlights surfaces perpendicular to the camera, while darkening those at an angle. Decreases shadowing and has very little brightness.	Thickness of mirror can produce a double image	Useful for detecting flaws on shiny and flat surfaces, making measurements or inspections on shiny or transparent objects.

2.4 Data Registration and Integration.

Sometimes multiple views are needed to acquire data over the entire surface, and the registration is used to determine the transformation of data from two different views so that data can be integrated under the same coordinate system. That is why the 3D registration problem establishes two problems, calculating the displacement between two-point clouds and estimating the point correspondence between the point clouds. In the literature there are many algorithms among which the Iterative closest point algorithm stands out. Which seeks to obtain a global solution to the registration problem [38]. Some data registration and integration techniques are given in the following subsections.

2.4.1 Colored Point Cloud.

Some 3D sensors are often coupled to an RGB camera, with which research on color depth registration is being carried out. Registering two cameras means knowing the relative position and orientation of one scene with respect to another [39]. In principle, color integration consists of re-projecting each 3D point onto the RGB image to adopt its color. When reprojected in 3D, the generated point cloud contains six information fields, three for spatial coordinates and three with color values. However, due to occlusion, not all reconstructed 3D points in the scene are visible from the RGB camera, so some points may lack color information [38]. Fig 8 shows the result of the colorization of the point cloud shown on the left of the same figure.

(a) (b) (c)

Fig 8 Point cloud color map registration, a) Depth information, b) Color information and c) Colored Point Cloud through RGB registration.

2.4.2 Iterative Closest Point (ICP).

In general, the ICP algorithm iterates over two steps: Find correspondence set K={(p,q)} from target point cloud P, and source point cloud Q transformed with current transformation matrix T. And update the transformation T by minimizing an objective function E(T) defined over the correspondence set K. In the last 20 to 30 years a lot of variants of this algorithm have been developed in which a fine or coarse registration is searched [38], [40].

For the development of this thesis is shown the research done by Jaesik Park et al. [39], which developed an ICP algorithm in which the values of color intensity at each point are taken as an optimization parameter. Where the goal is to find the optimal T transformation that aligns Q with P, being these two clouds of colored points. Through the objective function of the following equation

$$E(T) = (1 - \sigma)E_C(T) + \sigma E_G(T) \tag{5}$$

Where T is the transformation matrix to be estimated. E_C and E_G are the photometric and geometric terms, respectively. $\sigma \in [0,1]$ is a weight that balances the two terms.
The geometric term E_G is a point to plane objective function.

$$E_G(T) = \sum_{(p,q)\in K} ((p - T_q) \cdot n_p)^2 \tag{6}$$

Where K is the correspondence set in the current iteration. n_{p} is the normal of point p.
The color term E_C measures the difference between the color of point q (denoted as $C(q)$) and the color of its projection on the tangent plane of p.

$$E_C(T) = \sum_{(p,q)\in K} (\mathrm{C}_P(\mathrm{f(Tq)}) - C(q))^2 \tag{7}$$

Where C_P is a precomputed function continuously defined on the tangent plane of p. And function f projects a 3D point to the tangent plane.

Chapter 3 Methodology for trajectory planning based on a stereo vision system embedding RGB data.

This chapter shows the development and implementation of the methodology for the processing of depth and color images, allowing the 3D reconstruction of the captured scene. The task to be developed with this 3D information, will be to locate and segment the points that belong to the target weld area, to post process a correct robot trajectory planning.

The core of our algorithm is based on colorimetric segmentation from the point cloud data and post-processing the trajectory calculation. Therefore, the methodology proposes to draw the line where the welding is planned to be applied by coloring with a permanent metallic marker over the workpiece surface, the color marker should be distinguishable from the rest of the workpiece surface. In this case the work piece will be composed of a three-dimensional metal surface that requires a weld bead in a V-type joint. With this approach will be possible to separate the point cloud in two, to post-process only the point cloud that corresponds to the weld bead and smooth the path planning.

3.1 Vision Sensor - Intel RealSense D435.

This work is based on the Intel RealSense D435 [41] an active stereo depth camera that uses Intel's custom ASIC, the Intel RealSense Vision Processor D4, to compute the stereo depth data for real-time. It also has an optional infrared (IR) projector that assists in improving the depth accuracy by projecting a non-visible static IR pattern when the scene texture is low. Additionally, can get up to 848×480 resolution at up to 90 FPS. More characteristics can see in TABLE 2.

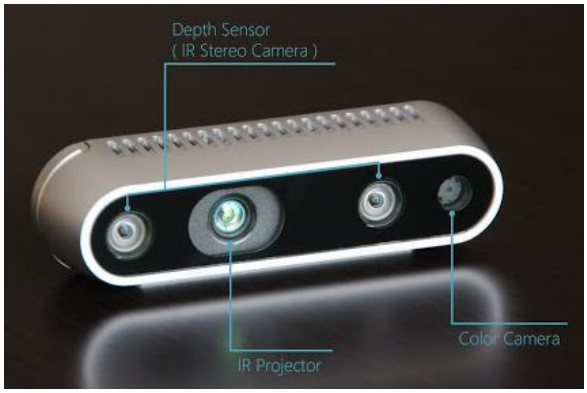


Fig 9 Intel RealSense D435.

TABLE 2 The technical specifications of Intel RealSense D435.

Depth Technology	Active infrared (IR) stereo
Image Sensor Technology	Global shutter: 1.4µm × 1.4µm pixel size
Depth Field of View (FOV) (Horizontal × Vertical) for HD 8:5	91.2 × 65.5° (±3°)
Depth Stream Output Resolution	Up to 1280 × 800 pixels
Depth Stream Output Resolution	Up to 90 fps
Minimum Depth Distance (Min-Z)	0.105 m
Maximum Range	~10 m
RGB Sensor Resolution and Frame Rate	Up to 1920 × 1080 pixels at 30 fps
RGB Sensor FOV (Horizontal × Vertical)	69.4 × 42.5° (±3°)
Camera Dimension (Length × Depth × Height)	90 mm × 25 mm × 25 mm
Connector	USB Type-C

Although in its development, Intel sought to impact the use of technologies such as facial recognition, gaming or augmented reality, due to its technological parameters as well as its compact dimensions and weight, have represented a good low-cost device to perform reverse engineering or to incorporate it to mobile robots, features that have been tested in the articles [41], [42]. In one of Carfagni et al. tests, the camera ability to detect a flat surface was evaluated, as well as the error in depth while the distance to the plane increased. Fig 10 shown the results of both tests, where an improvement in accuracy and precision can be observed with respect to its predecessor, Intel SR300, in which D415 device achieve a reconstruction on the true plane. In addition, with respect to the depth error, the test showed errors below 1 mm in a range below 700 mm.

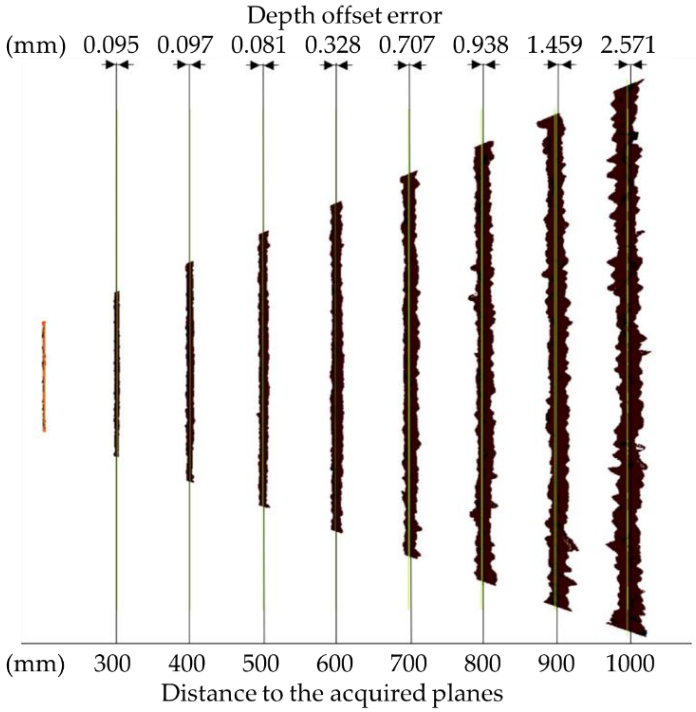


Fig 10 Evaluation of the depth-offset error, measured in (mm), for the target planes in the range 200–1000 mm with a pitch of 100 mm.

3.2 Lighting system.

According to the illumination systems described in the Section 2.3, point cloud tests were performed with dark field and diffuse dome illumination systems. Since these lighting systems can provide homogeneous surface illumination and reduce disturbances on surfaces with high reflective qualities. Next, lighting techniques and sources used to conduct the experiments.

3.2.1 Dark Field lighting.

To build this lighting system, two 18-watt LED bars were used on opposite sides of the work area, in addition to two 10-watt LED bulbs located perpendicular to the LED bars to illuminate the work area from 4 directions. The Fig 11 shows how the LED bars illuminate the work piece from a low angle of incidence following the dark field configuration.

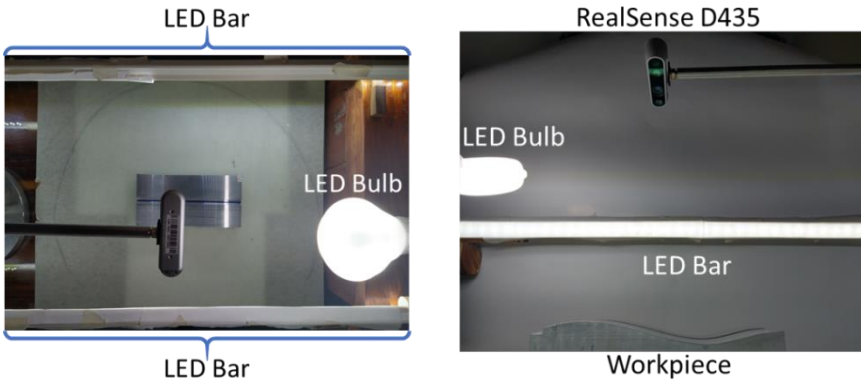
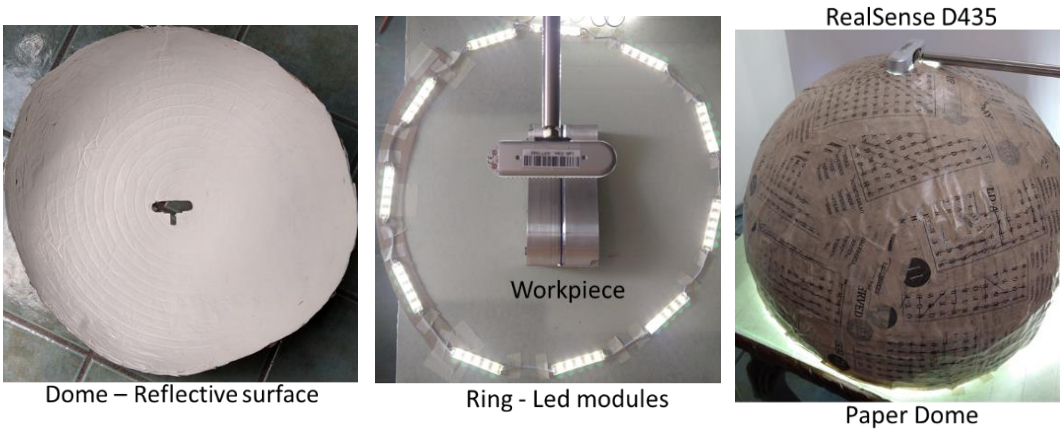


Fig 11 Lighting with dark field configuration, a) top view from the camera and b) side view showing low-angle lighting.

3.2.2 Diffuse dome lighting.

The dome illumination was made by molding a stiffened paper semi-sphere with 28 cm radius to ensure that at least the tests performed at 30 cm allowed the workpiece to remain inside the dome, the inside was coated with white matte paint to produce the reflective zone of the dome. The luminous source was provided by 10 modules LED of 12 Volts in the periphery of the dome, vertically oriented. In the following image we can see the light dome with the camera installed at the top, as well as the central graphic shows the ring of LED modules.



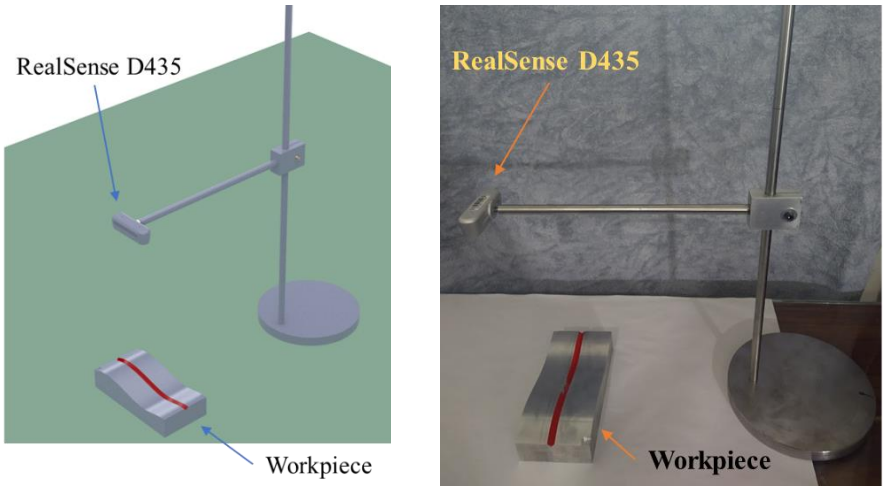
(a) (b) (c)

Fig 12 Dome lighting configuration, a) reflective zone inside the dome, b) ring of led modules and c) dome in operation with the camera in the upper openning.

3.3 Experimental setup.

In order to obtain a comparable result with the proposals presented in the state-of-the-art, where the average error in the location of the trajectories is less than 1 mm [4][26], coupled with the fact that in the tests carried out by Carfagni et al. [42] in 3D reconstruction, the RealSense cameras showed similar results in close ranges.

This study proposes a physical setup, where the D435 camera physically supported on a test arm that allows a vertical movement with an image acquisition from a top view of the workpiece (Fig 13), additional design and constructive considerations for the test platform are described in TABLE 3. At a distance ranging from 30 to 70 cm over the welding work zone, which results in a workspace of between (69 × 39) to (97 × 55) cm respectively.



(a) (b)

Fig 13 Experimental configuration for image acquisition from different heights. a) CAD Design setup. b) The physical configuration implemented.

TABLE 3 Technical considerations for the test platform

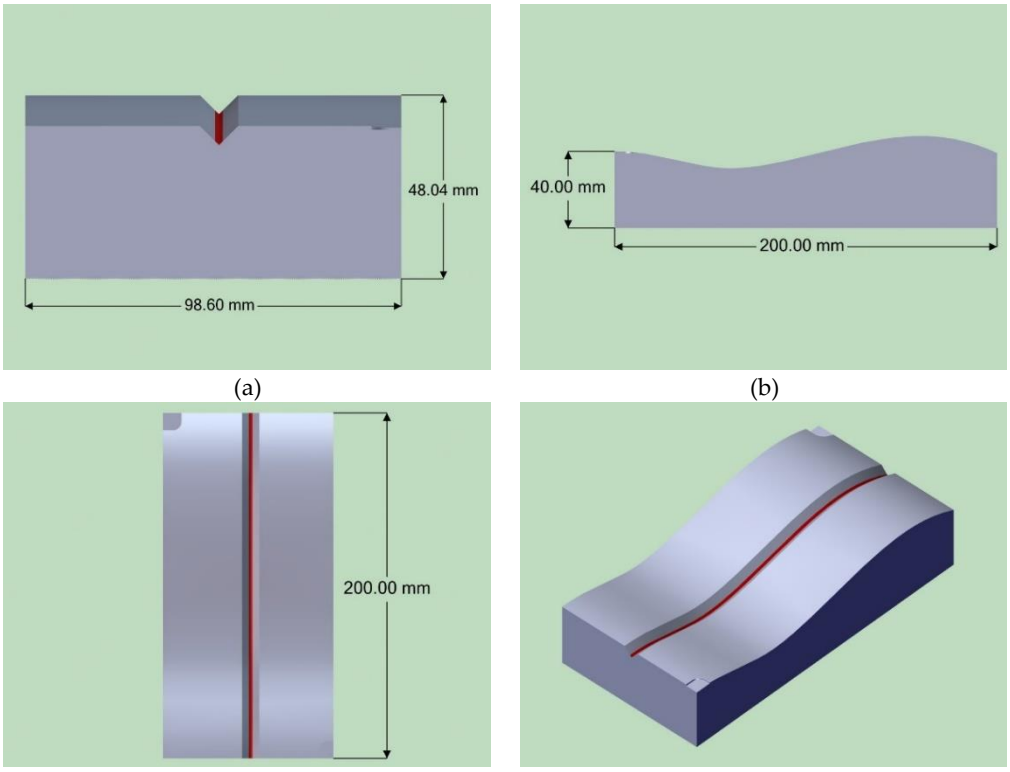
Design	**Materials and Manufacturing**
A base sufficiently large and heavy to prevent movement	8" diameter circular plate with ½" thickness
A round rod that allows for vertical movement	Stainless steel rod in ½" diameter
Horizontal arm support of the camera, which allows the displacement on the pedestal rod.	Aluminum cube with locking slot on pedestal rod
RealSense camera mounting arm with locking nut to secure the camera.	Stainless steel rod in ½" diameter

3.4 Workpiece used for experiment.

The workpieces were designed in such a way that we can know, in advance, the geometric characteristics of the part could be mathematically parametrized. Therefore, it will be necessary to determine which will be our target workspace, to determine the size of the workpieces, as well as the material used. For the purpose in this project, it is proposed that the work objects should not exceed 30 x 30 cm and be manufactured in aluminum 6061 T6, considering that aluminum is a highly moldable and reflective material, which could serve as a parameter to measuring light disturbances in the vision system.

The CAD models of the Fig 14 show the design of the test piece, additional schematic drawings and files can be found in Appendix A, which consists of two pieces designed to simulate a V-type weld joint, one of the most researched in the literature, as shown in Section 1.3. The assembly of these two pieces results in a 20 x 10 cm test piece, which involves an assembly design to guarantee the visualization of a single surface during the tests.

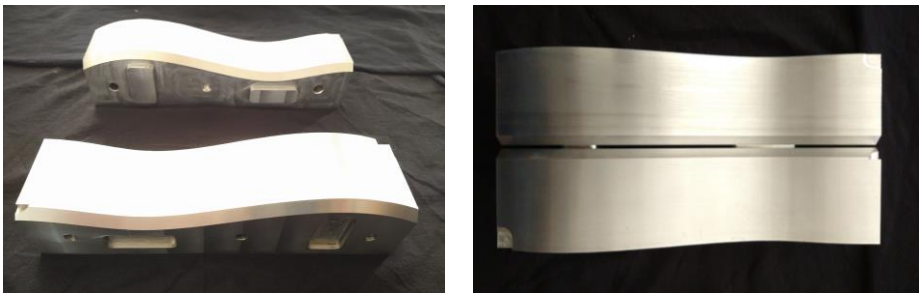
Finally, the design of these parts was carried out in SolidWorks [43] CAD software and later mechanized in a HAAS VF3ss CNC machine, so that the workpiece coincides with the CAD model, since machines like HAAS report positioning errors of less than 0.05 millimeters [44]. The CNC planning control required for the HAAS VF3ss machine was developed using WorkNc [45] CAD-CAM software. In Fig 15 you can see the parts once machined.



(a) (b)

(c) (d)

Fig 14 Test V-type butt sample model with three-dimensional welding trajectory, a) front view, b) side view, c) top view and d) Isometric view.



(a) (b)

Fig 15 Workpiece manufactured in 6061 T6 aluminum, a) separated plates and b) plates joined forming a V-type joint.

3.5 Software for image processing and trajectory planning.

The system execution in which the image acquisition and trajectory planning algorithms were carried out on a personal computer with the Windows 10 operating system and operating with an Intel i7 CPU @ 2.40GHz, with the necessary USB 3.0 ports required for the communication with the RealSense D435 camera.

One of the advantages of working with the RealSense D435 camera, is that Intel has developed an open-source software called Intel RealSense SDK 2.0 [46], which provides the

drivers and communication protocols needed to acquire depth and color streaming. In addition to including the basic applications to display, record and calibrate the sensors. These applications are described below.

- Intel RealSense Viewer: This application can be used view, record and playback depth streams, set camera configurations and other controls.
- Depth Quality Tool: This application can be used to test depth quality, including distance to plane accuracy and Z accuracy.

Finally, the Intel SDK being an open-source software, has support for different programming languages, such as: C ++, Python, MATLAB, LabView and more. For the development of this work, we chose the Python programming language because it is one of the languages supported by the simulation environment that will be presented later in Chapter 4.

Taking all the above into account, TABLE 4 shows the Python libraries used in this methodology to achieve trajectory planning.

TABLE 4 Required libraries for the algorithm development.

Library	**Function**	**Documentation**
Pyrealsense2	The official python wrapper for Intel RealSense SDK 2.0 provides the C++ to Python binding required to access the Intel SDK.	[47]
OpenCV (Open-Source Computer Vision Library)	It is an open-source computer vision and machine learning software library. Which provide more than 2500 optimized algorithms for computer vision applications.	[48]
Numpy	It is a library that allows mathematical and logical operations, among other manipulations in vectorial and matrix arrays, as well as multidimensional arrays.	[49]
Open3D	It is an opensource library that supports 3D data and provides a complete set of basic processing algorithms such as sampling, visualization, and data conversion between point clouds, meshes, and RGB-D images.	[50]
Scipy.interpolate	SciPy is open-source software for mathematics, science and engineering. One of its sub packages is the scipy.interpolate library which has several spline interpolation functions in 1-D and multidimensional.	[51]

3.6 Development of the trajectory planning algorithm.

Fig 16 shows the necessary steps for the definition of parameters and processing of the images that will carry out the extraction of the points corresponding to the weld bead. The main code of the trajectory extraction algorithm can be found in the Appendix B.

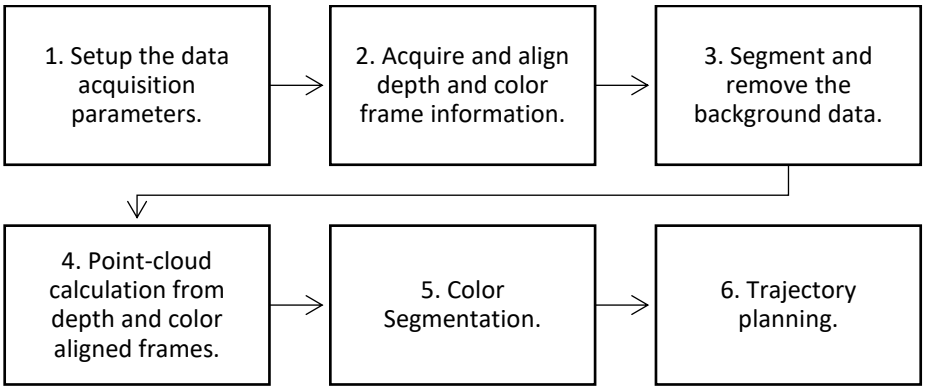


Fig 16 Diagram of the methodology for trajectory planning based on a stereo vision system embedding RGB data.

Following, the objective of each of the blocks of the diagram in the figure is defined.

1. Setup the data acquisition parameters.

As shown in Section 3.1, the vision system implemented has different sensors, which can operate in a wide range of configurations. Therefore, it is important to define within the algorithm the parameters with which the image acquisition is developed.

2. Acquire and align depth and color frame information.

It is necessary to align the depth and color frames to make a 3D reconstruction faithful to the captured scene. This is achieved through the pyrealsense2 [47] library which has an algorithm that aligns the depth image with another image, in this case the color image.

3. Segment and remove the background data.

Sometimes we seek to process a region of interest ROI, in this case the ROI is defined by the distance at which the test object is located relative to the camera. That is why we plan a clipping distance at which all information beyond our ROI is segmented and removed.

4. Point-cloud calculation from depth and color aligned frames.

Once again, the pyrealsense2 library is in charge of calculating the point cloud, since it has the intrinsic values of the stereo vision system and can carry out the calculations presented in Section 2.2 for the point cloud acquisition.

5. Color Segmentation.

This block represents the core of the proposed methodology, in which the point cloud is vectorized to an XYZRGB format using the Numpy and OpenCV library tools. Two types of segmentation are implemented, first as the RealSense sensor incorporates an RGB sensor which by default provides the color information of the scene in an RGB color space, a color segmentation is applied by filtering the color channels depending on the marker that was applied on the workpiece.

However, RGB is additive color model in which red, green and blue mixing components

produce a non-uniform characteristics property in chrominance and luminance. For that this color model is not preferred for color based detection and color analysis. The literature has found that HSV color space is one of the most robust color spaces for color segmentation when the illumination changes on the object surface. HSV color model is human visual system that decouple the lightness, brightness or value information from the chromatic information.

Taking the above into account, to improve the selection of the points of interest, a change in the color space to HSV (Hue Saturation Value) was used. A threshold was applied to the Hue channel to find the color region, as well as to the saturation channel as a parameter for brightness.

6. Trajectory planning.

In order to calculate the trajectory from the segmented data in the previous module, a spline cubic interpolation algorithm is implemented to obtain a smooth planning of the target weld seam points [27], [52], [53].

Chapter 4 Methodology for a simulated test of a trajectory planning, based on a stereo vision system embedding RGB data.

Due to the circumstances caused by the COVID-19 suffered during the development of this project, which did not allow the normal use of university facilities and laboratories. It was decided to develop a simulation methodology to validate the planning of trajectory presented in the Chapter 3 over an industrial manufacturing environment. In this work is simulated the acquisition of images and trajectory planning applied to industrial robot inside an environment of simulation. In the Fig 17 the operation diagram of this methodology can be observed, whose blocks will be described in the following subsections.

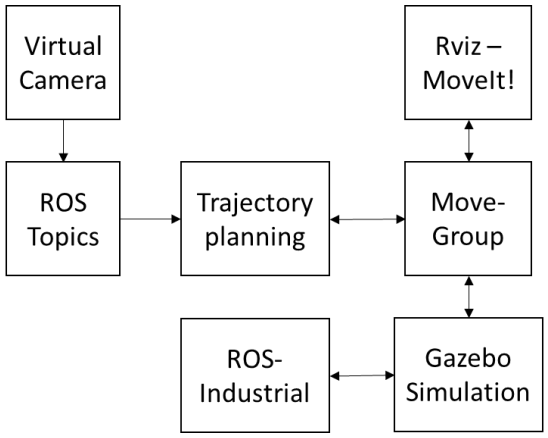


Fig 17 Basic diagram showing the infrastructure to simulate the trajectory planning algorithm in ROS.

4.1 Robot Operating Software ROS.

ROS is an open-source platform that incorporates a collection of tools and libraries to predict the behavior of robust robotic systems, with wide adoption in academia and industry around the world. The ROS framework uses the concept of packages, nodes, topics, messages and services, that allow the software developers to fulfill their philosophy: 1. Peer to peer technology (a number of processes connected at runtime), 2. Multilingual (C++, Python, Octave), 3. Tools-based, 4. Thin (all driver and algorithm development to occur in standalone libraries that have no dependencies) and 5. free and Open-Source [54]. To work with this software, although there is a version of ROS for Windows, the software has main support in the operating system Linux, for that reason was selected as operating system, Ubuntu 18.04.3 LTS and ROS Melodic Morenia, which have support until 2023.

Some of the tools that will be used within the ROS environment will be the following:

1. **Rviz**: Many current ROS applications have a robot which provides an item for its visualization. The given ROS-topic can contain a single geometric primitive such as a

sphere or a box, a list of points or triangles, or can point to a 3-D model. In such a way, the visualization engine Rviz (Robot Visualizer) receive the message and display it to the user [55].

2. **Gazebo**: A 3-D simulator designed to accurately reproduce the dynamic environments a robot may encounter. That is capable of simulating a population of robots, sensors and objects, to generates both realistic sensor feedback and physically plausible interactions between rigid body [56].
3. **MoveIt**!: Set of tools for mobile manipulation in ROS. that solves the problems of movement planning, manipulation, 3D perception, kinematics, control and navigation. It provides an easy-to-use platform to develop advanced robotic applications, evaluate new robot designs and build integrated robotic products. Also, there is a RViz plugin, which enables motion planning from RViz itself [57].
    a. **Move_Group**: The main node of the MoveIt! software which incorporates several tools. Among which provide the infrastructure to manipulate the robot's information such as point cloud, joint state of the robot, and transform (TF) of the robot in the form of topics [58].
    b. **Planning Groups**: ROS integrates KDL (Kinematics and Dynamics Library) an OROCOS library that provides kinematics solving for a large number of kinematic chains built in a robot description file (URDF) [59], as well as forward and inverse kinematics computation. The KDL, is a numerical Jacobian based solver that solves for the joint angles by solving the linear least squares problem, over a number of iterations, using Newton-Raphson (NR) [60].

The Fig 18 shows how our proposal described in the diagram in the Fig 17 works over the architecture of the ROS modules where move_group node serves as an integrator that provide a set of ROS actions and services. First the user manages the python algorithms that extract the trajectory and plan the robotic movements and on the other hand we incorporate the information coming from the 3D sensor, as well as the communication with the simulated robot.

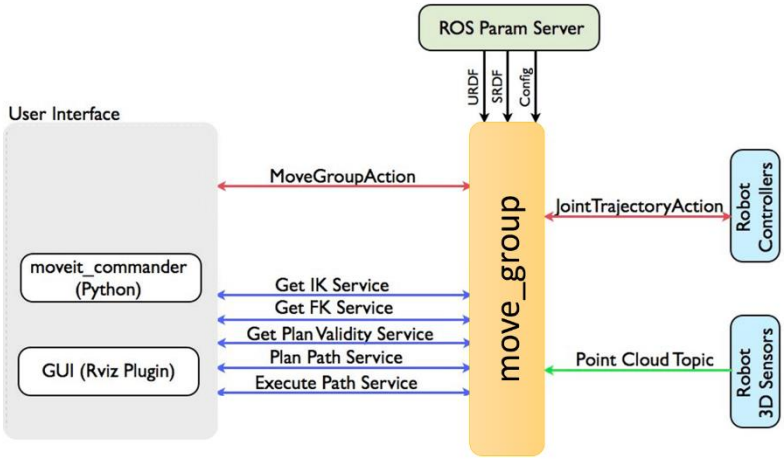


Fig 18 ROS-MoveIt! architecture diagram.

4.2 Virtual vision system.

The project involves the simulation of image acquisition for the welding path. First it was necessary to evaluate that the camera presented in Section 3.1 is capable of working within the proposed virtual environments. This means that first it will be necessary to create a ROS package in which the capacity of the RealSense D435 camera to generate data through the ROS topics will be tested. These tests will be carried out with the INTEL RealSense SDK and the ROS package that INTEL developed for its cameras. As mentioned in Section 3.5, this software can be found at [61] for the Linux distribution and the Intel RealSense camera integration package at [61]. It is important to mention that at [62] we can find a good description of the steps to follow for the installation of these packages in ubuntu.

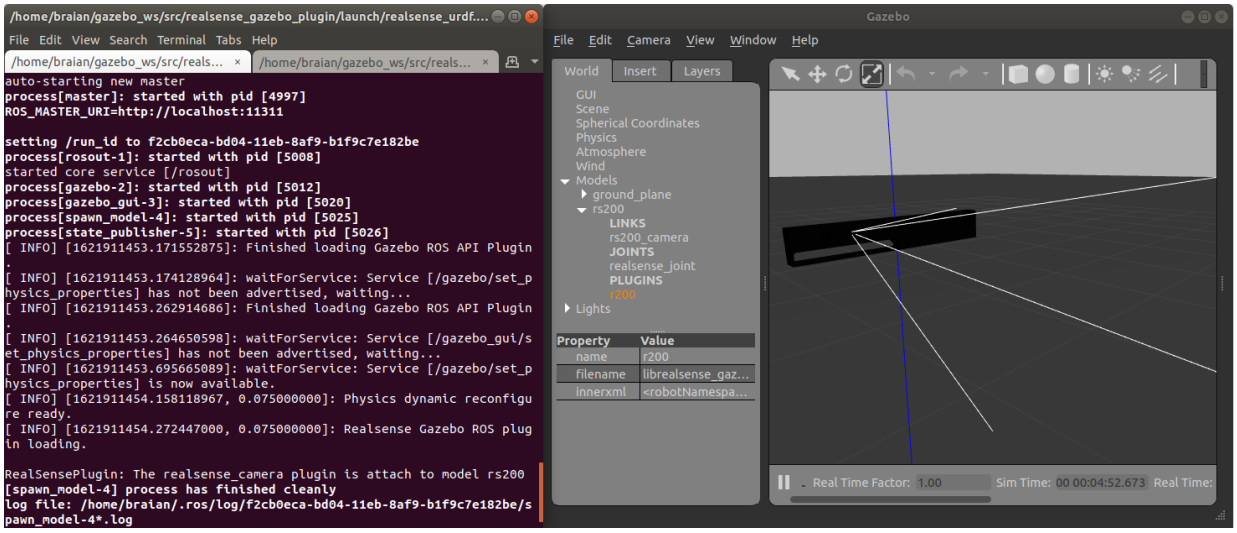
However, even though the software mentioned above is capable of providing information from the physical camera to virtual environments. There is no plug-in from Intel that can simulate image acquisition in the gazebo simulation environment. Nevertheless, it is possible to create sensors for this simulation environment, as is the case with some other more studied sensors such as Kinect [57]. For the development of this thesis, the sensor created by Salah Missri [63] was selected

4.2.1 RealSense Gazebo Plugin.

Salah Missri [63] developed the necessary packages to simulate the RealSense R200 model and connect the sensors with ROS, this package can be found at [64]. Among other things, this package incorporates two basic tools that can be launched into the ROS environment with the following command lines:

```
$ roslaunch realsense_gazebo_plugin realsense_urdf.launch
```

This tool launches virtual configuration of the camera, enabling the stereo vision sensors, infrared sensors and the RGB sensor, and their respective communication protocols with ROS. Fig 19 shows the launch file execution in the command window and how it is displayed the virtual model of the camera in Gazebo.



(a) (b)

Fig 19 Virtual camera launch file, a) launch file execution in the command window and b) camera model displays at Gazebo.

```
$ roslaunch realsense_gazebo_plugin depth_proc.launch
```

This launch file reads the information of the ROS topics that subscribe the information of the virtual camera sensors, to calculate the point cloud from the depth image. The result is a point cloud data subscribed to the ROS topic known as pointcloud2, which packs the XYZRGB information of the captured scene [57], Fig 20 shows the status of connect the virtual camera to ROS topics in the command window.

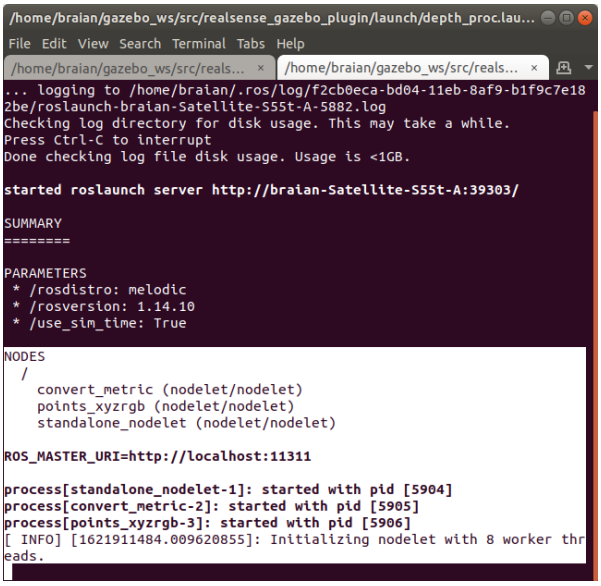


Fig 20 Virtual camera point cloud launch file status in the command window.

4.3 ROS-Industrial.

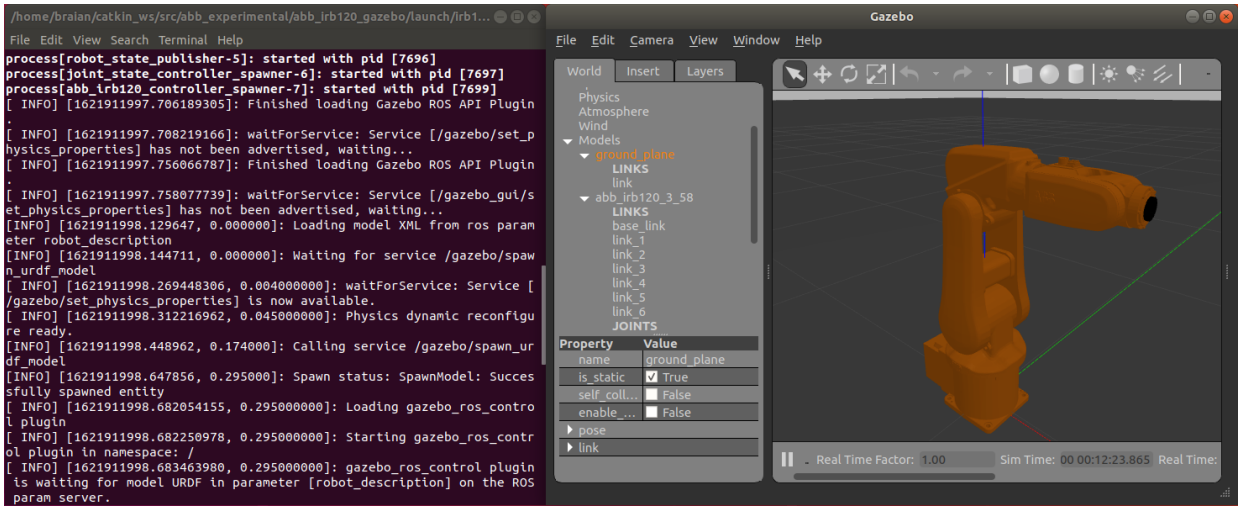
ROS software offers advanced capabilities for building and predicting the behavior of robotic systems, however, to extend these capabilities to manufacturing industry and operate common manipulators. The open-source ROS-Industrial is an interface that seeks to solve the control of industrial manipulators and grippers through the tools of MoveIt!, Gazebo and Rviz [57]. Currently the manipulators from the vendors ABB, Adept, Fanuc, Motoman and Universal Robots are supported by ROS Industrial. More information can be found at [65].

4.3.1 ABB ROS-Industrial

Since ABB is a very widely supplier of welding robots, the ABB Experimental package was selected, which has the support for the simulation in gazebo of the ABB IRB 120 and ABB IRB 1200 variants. The files necessary to operate these robots can be found in [66]. In addition, it is important to mention the robot ABB IRB 120 selected for the test of simulation of the present thesis, runs under the following lines of command:

```
$ roslaunch abb_irb120_gazebo irb120_3_58_gazebo.launch
```

Launch the robot model ABB IRB 120 into gazebo and load the ros_control controllers. Fig 21 shows the launch file execution in the command window and the robot displays in the virtual environment of Gazebo.



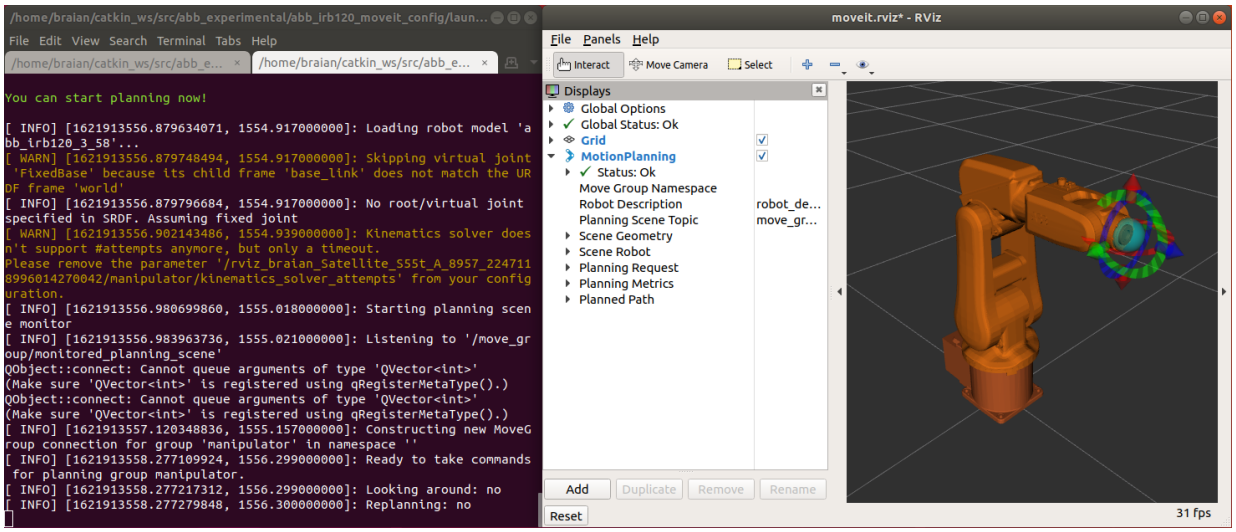
(a) (b)

Fig 21 Gazebo ABB IRB120 launch file, a) launch file execution in the command window and b) Robot ABB IRB120 displays at Gazebo.

```
$ roslaunch abb_irb120_moveit_config moveit_planning_execution_gazebo.launch
```

Launches Moveit! and ensures communication between Gazebo and the Move_Group robot description. Fig 22 shows reports that communication with the robot's move_group has been

established and shows the window for manipulation and path planning in the MoveIt interface.

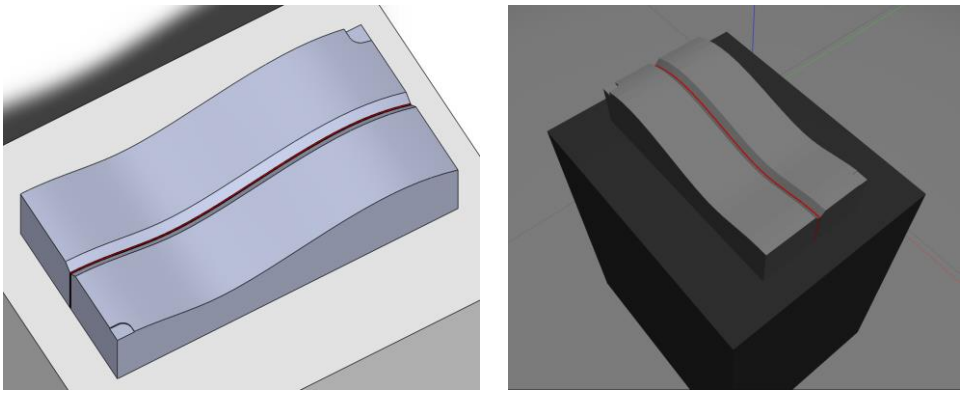


(a) (b)

Fig 22 MoveIt! ABB IRB120 launch file, a) launch file execution in the command window and b) MoveIt! interface for Robot ABB IRB120 manipulation and path planning.

4.4 Simulated test sample.

The sample part for testing is the same as described in Section 3.4. However, it is required to convert the CAD models into a Gazebo simulation environment. To achieve this the tool presented in [67] for the conversion of a SolidWorks CAD to a URDF (Unified Robot Description Format) file that incorporates the CAD model visual, as well as the kinematic and dynamic characteristics. However, to have a workpiece that would show in the simulation the target zone marked with a different color than the surface, it was necessary to divide the piece into three parts, where the central piece represents the target zone and in which a different color would be applied within the URDF file. In the Fig 23 you can see the test piece in CAD version and the visual result in Gazebo.



(a) (b)

Fig 23 Part design to be converted to URDF file. a) CAD version, b) Gazebo simulation

4.5 Trajectory planning algorithm in a Gazebo simulation.

Conceptually the algorithm to be implemented in this part is the same as the one described in chapter 3, however since the images from the RealSense D435 camera are no longer acquired, the pyrealsense library is no longer necessary. Instead, all the libraries related to the ROS simulation system are added. For example, we can subscribe to the ROS-topics and receive the information from the virtual camera or control the manipulator through these libraries. TABLE 5 is a list of the libraries used and their functions.

TABLE 5 Algorithm libraries within the ROS environment.

Library	**Function**	**Documentation**
Rospy	A library for ROS. That enables Python programming interface with ROS Topics, Services, and Parameters.	[68]
CvBridge	The ROS delivers the images data as sensor_msgs/Image ROS topic messages, CvBridge is a ROS library that provides an interface between ROS and OpenCV.	[48]
Ros_numpy	Library for converting ROS messages to and from numpy arrays. In particular it provides the point_cloud2 sub package that converts the ROS message "pointcloud2" into an XYZ array.	[69]
Moveit_commander	Provide functionality for most operations that the MOVE_GROUP require, such as setting joint or pose goals, creating motion plans and moving the robot	[70]

Chapter 5 Experiments and results.

After analyzing in detail, the theoretical justification, as well as the considerations taken in the implementation, we proceed to study the performance, in a first point of the proposed vision system and in a second step the trajectory planning by the proposed algorithm.

First, the proposed experiments are described to evaluate the objectives set out in the first chapter, and then present the graphics, tables or images acquired with the performance obtained. Finally, in the discussion analyze the results obtained.

A welding robot is expected to locate the weld bead in the same place repeatedly, in terms of repeatability most welding robots satisfy this aspect with repeatability value at the order of 0.1 mm [71]. However, it is important to set the tolerance on the fixture and component dimensions within the specified limits. The variation in the positioning of an arc welding robot should not exceed ±0.5 mm [72]. That is why the state of the art seeks to obtain errors in the extraction and trajectory tracking below 1 mm.

5.1 Depth map acquisition tests at different heights from a flat surface.

In order to evaluate the performance of the RealSense camera, we proceeded to execute the methodology described by Carfagni et al [42], which seeks to measure the error with which the sensor is capable of measuring the distance to a flat surface. For this experiment we used the test platform described in Section 3.3 and the RealSense Quality Tool software. These tools were used to obtain the XY density of the points acquired in a height range of 200-500 mm, plus the RMS error to acquire the plane. In the TABLE 6 the results obtained can be seen, where similar results are found to the RealSense D415 presented by Carfagni et al [42] in which the D435 model is also capable of capturing a flat surface without apparent distortion and an error percentage with an accuracy below 1% to the true plane.

TABLE 6 Depth error to acquire a flat surface.

Distance mm	X mm	Y mm	Depth scale mm	Plane fit RMS
200	0.324	0.238	± 1	0.01%
300	0.486	0.197	± 1	0.04%
400	0.648	0.238	± 1	0.08%
500	0.808	0.233	± 1	0.13%

It is important to mention that the distance between the camera and the target surface was physically measured with a Fowler caliper height gauge, Fig 24.

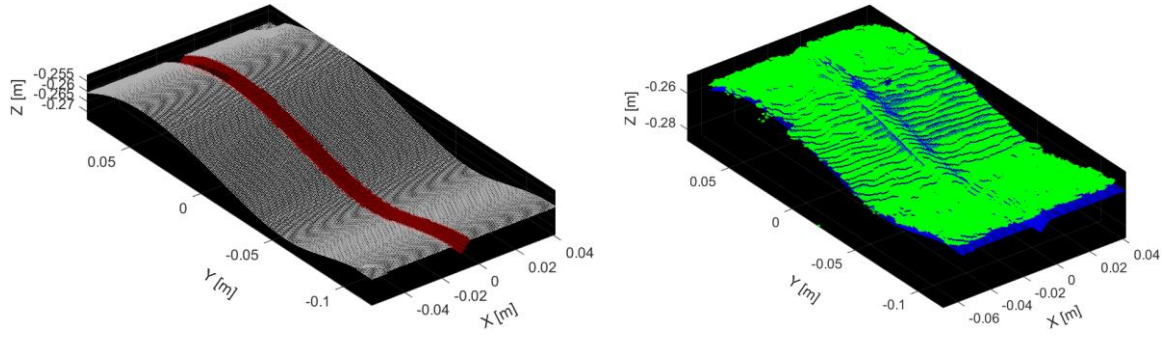
Fig 24 Fowler caliper height gauge.

5.2 Results of the reconstruction through point cloud with RealSense D435 sensor.

The setup was having the RealSense D435 camera located at 30 cm from the top of a flat surface, where the test piece described in Section 3.4 was placed. With this configuration the 3D reconstruction of the surface was carried out, through the first three blocks of the algorithm presented in Chapter 3 to finally obtain the point cloud of the test-piece.

The real point cloud of the test piece was generated by the SolidWorks software which allows exporting the pieces to a Polygon File Format (.ply). Once we have the target surface and the one calculated by the camera, we proceed to run the Iterative Closest Point ICP registration algorithm with which we can estimate the Euclidean point distance [73] between the target surface and the 3D reconstruction.

Fig 25a shows the point cloud of the actual target surface. On the other hand, the Fig 25b shows the result of the registration between both point clouds, implemented with the algorithm described in Section 2.4.2 which gives us the distance between the points of the 3D reconstruction to the closest point on the target surface. Three tests were carried out and the results are shown in the TABLE 7 where the average distance and standard deviation is computed, the results prove that in terms of reconstruction the sensor provides a standard deviation between points below 1 mm, which means that a welding trajectory extraction with an error in the same proportion would be expected.



(a) (b)

Fig 25 3D Reconstruction evaluation. a) Target point cloud, b) ICP Color registration result between target and 3D.

TABLE 7 Average and standard deviation between target and point cloud reconstruction.

	Average	Standard deviation
Test 1	0.704 mm	0.378 mm
Test 2	1.053 mm	0.623 mm
Test 3	1.284 mm	0.738 mm

5.3 Testing Trajectory Extraction of a V-type butt joint.

Once again, these experiments were conducted with the test stand described in Section 3.3 at a distance of 300 mm above the surface. However, to carry out this experiment it was first necessary to perform experiments to select the best lighting technique and the way in which the color segmentation filters would be applied, in an RGB or HSV color space.

5.3.1 Previous work - Testing Trajectory Extraction with RGB segmentation and dark field lighting.

The following subsections show the results in path extraction tests obtained with dark field illumination. It will be presented in two subsections. First the results on color segmentation in RGB color space, second the trajectory extraction tests.

5.3.1.1 RGB point cloud segmentation with dark field lighting.

The experiments described below were carried out with dark field illumination. It has take into account that the RealSense sensor incorporates an RGB sensor which by default provides the color information of the scene in an RGB color space (RED, GREEN, BLUE) in the range of 0 to 255 for each channel, giving up to 16,777,216 possible values per pixel. That is why we made the tests where the weld area of the test piece was marked with the primary colors RGB and proceeded to run the color segmentation algorithm and evaluate the visual performance.

Commonly a pixel showing a pure red color would have a value of (255, 0, 0). However, in the code developed in Python, we would get a value of (0, 255, 255). Taking this into account the segmentation algorithm looks for all those points that have an RGB value below the control parameter.

Fig 26 shows that it was possible to segment the welding area from the rest of the surface with the three test colors, however as the image shows the tests with red and blue were achieved with a control value over the threshold of each color channel implemented in python equal to 0-5 and for the test of the green color it was necessary to set this parameter to 0-15. Another conclusion of these tests is that the test with the blue and green color, visually show a weld bead with an RGB value that would correspond to the color black, which sometimes the algorithm can confuse with areas that do not belong to the target area.

Fig 26 Color segmentation results with dark field lighting.

Taking this into account and given that in numerous tests, the red color segmentation was the one that showed the best results, it was decided that the color for the trajectory extraction tests would be carried out with the color red.

5.3.1.2 Testing trajectory extraction with dark field lighting.

In this stage it was implemented in its totality the experimentation of the algorithm presented in Chapter 3. In which once captured the zone where it is intended to apply the weld bead, the algorithm implements a spline cubic interpolation. Which calculates the path that smoothest the planning of the welding points that would require a robotic manipulator.

In order to evaluate the computed path, we proceeded to obtain the points interpolated over this trajectory from the test piece designed in the SolidWorks software, to later compare and match with the computed path through the ICP algorithm and calculate the RMSE error of each of the points. Fig 27 shows the 3D representation of the target path vs. the calculated path.

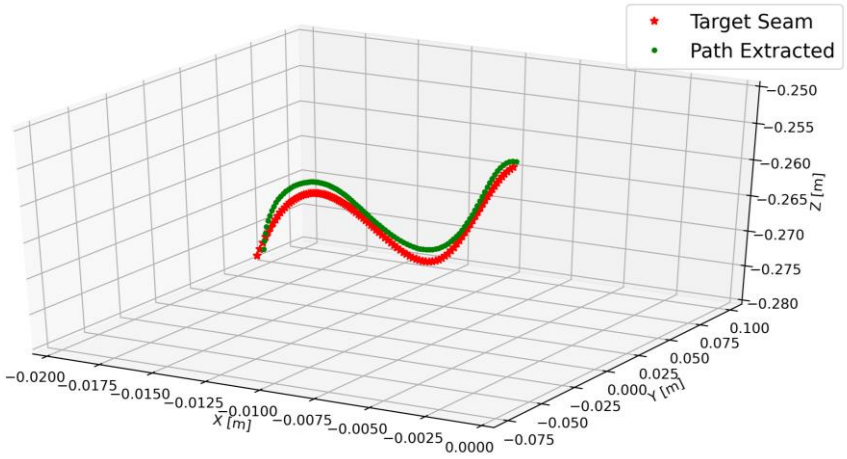


Fig 27 Target path vs. computed trajectory.

The TABLE 8 shows the RMSE values in three tests were conducted and a comparison with the work done by Yang et al. [53] in which it is observed that the results obtained with our proposal show greater error in the planning of the trajectory. Although the error values for Y and

Z are higher than 1 mm, it is important to mention that the points of the calculated path do not follow a linear distribution since the cubic spline interpolation incorporates a higher number of points where it is required to smooth the path. To better understand the behavior of the points, the XYZ values were plotted in Fig 28. In these plots the values for Z are the ones that really shows a significant displacement of the desired trajectory.

TABLE 8 Trajectory RMSE error with dark field lighting.

Test	Comparison method	X	Y	Z
1	CAD	0.047 mm	0. 694 mm	1.497 mm
2		0.050 mm	1.377 mm	1.636 mm
3		0.024 mm	1.011 mm	1.698 mm
Yang et al. [53]	Laser sensor	0.67 mm	0.75 mm	0.81 mm

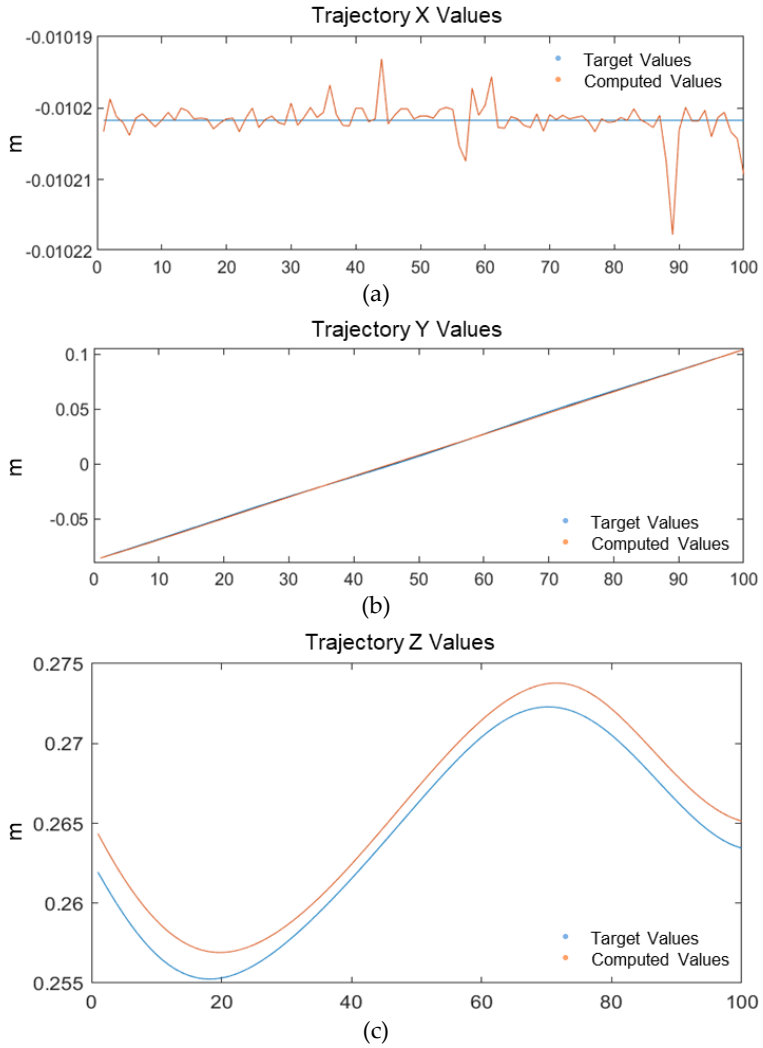


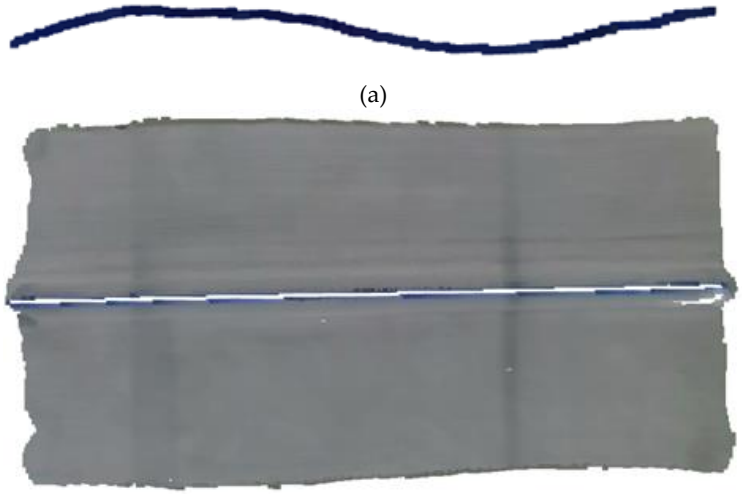
Fig 28 Graphs of the XYZ values of the real vs. the computed trajectory, a) X values, b) Y values and c) Z values.

5.3.2 Previous work - Testing Trajectory Extraction with RGB segmentation and with Dome lighting.

The following subsections show the results in path extraction tests obtained with dome lighting. It will be presented in two subsections. First the results on color segmentation in RGB color space, second the trajectory extraction tests and a comparative with the results obtained with dark field lighting.

5.3.2.1 RGB point cloud segmentation with Dome lighting.

The experiments described below were carried out with dome illumination, and once again following the camera's own RGB channel segmentation methodology, Fig 29 shows the result of segmenting the blue marker on the workpiece, as can be seen both the surface and the segmented area have a uniform illumination compared to the dark field illumination system. It is also worth mentioning that the acquisition of the point cloud is carried out in a direct way, unlike the previous system in which the manipulation of the angle and position of the light influences to achieve the results shown.



(a)

(b)

Fig 29 Color segmentation results with dome lighting, a) Colored points segmented and b) the remaining surface of the workpiece.

The image above shows only the color segmentation over the workpiece, and although the segmentation of the three colors was carried out correctly without presenting the problems of the dark field illumination system, segmentation tests were performed on a dark surface, for this purpose, Fig 30 shows a 2.5" black (RHS) rectangular hollow structure was used, in which color markers: red, green, blue, gold and silver were applied to analyze the segmentation by filtering the RGB channels.

Fig 30 Rectangular hollow structure RHS with red, green, blue, gold and silver color markers.

In these tests colors red, green and blue were not possible to be segmented correctly since they have RGB values very close to the surface, on the other hand the gold and silver colors could be segmented, however an additional filter would be necessary to eliminate the contour noise of the piece that remained in the point cloud, as shown in Fig 31.

Fig 31 Gold color marker segmented.

5.3.2.2 Testing trajectory extraction with Dome lighting.

Following the methodology previously used with lighting system for the calculation of the trajectory, this time the blue color segmentation was selected for these tests. Given the RMSE analysis problems in the previous tests, the strategy to evaluate the computed trajectory was changed, once the trajectory was computed, the points of the target path were obtained by filtering the points that have the smallest Euclidean distance between the two trajectories, thus obtaining the points of the target path that follow the same distribution as the computed trajectory. The Fig 32 shows the 3D representation of the target path vs. the calculated path.

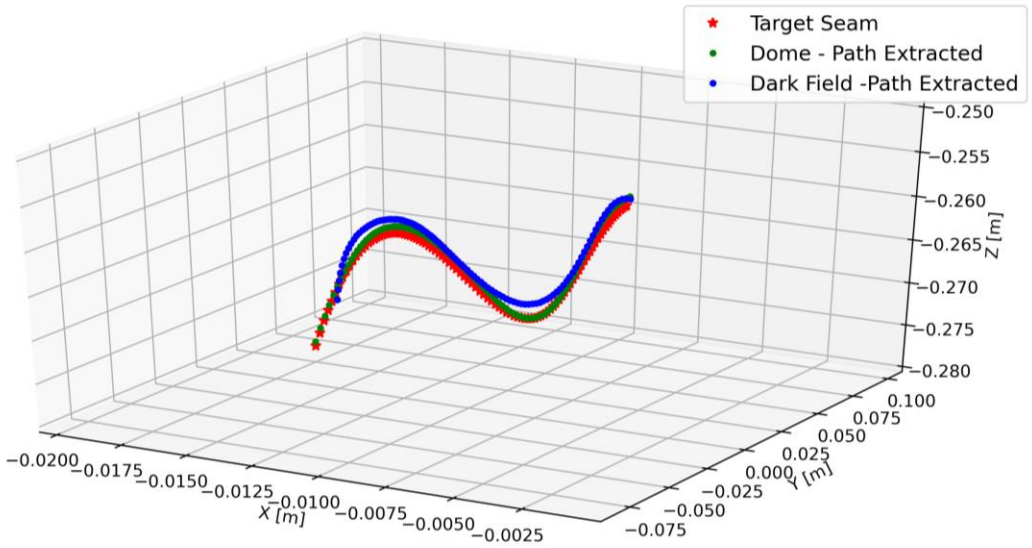


Fig 32 Target path vs. computed trajectory with dome and dark field lighting.

The TABLE 9 shows the RMSE values in four tests where it can be observed that only one error persists in Z, however, according to the previous results TABLE 8, the offset with the surface is reduced, in addition, the error in the Y coordinates no longer shows a mismatch. Comparing these results with the work of Yang et al. [53], we can already observe that in some tests we have comparable results in the Z-error, however the error range is higher, oscillating between 1.15 and 0.75 mm.

TABLE 9 Trajectory RMSE error with RGB Dome lighting.

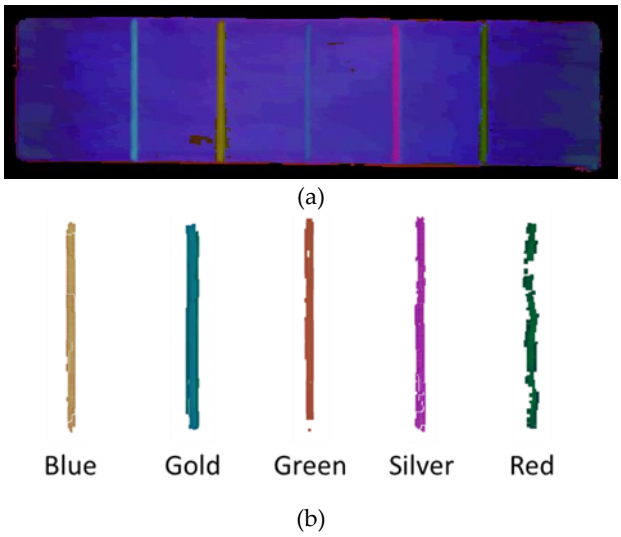
Test	Comparison method	X	Y	Z
1	CAD	0.036 mm	0. 161 mm	0.763 mm
2		0.038 mm	0.166 mm	0.748 mm
3		0.037 mm	0.212 mm	1.099 mm
4		0.033 mm	0.221 mm	1.140 mm
Yang et al. [53]	Laser sensor	0.67 mm	0.75 mm	0.81 mm
Zhou et al. [29]	teaching-playback	Pose error equal to 0.7 mm		

5.3.3 Testing Trajectory Extraction with HSV segmentation and with Dome lighting.

After improving the results obtained with the dome lighting system, we proceeded to evaluate a second color segmentation technique, due to the problems encountered in section 5.3.2.1 for color segmentation with RGB color space on surfaces different from the workpiece. The following subsections show the results in path extraction tests obtained with a color segmentation in a HSV color space. It will be presented in two subsections. First the results on color segmentation in HSV color space, second the trajectory extraction tests and a comparative result obtained with RGB segmentation.

5.3.3.1 HSV point cloud segmentation with Dome lighting.

Taking the results with dome illumination and due to the weakness of RGB segmentation over dark surface in an RHS, tests were carried out in HSV color space, because this color space resembles the perception in which human beings perceive colors and is less sensitive to light. Fig 33 shows that in an HSV color space it was possible to segment the five colors applied to the RHS, in the upper part we can see the transformation of figure Fig 30 to HSV color space and in the lower part the segmented point clouds.



(a)

(b)

Fig 33 a) HSV image of RHS with red, green, blue, gold and silver color markers and b) color segmentation of each color marker.

As mentioned before for trajectory extraction over test sample, the segmentation was performed by applying a threshold in the HSV color space channels, using Hue channel to find the color region, as well as to the saturation channel as a parameter for brightness. TABLE 10 shows the thresholds applied to achieve the segmentation of each color marker.

TABLE 10 Color threshold for point cloud segmentation in HSV channels.

	Hue	Saturation
Test RED	160 -180	100 - 255
Test GREEN	30 - 50	80 - 255
Test BLUE	110 - 120	110 - 255

Fig 34 shows the result of generating the point cloud of the test piece to which a blue color marker was applied in the weld zone, on the left is the target point cloud with color information in HSV color space, while the image on the right shows the result of the segmentation of the weld bead by applying the color filter to the point cloud.

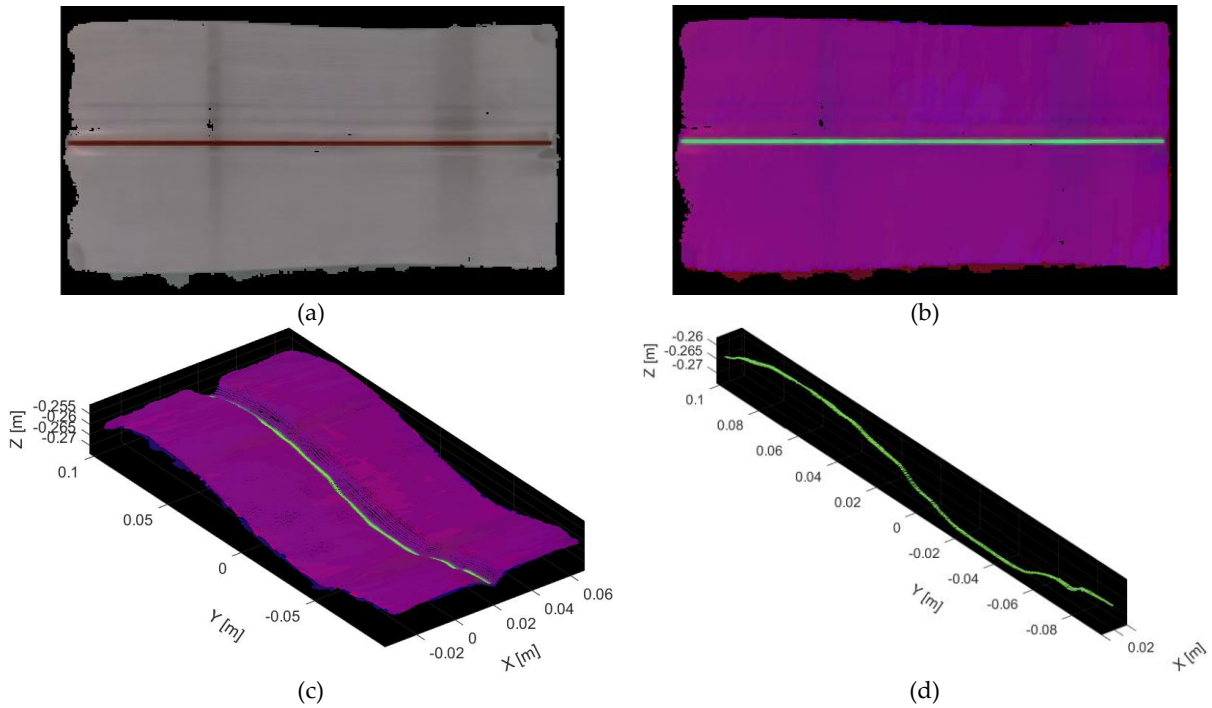


Fig 34 Color Segmentation. a) RGB image, b) Image with HSV transformation, c) Point Cloud with HSV data and d) Points of seam filter by color segmentation.

5.3.3.2 Testing Trajectory Extraction of a V-type butt joint.

Once the segmentation with HSV values was tested, the previous algorithms are applied to compute the trajectory, Fig 35 shows the 3D representation of the target path vs. the calculated path.

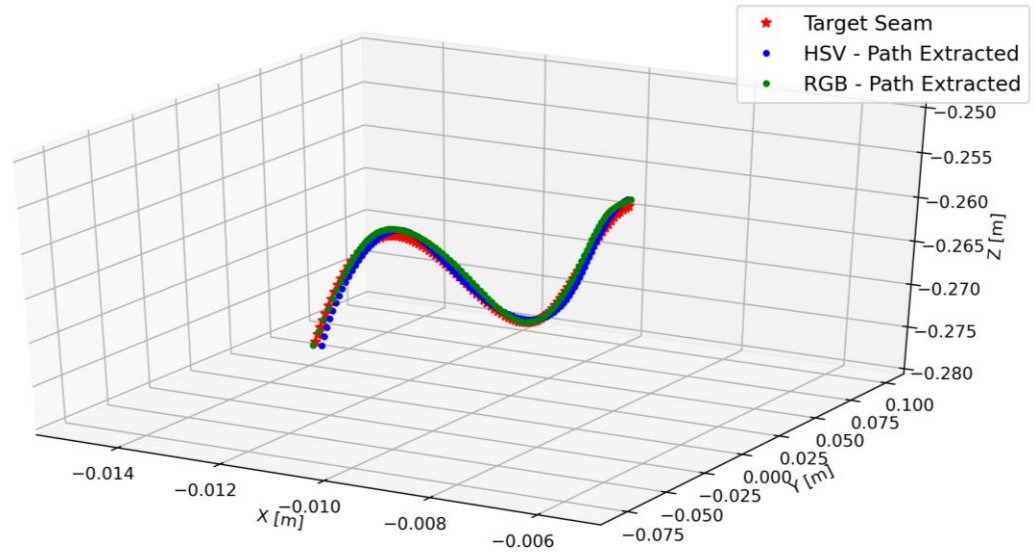


Fig 35 Target path vs. computed trajectory with RGB and HSV color space.

The TABLE 11 shows the RMSE values in four tests were conducted. If we compare the results with those obtained in the RGB segmentation, we can see that the color space transformation does not influence the error of the trajectory computation, the difference lies in the ability to segment where the HSV space helps the processing of data on other types of surfaces.

TABLE 11 Trajectory RMSE error for V-type butt joint with HSV color space.

Test	X	Y	Z
1	0.063 mm	0. 184 mm	0.952 mm
2	0.046 mm	0.195 mm	1.059 mm
3	0.010 mm	0.145 mm	0.739 mm
4	0.011 mm	0.145 mm	0.647 mm

In order to have a better visual interpretation of the trajectory estimation, a point cloud was constructed where the points of the area marked with color are substituted by the calculated path, as can be seen in Fig 36.

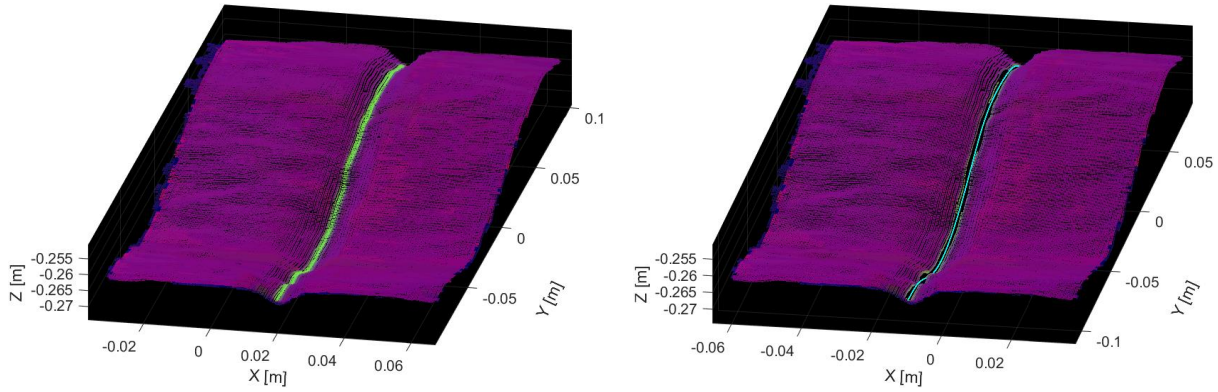
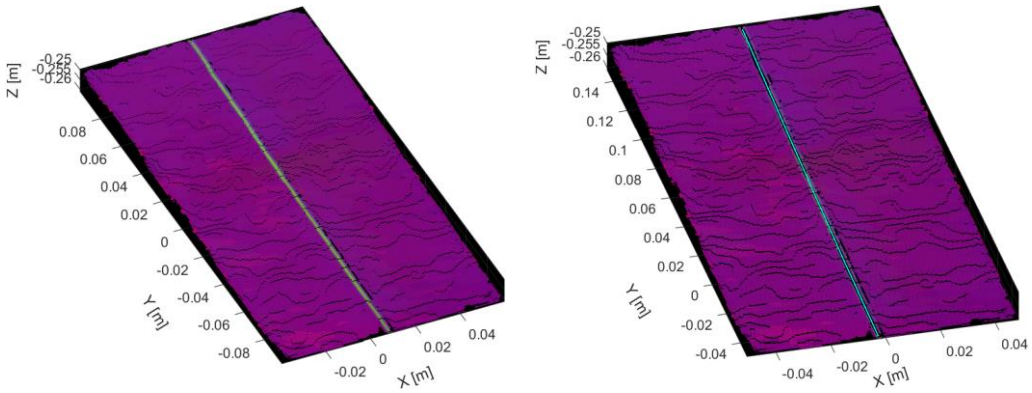


Fig 36 V-type butt joint trajectory extraction a) Point Cloud with HSV data, b) Surface with the computed path.

5.4 Testing Trajectory Extraction of a straight butt joint.

As mentioned before there are five basic welding joint types commonly used in the industry, so a straight butt joint was constructed with a length of 20 cm and an inclination of 3° degrees above the surface in order to demonstrate the flexibility of the system. Applying the previous algorithms, it has also been possible to extract the trajectory, Fig 37 show the tested surface reconstruction to which a straight blue line was applied and the image on the left shows the trajectory calculated over the point cloud surface.



(a) (b)

Fig 37 Straight butt joint trajectory extraction a) Point Cloud with HSV data, b) Surface with the computed path.

The TABLE 12 shows the RMSE values and the average and standard deviation between the calculated trajectory and the desired line model trajectory, Fig 37, within three tests that were conducted. Similar findings to previous results in RMSE and standard deviation show the flexibility of the system as a global acquisition system regardless of the workpiece.

TABLE 12 Trajectory RMSE error for a straight butt joint.

Test	Comparison method	X	Y	Z	Average	Standard deviation
1	CAD	0.142 mm	0.075 mm	0.683 mm	0.6 mm	0.2 mm
2		0.124 mm	0.072 mm	0.530 mm	0.5 mm	0.2 mm
3		0.180 mm	0.069 mm	0.494 mm	0.5 mm	0.2 mm
Fillet Joint Yang et al. [25]	teaching-playback	0.64 mm	0.67 mm	0.73 mm	-	-
Y-shaped joint Zhou et al. [29]	teaching-playback	Pose error equal to 0.8 mm			-	-

5.5 Experimental results in a simulation environment.

As mentioned in Chapter 4, to evaluate the proposed vision system, integrated to an industrial robot. For which, it will be evaluated in this section with the methodology described in the previous Chapter. The files needed to run the simulation presented here can be found in the Appendix C.

The components that are displayed in the Gazebo environment (ABB IRB120 robot, virtual camera and workpiece) were already described in the previous chapter, however it is important to describe the location of these components in relation to the coordinated space of the simulation environment, TABLE 13 shows this information.

TABLE 13 Location of the components within the virtual environment

Component	X	Y	Z
ABB IRB 120	0 m	0 m	0 m
Virtual camera	0.25 m	-0.25 m	0.8 m
Workpiece	0.25 m	-0.25 m	0.3 m

5.5.1 Experimental results in a simulation environment.

For the development of this experiment, we implemented the algorithm described in Chapter 4, which allows us to communicate with the ROS simulation environment and acquire the information from the virtual camera. For the implementation of this experiment a roslaunch file was developed in which incorporates and launches to the simulation environment, the virtual camera described in Section 4.2.1 and at the same time the test piece described in Section 4.4. In Fig 38 you can see the simulation executed of these components.

Command line required to launch the experimental configuration:

```
$ roslaunch test_sample model.launch
```

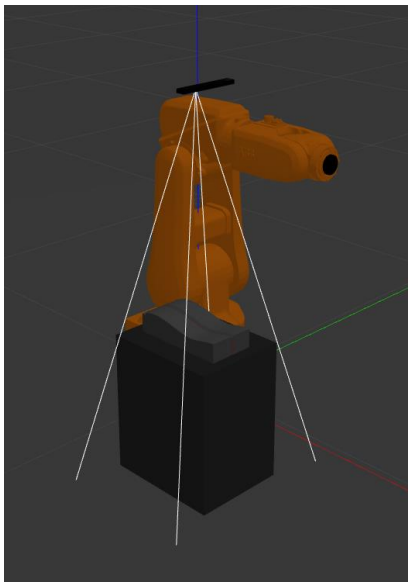
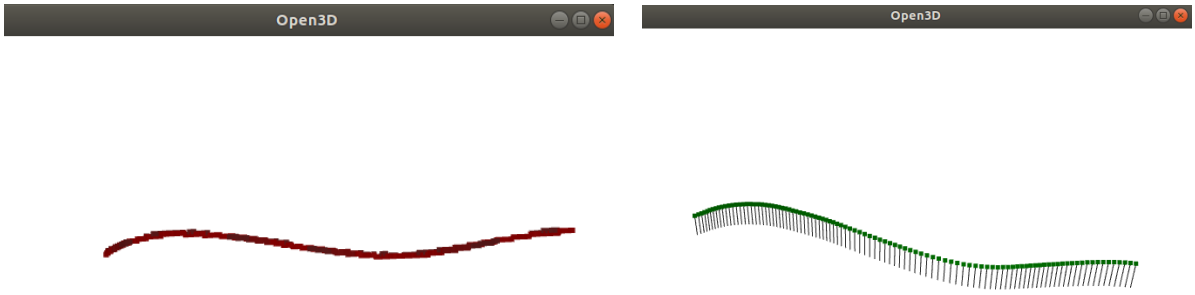


Fig 38 Simulation setup with the camera and workpiece in gazebo.

The experimentation of color segmentation and trajectory calculation was implemented, which yielded very similar results to those shown on the real camera. Fig 39a shows the point cloud corresponding to the target weld bead segmentation, and Fig 35Fig 39b shows the calculated points for the trajectory planning with the normal vector to each point with respect to the point cloud.



(a) (b)

Fig 39 Algorithm test. a) Weld bead color segmentation, b) Trajectory planning points.

In the present work the vector normal to each point in the point cloud was used to determine the robot pose, however during a real welding process, the robot must be able to adapt to the workpiece complexity and adjust the pose of the welding robot. When solving the robot pose, the weld bead pose must be considered as shown in Fig 40. Where $\vec{a}$ is the tangent vector to the trajectory and $\vec{n}$ is the vector normal to the workpiece surface [53].

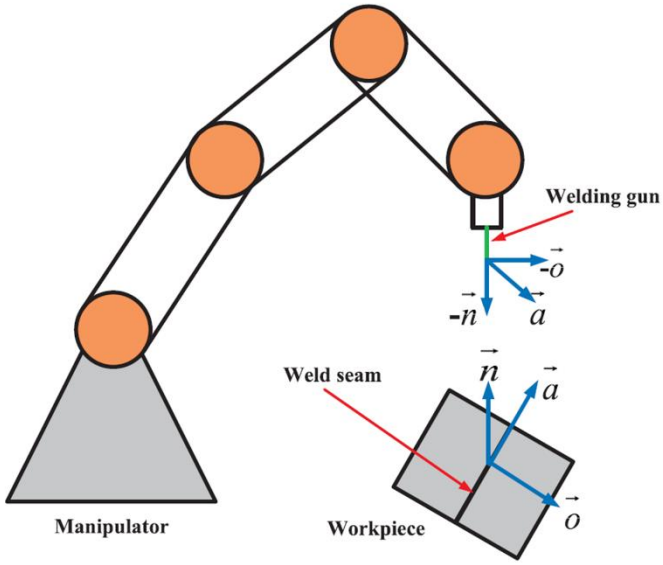


Fig 40 A schematic of seam pose and the pose welding gun.

5.5.2 Simulation of the trajectory of the points acquired by the camera with an ABB Robot.

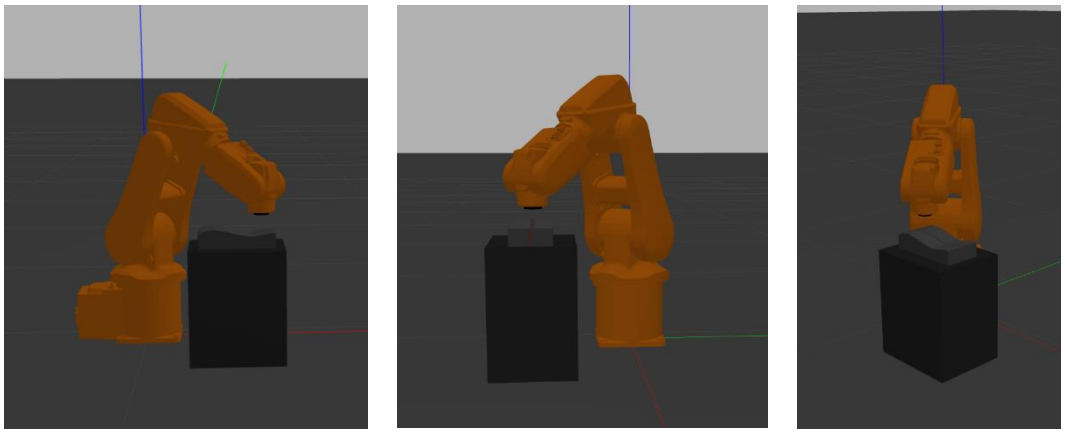
For the simulation of the path calculated in the previous section is necessary to run the ROS tools that run the control of a robot, fortunately the packages of ROS-Industrial already pre-configure the simulation environment, running the tools of Moveit!, Rviz and Gazebo.

It only remains to convert the path points to the robot's reference system, which is located in the position (0,0,0) according to TABLE 13. Thus, it is only necessary to apply the transformation matrix relative to the camera equation refer to 8.

$$\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0.25 \\ 0 & 1 & 0 & -0.25 \\ 0 & 0 & -1 & 0.8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

These R and T transformation matrices will give us the necessary parameters to interpolate the inverse kinematics of the robot and thus execute the trajectory planning. To achieve this, we used the tools provided by Movit! for which we defined a function within the python code that converts the transformation matrix of the points calculated by the algorithm of the previous section, to the final pose.

Fig 41 present a collection of images that show the robot's trajectory tracking over the workpiece.



(a) (b) (c)

Fig 41 Representative images of trajectory tracking from different perspectives.

Chapter 6 Conclusion

As a methodology for improve programming welding robot, this thesis proposes a 3D path extraction method based on color point cloud segmentation system. The major conclusions are generalized as follows:

- A welding robot sensor based on stereo vision and RGB sensor is implemented in this paper that could finish the 3D color reconstruction task of welding work piece, with a reconstruction standard deviation less than 1mm, which is a parameter comparable to that shown by Carfagni [42] for similar stereo devices.

- In order to achieve quick and robust weld path ex-traction, a color segmentation based on point cloud reconstruction perform the 3D path extraction task, with thresholds in RGB and HSV color space and an interpolation of the segmented points, the trajectory extraction results show errors close to or below 1.1 mm for V-type butt joint and below to 0.6 mm for a straight butt joint, comparable with other stereo vision work where Yang et al [20] show that the measurement resolution is less than 0.7 mm for V-type butt joint, and in contrast Zhou et al. [23] show a pose accuracy RMSE of 0.8 mm fora cylinder butt joint using a RealSense D415 sensor.

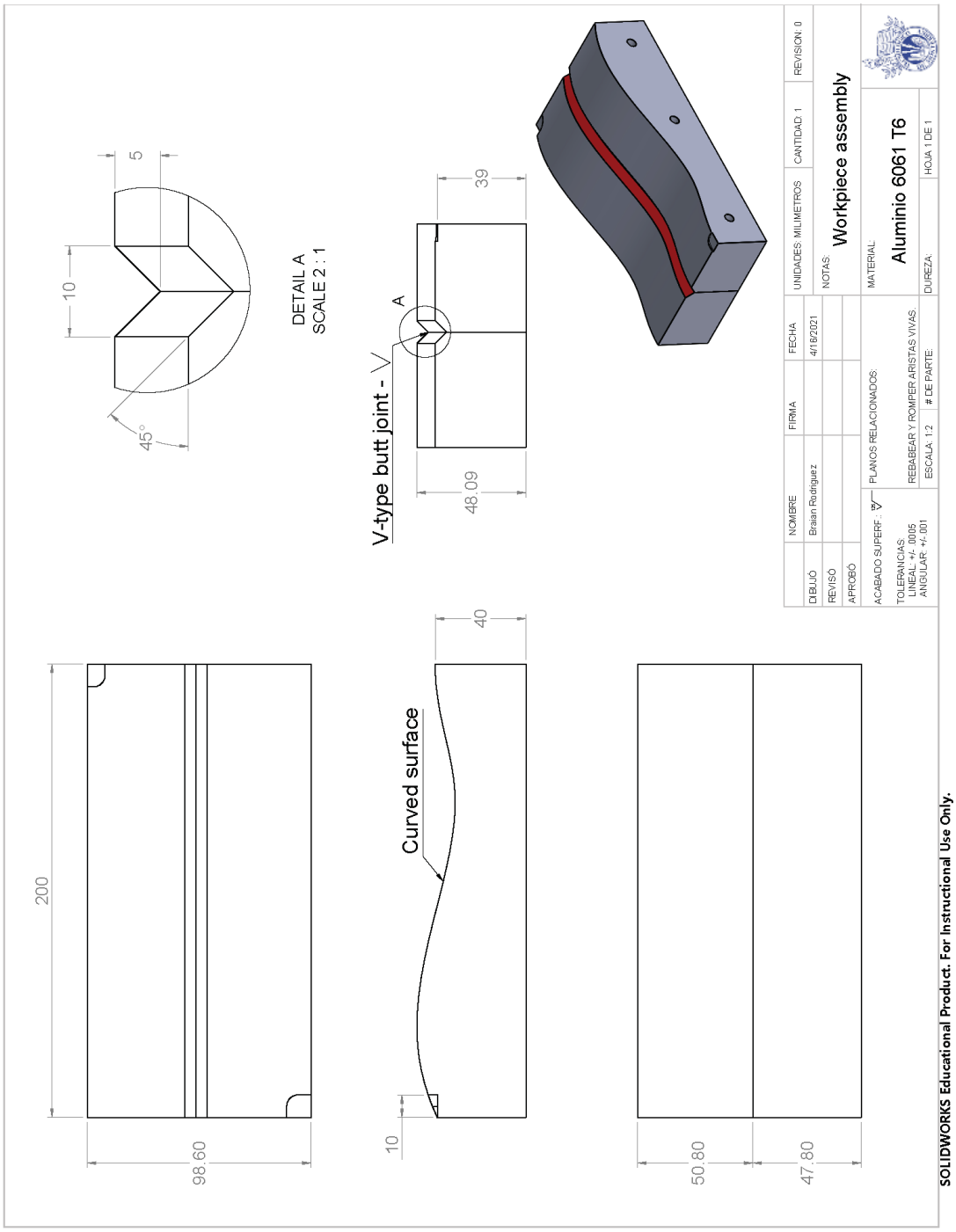
- Being illumination an important issue in image acquisition, the proposed algorithm was tested with dark field illumination and dome illumination systems, this latter providing the best results by having a homogeneous illumination over the surface and allowing color segmentation allowing a more accurate search in the color channels.

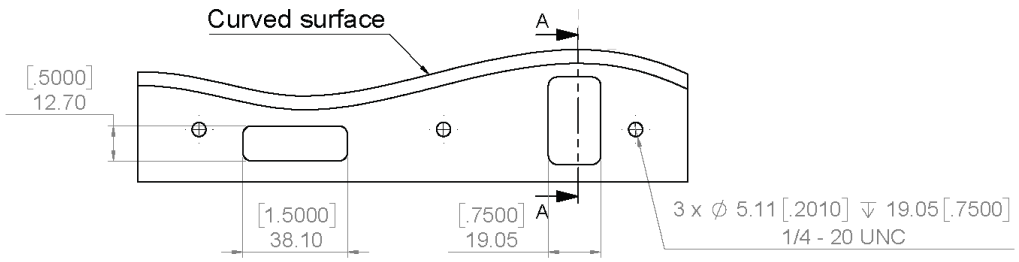
- The color segmentation algorithm was tested in RGB and HSV color space in order to verify its ability to calculate the trajectory on different materials and shapes. The HSV color space performs the best in the segmentation of five different colors over a dark surface.

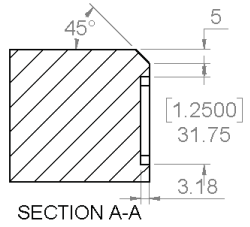
- In addition to the above, the adaptability of the proposed trajectory extraction system, being a global capture system, shows results that encourage experimentation not only in V-type welding one of the more study in literature, but also in other types of welding that would give a differential over most of the proposals found in the literature.

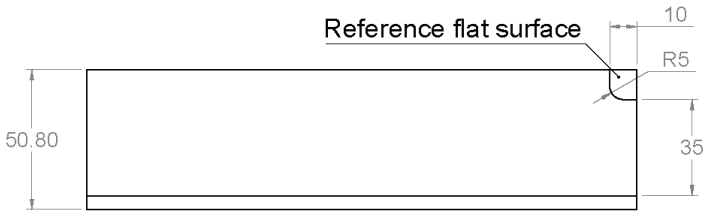
Appendix A

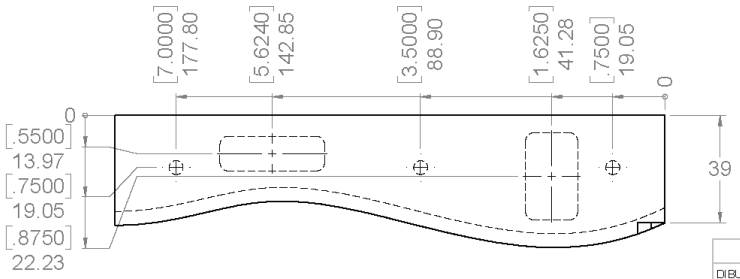
Workpiece drawings.
CAD designs and drawings can be found at https://drive.google.com/drive/folders/1gX444ZpC6TpMrBlFHZNVJMnO88kPwAfG?usp=sharing

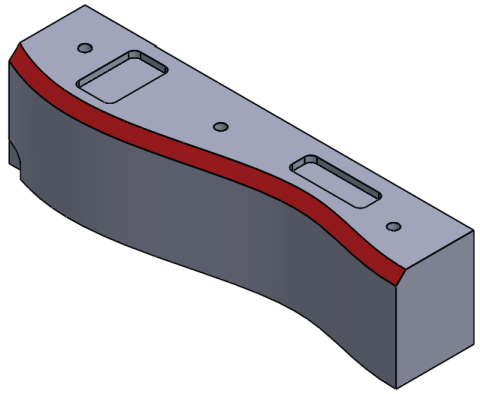




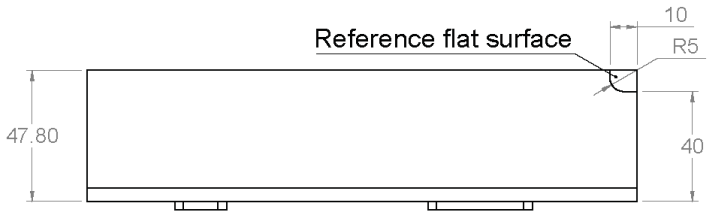


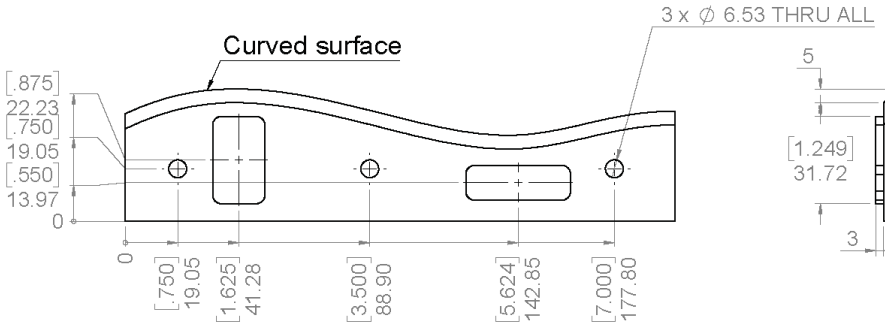


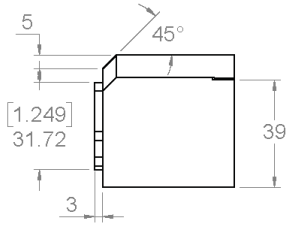


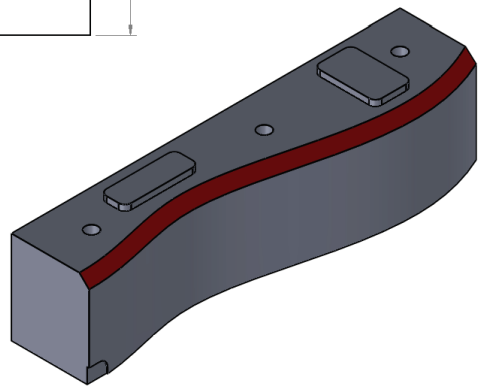


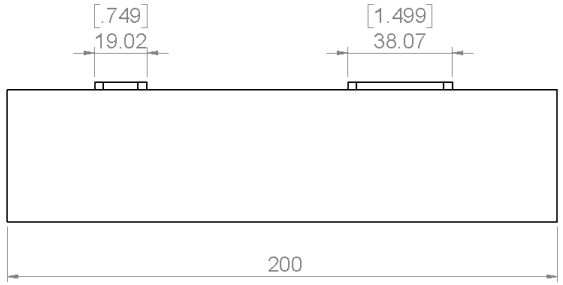
	NOMBRE	FIRMA	FECHA	UNIDADES: MILIMETROS	CANTIDAD: 1	REVISION: 0
DIBUJÓ	Braian Rodriguez		4/16/2021	NOTAS: Workpiece female section		
REVISÓ						
APROBÓ						
ACABADO SUPERF.:	PLANOS RELACIONADOS:			MATERIAL: Aluminium 6061 T6		
TOLERANCIAS: LINEAL: +/- .0005 ANGULAR: +/-.001	REBABEAR Y ROMPER ARISTAS VIVAS.					
	ESCALA: 1:2	# DE PARTE:		DUREZA:	HOJA 1 DE 1	











	NOMBRE	FIRMA	FECHA	UNIDADES: MILIMETROS	CANTIDAD: 1	REVISION: 0
DIBUJÓ	Braian Rodriguez		4/16/2021	NOTAS: Workpiece male section		
REVISÓ						
APROBÓ						
ACABADO SUPERF.:	PLANOS RELACIONADOS:			MATERIAL: Aluminium 6061 T6		
TOLERANCIAS: LINEAL: +/- .0005 ANGULAR: +/-.001	REBABEAR Y ROMPER ARISTAS VIVAS.					
	ESCALA: 1:2	# DE PARTE:		DUREZA:	HOJA 1 DE 1	

Appendix B

Main code.
Additional code from the tests performed can be found at:
https://drive.google.com/drive/folders/1wU2NTEx3-apYg5Xb7558ErhR12jjWLb4?usp=sharing

```
# Import Realsense library
import pyrealsense2 as rs
# Import Numpy for easy array manipulation
import numpy as np
# Import OpenCV for easy image rendering
import cv2
# Import 3D Point cloud manage library
import open3d as o3d
# Import libraries for iterpolate a smoth path
import copy
from scipy import interpolate

# Create a pipeline
pipeline = rs.pipeline()

#Create a config and configure the pipeline to stream
#  different resolutions of color and depth streams
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)

# Start streaming
profile = pipeline.start(config)

# Getting the depth sensor's depth scale
depth_sensor = profile.get_device().first_depth_sensor()
depth_sensor.set_option(rs.option.depth_units, 0.0005)
depth_scale = depth_sensor.get_depth_scale()
print("Depth Scale is: " , depth_scale)

# Removing the background of objects more than
#  clipping_distance_in_meters meters away
clipping_distance_in_meters = 0.29 #1 meter
clipping_distance = clipping_distance_in_meters / depth_scale

# Create an align object
# rs.align allows us to perform alignment of depth frames to others frames
# The "align_to" is the stream type to which we plan to align depth frames.
align_to = rs.stream.color
align = rs.align(align_to)

pc = rs.pointcloud()

def callback(x):
    pass

cv2.namedWindow('image')

# initial track bar limits
ilowH = 0
ihighH = 255
```

```
ilowS = 0
ihighS = 255

ilowV = 0
ihighV = 255

# create trackbars for color change
cv2.createTrackbar('low Hue','image',ilowH,255,callback)
cv2.createTrackbar('high Hue','image',ihighH,255,callback)

cv2.createTrackbar('low Saturation','image',ilowS,255,callback)
cv2.createTrackbar('high Saturation','image',ihighS,255,callback)

#cv2.createTrackbar('low Value','image',ilowV,255,callback)
#cv2.createTrackbar('high Value','image',ihighV,255,callback)

# Streaming loop
try:
    while True:
        # Get frameset of color and depth
        frames = pipeline.wait_for_frames()

        # Align the depth frame to color frame
        aligned_frames = align.process(frames)

        # Get aligned frames
        aligned_depth_frame = aligned_frames.get_depth_frame()
        color_frame = aligned_frames.get_color_frame()

        # Validate that both frames are valid
        if not aligned_depth_frame or not color_frame:
            continue

        depth_image = np.asanyarray(aligned_depth_frame.get_data())
        color_image = np.asanyarray(color_frame.get_data())

        # Remove background
        Black_color = 0
        depth_image_3d = np.dstack((depth_image,depth_image,depth_image))
        bg_removed = np.where((depth_image_3d > clipping_distance) |
                              (depth_image_3d <= 0), Black_color, color_image)

        # Render images
        hsv = cv2.cvtColor(bg_removed, cv2.COLOR_RGB2HSV)
        images = np.hstack((hsv, bg_removed))
        cv2.namedWindow('Align', cv2.WINDOW_AUTOSIZE)
        cv2.imshow('Align', images)

        ilowH = cv2.getTrackbarPos('lowH', 'image')
        ihighH = cv2.getTrackbarPos('highH', 'image')
        ilowS = cv2.getTrackbarPos('lowS', 'image')
        ihighS = cv2.getTrackbarPos('highS', 'image')
        #ilowV = cv2.getTrackbarPos('lowV', 'image')
        #ihighV = cv2.getTrackbarPos('highV', 'image')
        ilowV = 0
        ihighV = 255
        # Read frame
        lower_hsv = np.array([ilowH, ilowS, ilowV])
        higher_hsv = np.array([ihighH, ihighS, ihighV])
```

```
mask = cv2.inRange(hsv, lower_hsv, higher_hsv)
cv2.imshow('image', mask)

# Point cloud segmentation
pc.map_to(color_frame)
points = pc.calculate(aligned_depth_frame)

vertices = np.asanyarray(points.get_vertices(dims=3))
image_Points = np.reshape(vertices , (-1,3))
image_Colors = np.reshape(hsv , (-1,3))

# Hue color Seam Point Cloud
hu = image_Colors[:,0]
hue3 = np.reshape(np.dstack((hu,hu,hu)),(-1,3))

# Saturation color Seam Point Cloud
S = image_Colors[:,1]
SS3 = np.reshape(np.dstack((S,S,S)),(-1,3))

Col_Points1 = np.where(SS3 < ilowS, 0 , image_Points)
Col_Points = np.where((hue3 > ihighH) | (hue3 < ilowH), 0 , Col_Points1)
#Col_Points = np.where(hue3 > 180, 0 , Col_Points2)
CC_Points = Col_Points
CC_Points = CC_Points[~np.all(CC_Points == 0, axis=1)]

CC_Colors = image_Colors
CC_Colors = CC_Colors[~np.all(Col_Points == 0, axis=1)]

# Segmented Seam Point Cloud
pcd = o3d.geometry.PointCloud()
pcd.points = o3d.utility.Vector3dVector(CC_Points)
pcd.colors = o3d.utility.Vector3dVector(CC_Colors.astype(np.float) / 255.0)

# Surface Point Cloud
Surface_Points1 = np.where((SS3 > ilowS)|(SS3 == 0), 0 , image_Points)
Surface_Points = np.where((hue3 < ilowH), 0 , Surface_Points1)
RS_Points = Surface_Points
RS_Points = RS_Points[~np.all(RS_Points == 0, axis=1)]

RS_Colors = image_Colors
RS_Colors = RS_Colors[~np.all(Surface_Points == 0, axis=1)]

# Segmented Surface Point Cloud
pcd_S = o3d.geometry.PointCloud()
pcd_S.points = o3d.utility.Vector3dVector(RS_Points)
pcd_S.colors = o3d.utility.Vector3dVector(RS_Colors.astype(np.float) / 255.0)

# Path planining
source_fin = copy.deepcopy(pcd)
source_fin.paint_uniform_color([0, 1, 0])

s_points = np.asarray(source_fin.points)

# Cubic sline interpolation
num_true_pts = 100
tck, u = interpolate.splprep([s_points[:,0],s_points[:,1],s_points[:,2]], k=5,  s=0.0017)
x_knots, y_knots, z_knots = interpolate.splev(tck[0], tck)
u_fine = np.linspace(0,1,num_true_pts)
x_fine, y_fine, z_fine = interpolate.splev(u_fine, tck)
```

```
        # Path Point Cloud
        path = np.dstack((x_fine, y_fine, z_fine))
        path = np.reshape(path , (-1,3))

        path_pc = o3d.geometry.PointCloud()
        path_pc.points = o3d.utility.Vector3dVector(path)
        path_pc.paint_uniform_color([0, 1, 1])

        #o3d.visualization.draw_geometries([pcd])
        key = cv2.waitKey(1)
        if key == ord("v"):
            o3d.visualization.draw_geometries([pcd])
            o3d.visualization.draw_geometries([pcd_S])
            o3d.visualization.draw_geometries([path_pc])
        if key == ord("s"):
            print("Saving to Point Cloud to .ply...")
            o3d.io.write_point_cloud("3D_Seam.ply", pcd)
            o3d.io.write_point_cloud("3D_Surface.ply", pcd_S)
            o3d.io.write_point_cloud("Path.ply", path_pc)
            print("Done")
        if key & 0xFF == ord('q') or key == 27:
            cv2.destroyAllWindows()
            break

finally:
    pipeline.stop()
```

Appendix C

Main code.
Additional documentation of the required files needed to simulate the system described can be found at:
https://drive.google.com/drive/folders/1d0QRjxsIg1Gd_9ttIHs9lOxRykdeMECG?usp=sharing

```
#!/usr/bin/env python

# Import Numpy for easy array manipulation
import numpy as np
# Import OpenCV for easy image rendering
import cv2
# Import Open3D
import open3d as o3d

# import ROS stuff
import rospy
import time
from std_msgs.msg import Header
from sensor_msgs.msg import PointCloud2, PointField
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError
import sensor_msgs.point_cloud2 as pc2
import ros_numpy

#import sklearn
import math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy import interpolate

class PointCloud():
   def __init__(self):
      rospy.init_node("PointCloud")
      self.pc_cloud= rospy.Subscriber("/r200/camera/depth_registered/points",PointCloud2,self.cloud_callback)
      self.color_sub= rospy.Subscriber("/r200/camera/color/image_raw",Image,self.color_callback)
      self.bridge_object= CvBridge()
      #self.xyz_array = None
      #self.color_frame = None
      self.c=0
      self.d=0

      #To adjust the execution rate of the while Loop
      ros_rate = rospy.Rate(10) #10Hz
      # keep looping
      while not rospy.is_shutdown():
         #image_Points = np.reshape(self.xyz_array , (-1,3))
         if self.c==1 and self.d==1:
            cv2.imshow('Color', self.color_frame)
            #print(self.xyz_array)
            image_Points = self.xyz_array
            image_Colors = np.reshape(self.color_frame , (-1,3))

            # Red Seam Point Cloud
            Grey = 153
            clipping_distance = 0.49
```

```
depth_image_3d = image_Points[:,2]
depth_image_3d = np.reshape(np.dstack((depth_image_3d,depth_image_3d,depth_image_3d)),(-1,3))
bg_removed = np.where((depth_image_3d > clipping_distance) | (depth_image_3d <= 0), Grey, image_Colors)

# Red Seam Point Cloud
Set_r = 30

red = bg_removed[:,1]
red3 = np.reshape(np.dstack((red,red,red)),(-1,3))

Red_Points = np.where(red3 > Set_r, 0 , image_Points)
RR_Points = Red_Points
RR_Points = RR_Points[~np.all(RR_Points == 0, axis=1)]
#print(RR_Points)
#RR_Points = np.delete(RR_Points,np.where(~RR_Points.any(axis=1))[0], axis=0)
RR_Colors = bg_removed
RR_Colors = RR_Colors[~np.all(Red_Points == 0, axis=1)]
#RR_Colors = np.delete(RR_Colors,np.where(~Red_Points.any(axis=1))[0], axis=0)

pcd = o3d.geometry.PointCloud()
pcd.points = o3d.utility.Vector3dVector(RR_Points)
pcd.colors = o3d.utility.Vector3dVector(RR_Colors.astype(np.float) / 255.0)

s = RR_Points
ss = s[s[:,1].argsort()]

num_true_pts = 100
tck, u = interpolate.splprep([ss[:,0],ss[:,1],ss[:,2]], k=5,  s=0.00015)
x_knots, y_knots, z_knots = interpolate.splev(tck[0], tck)
u_fine = np.linspace(0,1,num_true_pts)
x_fine, y_fine, z_fine = interpolate.splev(u_fine, tck)

f = np.dstack((x_fine, y_fine, z_fine))
f = np.reshape(f , (-1,3))

ymin = min(y_fine)
ymax = max(y_fine)

tpcd = o3d.geometry.PointCloud()
tpcd.points = o3d.utility.Vector3dVector(f)
tpcd.paint_uniform_color([0, 1, 0])

#tpcd.estimate_normals()
tpcd.estimate_normals(search_param=o3d.geometry.KDTreeSearchParamHybrid(radius=0.00025, max_nn=30))
#print(np.asarray(pcd.normals)[:10, :])

xmin = min(x_fine) - 0.005
xmax = max(x_fine) + 0.005

fig1 = plt.figure(1)
ax3d = fig1.add_subplot(111, projection='3d')
plt.xlim(xmin, xmax)
#ax3d.plot(x_true, y_true, z_true, 'b')
#ax3d.plot(ss[:,0], ss[:,1], -ss[:,2], 'r*')
ax3d.plot(x_knots, y_knots, -z_knots, 'go')
ax3d.plot(x_fine, y_fine, -z_fine, 'g')
#fig1.show()
#plt.show()

#key = cv2.waitKey(1) & 0xFF
```

```
            key = cv2.waitKey(1)
            if key == ord("v"):
                o3d.visualization.draw_geometries([pcd])
                o3d.visualization.draw_geometries([tpcd])
            if key == ord("s"):
                o3d.io.write_point_cloud("Seam.ply", tpcd)
            if key == ord("p"):
                fig1.show()
                plt.show()
                break
            if key & 0xFF == ord('q') or key == 27:
                cv2.destroyAllWindows()
                break
            #if key == ord("v"):
            #    o3d.visualization.draw_geometries([pcd_S])
        ros_rate.sleep()


    def cloud_callback(self,cloud):
        self.xyz_array = ros_numpy.point_cloud2.pointcloud2_to_xyz_array(cloud)
        #print(self.xyz_array)
        self.d=1

    def color_callback(self,data):
        try:
            self.color_frame = self.bridge_object.imgmsg_to_cv2(data,desired_encoding="rgb8")
            self.c=1
        except CvBridgeError as e:
            print(e)

if __name__ == '__main__':
    PointCloud()
```

Bibliography

[1] J. Ogbemhe and K. Mpofu, "Towards achieving a fully intelligent robotic arc welding: A review," *Ind. Rob.*, vol. 42, no. 5, pp. 475–484, 2015.

[2] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Robotics and Computer-Integrated Manufacturing Recent progress on programming methods for industrial robots," *Robot. Comput. Integr. Manuf.*, vol. 28, no. 2, pp. 87–94, 2012.

[3] M. Kos, E. Arko, H. Kosler, and M. Jezeršek, "Remote laser welding with in-line adaptive 3D seam tracking," *Int. J. Adv. Manuf. Technol.*, pp. 4577–4586, 2019.

[4] L. Yang, E. Li, T. Long, J. Fan, and Z. Liang, "A Novel 3-D Path Extraction Method for Arc Welding Robot Based on Stereo Structured Light Sensor," *IEEE Sens. J.*, vol. 19, no. 2, pp. 763–773, 2019.

[5] Q. Zhao, X. Li, J. Lu, and J. Yi, "Monocular Vision-Based Parameter Estimation for Mobile Robotic Painting," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 10, pp. 3589–3599, 2019.

[6] P. Maiolino, R. Woolley, D. Branson, P. Benardos, A. Popov, and S. Ratchev, "Flexible robot sealant dispensing cell using RGB-D sensor and off-line programming," *Robot. Comput. Integr. Manuf.*, vol. 48, no. April, pp. 188–195, 2017.

[7] F. Q. Liu, Z. Y. Wang, and Y. Ji, "Precise initial weld position identification of a fillet weld seam using laser vision technology," *Int. J. Adv. Manuf. Technol.*, vol. 99, no. 5–8, pp. 2059–2068, 2018.

[8] L. Pérez, Í. Rodríguez, N. Rodríguez, R. Usamentiaga, and D. F. García, "Robot guidance using machine vision techniques in industrial environments: A comparative review," *Sensors (Switzerland)*, vol. 16, no. 3, 2016.

[9] P. Kiddee, Z. Fang, and M. Tan, "Visual recognition of the initial and end points of lap joint for welding robots," *2014 IEEE Int. Conf. Inf. Autom. ICIA 2014*, no. July, pp. 513–518, 2014.

[10] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[11] X. Z. Chen and S. B. Chen, "The autonomous detection and guiding of start welding position for arc welding robot," *Ind. Rob.*, vol. 37, no. 1, pp. 70–78, 2010.

[12] M. Dinham and G. Fang, "Autonomous weld seam identification and localisation using eye-in-hand stereo vision for robotic arc welding," *Robot. Comput. Integr. Manuf.*, vol. 29, no. 5, pp. 288–301, 2013.

[13] P. V. Hough, "Method and means for recognizing complex patterns," 1962.

[14] H. Ma, S. Wei, T. Lin, S. Chen, and L. Li, "Binocular vision system for both weld pool and root gap in robot welding process," *Sens. Rev.*, vol. 30, no. 2, pp. 116–123, 2010.

[15] A. Fernández Villán *et al.*, "Low-cost system for weld tracking based on artificial vision," *IEEE Trans. Ind. Appl.*, vol. 47, no. 3, pp. 1159–1167, 2011.

[16] D. Chang *et al.*, "A new seam-tracking algorithm through characteristic-point detection for a portable welding robot," *Robot. Comput. Integr. Manuf.*, vol. 28, no. 1, pp. 1–13, 2012.

[17] X. Li, X. Li, S. S. Ge, M. O. Khyam, and C. Luo, "Automatic Welding Seam Tracking and Identification," *IEEE Trans. Ind. Electron.*, vol. 64, no. 9, pp. 7261–7271, 2017.

[18] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Fluids Eng.*

Trans. ASME, vol. 82, no. 1, pp. 35–45, 1960.
[19] X. Li, X. Li, M. O. Khyam, and S. S. Ge, "Robust Welding Seam Tracking and Recognition," *IEEE Sens. J.*, vol. 17, no. 17, pp. 5609–5617, 2017.
[20] J. Zeng *et al.*, "A weld position recognition method based on directional and structured light information fusion in multi-layer/multi-passwelding," *Sensors (Switzerland)*, vol. 18, no. 1, 2018.
[21] J. Guo, Z. Zhu, B. Sun, and Y. Yu, "A novel multifunctional visual sensor based on combined laser structured lights and its anti-jamming detection algorithms," *Weld. World*, vol. 63, no. 2, pp. 313–322, 2019.
[22] L. Yang *et al.*, "A welding quality detection method for arc welding robot based on 3D reconstruction with SFS algorithm," *Int. J. Adv. Manuf. Technol.*, vol. 94, no. 1–4, pp. 1209–1220, 2018.
[23] R. Zhang, P. S. Tsai, J. E. Cryer, and M. Shah, "Shape from shading: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 8, pp. 690–706, 1999.
[24] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, 2015.
[25] L. Yang, Y. Liu, J. Peng, and Z. Liang, "A novel system for off-line 3D seam extraction and path planning based on point cloud segmentation for arc welding robot," *Robot. Comput. Integr. Manuf.*, vol. 64, no. September 2019, p. 101929, 2020.
[26] R. Xiao, Y. Xu, Z. Hou, C. Chen, and S. Chen, "An adaptive feature extraction algorithm for multiple typical seam tracking based on vision sensor in robotic arc welding," *Sensors Actuators A Phys.*, vol. 297, p. 111533, 2019.
[27] K. Zhang, M. Yan, T. Huang, J. Zheng, and Z. Li, "3D reconstruction of complex spatial weld seam for autonomous welding by laser structured light scanning," *J. Manuf. Process.*, vol. 39, no. December 2018, pp. 200–207, 2019.
[28] J. Fan, F. Jing, L. Yang, T. Long, and M. Tan, "A precise seam tracking method for narrow butt seams based on structured light vision sensor," *Opt. Laser Technol.*, vol. 109, no. 95, pp. 616–626, 2019.
[29] P. Zhou, R. Peng, M. Xu, V. W. Wu, and D. Navarro-Alarcon, "Path Planning with Automatic Seam Extraction over Point Cloud Models for Robotic Arc Welding," *IEEE Robot. Autom. Lett.*, 2021.
[30] Z. M. Bia and L. Wang, "Advances in 3D data acquisition and processing for industrial applications," *Robot. Comput. Integr. Manuf.*, vol. 26, no. 5, pp. 403–413, 2010.
[31] S. Foix, G. Alenyà, and C. Torras, "Lock-in time-of-flight (ToF) cameras: A survey," *IEEE Sens. J.*, vol. 11, no. 9, pp. 1917–1926, 2011.
[32] M. Z. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 993–1008, 2003.
[33] F. Ke, H. Liu, D. Zhao, G. Sun, W. Xu, and W. Feng, "A high precision image registration method for measurement based on the stereo camera system," *Optik (Stuttg).*, vol. 204, no. January, p. 164186, 2020.
[34] S. Zhang, "High-speed 3D shape measurement with structured light methods: A review," *Opt. Lasers Eng.*, vol. 106, no. March, pp. 119–131, 2018.

[35] J. Geng, "Structured-light 3D surface imaging: a tutorial," *Adv. Opt. Photonics*, vol. 3, no. 2, p. 128, 2011.
[36] R. Laganière, S. Gilbert, G. Roth, and S. Member, "Robust Object Pose Estimation From Feature-Based Stereo," vol. 55, no. 4, pp. 1270–1280, 2006.
[37] V. Suarez, "IMVR: ROBOT GUIADO POR VISIÓN EN TIEMPO REAL EN CELDA FLEXIBLE DE MANUFACTURA.," Instituto Tecnológico y de Estudios Superiores de Monterrey, 2009.
[38] X. Huang, J. Zhang, Q. Wu, L. Fan, and C. Yuan, "A Coarse-to-Fine Algorithm for Matching and Registration in 3D Cross-Source Point Clouds," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 10, pp. 2965–2977, 2018.
[39] J. Park, Q. Y. Zhou, and V. Koltun, "Colored Point Cloud Registration Revisited," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 143–152, 2017.
[40] A. L. Kleppe, L. Tingelstad, and O. Egeland, "Coarse Alignment for Model Fitting of Point Clouds Using a Curvature-Based Descriptor," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 811–824, 2019.
[41] M. S. Ahn, H. Chae, D. Noh, H. Nam, and D. Hong, "Analysis and Noise Modeling of the Intel RealSense D435 for Mobile Robots," *2019 16th Int. Conf. Ubiquitous Robot.*, pp. 707–711, 2019.
[42] M. Carfagni *et al.*, "Metrological and critical characterization of the intel D415 stereo depth camera," *Sensors (Switzerland)*, vol. 19, no. 3, 2019.
[43] Dassault Systèmes SolidWorks Corporation, "SolidWorks." [Online]. Available: https://www.solidworks.com/.
[44] I. Kuric, M. Košinár, and M. Císar, "Measurement and analysis of CNC machine tool accuracy in different location on work table," *Proc. Manuf. Syst.*, vol. 7, no. 4, 2012.
[45] Hexagon AB, "WorkNC," 2020. [Online]. Available: worknc.com.
[46] Intel Corporation, "Intel® RealSense™ SDK 2.0," 2018. [Online]. Available: https://github.com/IntelRealSense/librealsense.
[47] I. RealSense(TM), "Python Wrapper for Intel Realsense SDK 2.0," 2020. [Online]. Available: https://pypi.org/project/pyrealsense2/.
[48] OpenCV Team, "OpenCV," 2020. .
[49] N. Community, "NumPy Reference," *October*, vol. 1, no. October, pp. 1–1146, 2011.
[50] Q. Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv*, 2018.
[51] SciPy Community, "SciPy Reference Guide 0.17.1," p. 1229, 2013.
[52] M. de Graaf, R. Aarts, B. Jonker, and J. Meijer, "Real-time seam tracking for robotic laser welding using trajectory-based control," *Control Eng. Pract.*, vol. 18, no. 8, pp. 944–953, 2010.
[53] L. Yang and Y. Liu, "A Novel 3D Seam Extraction Method Based on Multi-Functional Sensor for V-Type Weld Seam," *IEEE Access*, vol. 7, pp. 182415–182424, 2019.
[54] M. Quigley, K. Conley, B. . Gerkey, J. Faust, T. Foote, and J. Wheeler, "ROS: an open-source Robot Operating System Morgan," *ICRA Work. Open Source Softw.*, 2019.
[55] D. Gossow, A. Leeper, D. Hershberger, and M. Ciocarlie, "ROS topics: Interactive markers: 3-D user interfaces for ROS applications," *IEEE Robot. Autom. Mag.*, vol. 18, no.

4, pp. 14–15, 2011.
[56] N. Koenig and A. Howard, “Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator,” in *IEEE International Conference on intelligent robots and systems*, 2004.
[57] J. Lentin and J. Cacace, *Mastering ROS for Robotics Programming - Second edition*, vol. 91. 2018.
[58] A. Martinez and E. Fernández, *Learning ROS for Robotics Programming*. 2013.
[59] D. Mronga and A. Aggarwal, “Motion Control of the Humanoid Robot AILA,” vol. 13–03, p. 24, 2013.
[60] K. Khokar, P. Beeson, and R. Burridge, “Implementation of KDL inverse kinematics routine on the atlas humanoid robot,” *Procedia Comput. Sci.*, vol. 46, no. Icict 2014, pp. 1441–1448, 2015.
[61] Intel Corporation, “ROS Wrapper for Intel® RealSense™ Devices,” 2018. [Online]. Available: https://github.com/IntelRealSense/realsense-ros.
[62] Intel Corporation, “realsense2_camera,” 2018. [Online]. Available: https://github.com/IntelRealSense/realsense-ros/blob/development/.travis.yml.
[63] S. Missri, “realsense_gazebo_plugin,” *GitHub Repos.*, 2020.
[64] S. Missri, “Intel RealSense Gazebo ROS plugin and model,” 2020. [Online]. Available: https://github.com/SyrianSpock/realsense_gazebo_plugin.
[65] ROS-Industrial Consortium, “ROS-Industrial,” 2020. [Online]. Available: https://rosindustrial.org/.
[66] ROS-Industrial repositories, “ABB Experimental.” [Online]. Available: https://github.com/ros-industrial/abb_experimental.
[67] R. K. Megalingam, D. Nagalla, R. K. Pasumarthi, V. Gontu, and P. K. Allada, “ROS based, simulation and control of a wheeled robot using gamer’s steering wheel,” *2018 4th Int. Conf. Comput. Commun. Autom. ICCCA 2018*, pp. 1–5, 2018.
[68] ROS core stacks, “Rospy.” [Online]. Available: https://github.com/ros/ros_comm.
[69] E. Wieser, “ros_numpy,” 2020. [Online]. Available: https://github.com/eric-wieser/ros_numpy.
[70] PickNik Robotics, “MoveIt!” [Online]. Available: https://moveit.ros.org/.
[71] J. Norrish, *Advanced welding processes*. Elsevier, 2006.
[72] K. Weman, *Welding processes handbook*. 2003.
[73] A. Rosenfeld and J. L. Pfaltz, “Sequential Operations in Digital Picture Processing,” *J. ACM*, vol. 13, no. 4, pp. 471–494, 1966.