1 2 9 0

UNIVERSIDADE Ð
COIMBRA

Tiago Matias Marques

DESENVOLVIMENTO DE UMA ESTRATÉGIA PARA DEFINIÇÃO DA ORIENTAÇÃO DE UMA FERRAMENTA DE DEPOSIÇÃO EM TAREFAS DE IMPRESSÃO

ALGORITMOS TENDO EM CONTA IMPRESSÃO DE METAIS

Dissertação no âmbito do Mestrado Integrado em Engenharia Mecânica, na área de Produção e Projeto, orientada pelo Professor Doutor Joaquim Norberto Cardoso Pires da Silva e apresentada ao Departamento de Engenharia Mecânica da Faculdade de Ciências e Tecnologias da Universidade de Coimbra.

Outubro de 2020

1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

Development of a Strategy for Defining the Orientation of a Deposition Tool in 3D Printing task– Algorithms Taking into Account Metal Printing

Submitted in Partial Fulfilment of the Requirements for the Master's Degree in Mechanical Engineering in The Speciality of Production and Project

Desenvolvimento de uma Estratégia para a Definição da Orientação de uma Ferramenta de Deposição em Tarefas de Impressão 3D– Algoritmos tendo em conta Impressão de Metais

Author
Tiago Matias Marques

Advisor
Professor Doutro Joaquim Norberto Cardoso Pires da Silva

Coimbra, October, 2020

“Engineering is the closest thing to magic that exists in the world.”

Elon Musk

ACKNOWLEDGEMENTS

Esta tese representa a conclusão de uma fase importante da minha vida, e como tal, quero deixar um agradecimento:

À minha família, por me apoiarem e me encorajarem a estudar.

Aos meus amigos, especialmente aqueles que viveram comigo, pelos bons momentos e por me ajudarem a superar os maus momentos.

Aos meus colegas do DEM, pelo apoio ao longo do curso.

Ao meu colega Francisco e colegas de laboratório, pela companhia e ajuda durante a realização deste trabalho.

E um agradecimento especial ao Professor Norberto Pires pela orientação e oportunidade de participar neste projeto.

Abstract

Thanks to technological developments, the Industrial Revolutions have led to an increase in the level of automation in companies through the implementation of new production methods and the introduction of robotized cells in production lines. However, some sectors, namely the metal construction sector, are still based on conventional processes that require the use of a high amount of human labor.

This thesis presents the contribution to the development of a 3D printing robotic cell, designed to automate the production of metallic parts for the construction sector, through the development of a strategy to define the orientation of a deposition tool coupled to the end of a robot.

In order to achieve this, a C# program was developed, with the capacity to analyze the geometry of a piece, that has been processed by a slicing software, and creates a file compatible with the robotic cell with the assignment of the printing tool's orientation parallel to the respective surface which leads to improvements in the printing process. These improvements can lead to faster printing time, less wasted material, and better superficial finish of the parts printed.

To verify the algorithm functioning, tests had to be performed in a virtual twin created in RobotStudio consisting of the same components on the physical system, namely an ABB IRB 4600 robotic arm, a linear IRBT 2006 track, and a TPS 400i CMT Advanced Fronius print tool.

Although the objectives defined during the development have been achieved by the right adjustment of the printing tool orientation according to the surface of the part, it was not possible to fully automate the strategy implemented, which leaves room for future improvements.

Keywords C#, Industrial Robot, RobotStudio, MAM, 3D printing.

Resumo

Graças aos desenvolvimentos tecnológicos, as Revoluções Industriais levaram a um aumento do nível de automação nas empresas através da implementação de novos métodos de produção e da introdução de células robotizadas nas linhas de produção. No entanto, alguns sectores, nomeadamente o sector da construção metálica, ainda se baseiam em processos convencionais que requerem a utilização de uma grande quantidade de mão-de-obra humana.

Esta tese apresenta a contribuição para o desenvolvimento de uma célula robotizada de impressão 3D, concebida para automatizar a produção de peças metálicas para o sector da construção, através do desenvolvimento de uma estratégia para definir a orientação de uma ferramenta de deposição acoplada ao fim de um robô.

Para esse objetivo, foi desenvolvido um programa em C#, com a capacidade de analisar a geometria de uma peça processada por um software de corte e cria um ficheiro compatível com a célula robótica com a atribuição da orientação da ferramenta de impressão paralela à respetiva superfície, o que conduz a um processo de impressão melhorado. Estas melhorias podem ser um tempo de impressão mais rápido, menos desperdício de material, e melhor acabamento superficial das peças impressas.

Para verificar o funcionamento do algoritmo, foi necessário realizar testes num virtual twin presente no RobotStudio que consiste nos mesmos componentes do sistema físico, nomeadamente um braço robótico ABB IRB 4600, uma pista linear IRBT 2006, e uma ferramenta de impressão TPS 400i CMT Advanced Fronius.

Embora os objetivos definidos durante o desenvolvimento tenham sido alcançados pelo ajuste correto da orientação da ferramenta de impressão de acordo com a superfície da peça, não foi possível automatizar totalmente a estratégia implementada, o que deixa espaço para futuras melhorias.

Palavras-chave: C#, Robô Industrial, RobotStudio, MAM, Impressão 3D.

Contents

LIST OF FIGURES

ACRONYMS/ ABBREVIATIONS

AM - Additive Manufacturing

AMC - Additive Manufacturing in steel Constructions

CMT - Cold Metal Transfer

CNC - Computer Numeric Control

DED - Directed Energy deposition

GUI - Graphical User Interface

IDE - Integrated Development Environment

M2M – Machine to Machine

MAM - Metallic Additive Manufacturing

OOP - Object Oriented Programming

PLC - Programmable Logic Controllers

RIA - Robotic Industries Associations

TCP - Tool Center Point

WAAM - Wire Arc Additive Manufacturing

1. INTRODUCTION

The first industrial revolution, at the end of the 18th century, created the industry with the allocation and concentration of producers in factories and the introduction of mechanized steam systems.

The second industrial revolution, in the following century, began mass production with the development of new technologies, the deployment of production lines, and the use of electricity.

The third industrial revolution, with the emergence of electronics, telecommunications, and information technologies, brought the automation of processes through the use of Programmable Logic Controllers (PLC) and robots.

The fourth revolution, the current one, is characterized by the fusion between the physical, digital, and biological world, the communication between machines, M2M, and the Internet of Things leading to self-control and communication between the various systems. (The Industrial Revolution: From Industry 1.0 to Industry 4.0 2019)

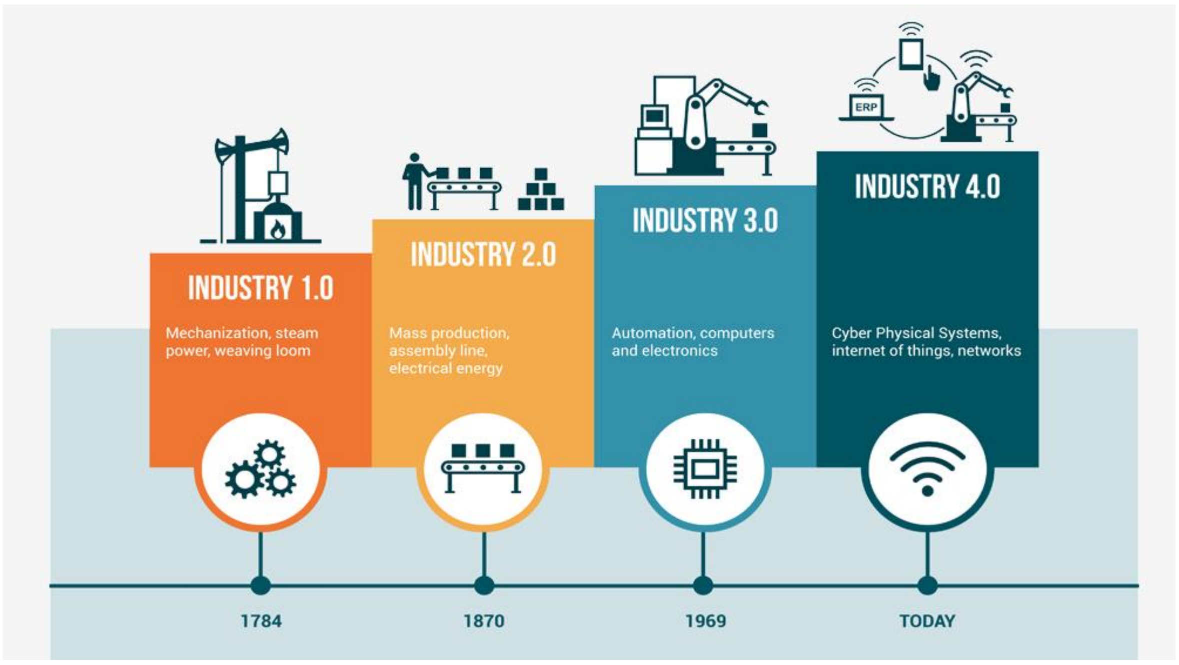

ERP
INDUSTRY 1.0
Mechanization, steam power, weaving loom
INDUSTRY 2.0
Mass production, assembly line, electrical energy
INDUSTRY 3.0
Automation, computers and electronics
INDUSTRY 4.0
Cyber Physical Systems, internet of things, networks
1784
1870
1969
TODAY

Figure 1- Industrial revolutions. (The Industrial Revolution: From Industry 1.0 to Industry 4.0 2019)

Although in the fourth industrial revolution, the metal construction sector lacks automation requiring a lot of human labor. The difficulty of automation in this sector is due to the structural uniqueness caused by the varied architectural and functional needs adjacent to each building/structure, and the high dimensions of the pieces.

Additive Manufacturing (AM), also known as 3d printing, becomes a promising technology for automating this industry thanks to its flexibility and implementation speed for the manufacture of single parts or small series. However, the current Metallic Additive Manufacturing (MAM) technologies are not compatible with this industry thanks to the limited size of the parts produced and the imperfections of the metallic pieces such as residual tensions or distortion. (Chalvin et al. 2020)

To overcome these problems, Professor Joaquim Norberto (Pires, Veiga, and Araújo 2009) of the University of Coimbra founded Additive Manufacturing in steel Constructions (AMC) to automate the process of printing large parts and globally improve the concept of MAM construction. For this, the development of a large print cell with the integration of a print tool to a 6-axis robotic arm using software enables the production of large parts with a single print instead of carrying out several operations using human labor.

The production of parts using additive manufacturing technologies follows a procedure that consists of modeling in drawing software and then running the model in slicing software. This software divides a 3D model into horizontal slices that correspond to the 3D printing layers and creates a file, using G-Code programming language, with the tool paths per layer that lead to the printing of the part and instructions for the tool.

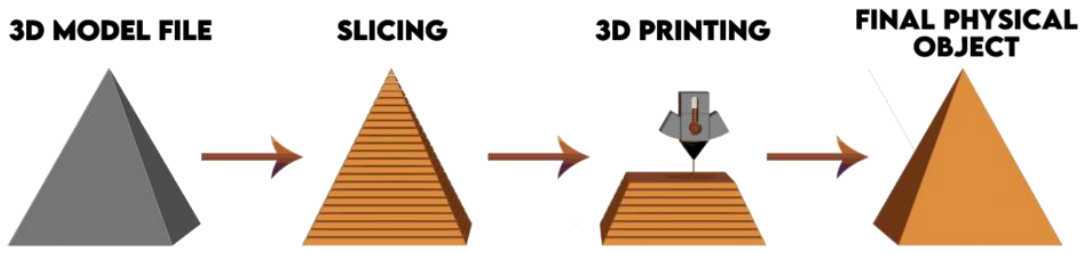


Figure 2- Steps to create 3d printed parts. (3D Printer Build)

This thesis presents the contribution to the AMC project through the development and simulation of a program that converts G-code files into a format compatible with the projected printing cell, being the main focus on automatically adjusting the tool orientation during printing, according to the surface of the part selected.

By implementing this tool orientation, improvement of the printing process is expected with the reduction/removal of support material and smooth surface finish. Both actions lead to a reduction in printing and surface machining time. Thus, the process becomes more efficient by minimizing the wasted material, printing, and surface finishing times. (DebRoy et al. 2018)(Wu et al. 2018)

1.1. Objectives

The main objective for this semester, as mentioned before, was the development of a program capable of converting the files with an automatic adjustment of the tool's orientation. To guarantee the fulfillment of the main objectives, the adjustment had to be implemented in the robot cell and tested. Since the robot cell was not operational, the tests had to be done in a virtual twin.

For this, there was a division of work. In general, my main focus was the development of the algorithm responsible for obtaining and manipulating the data while my colleague Francisco developed the area of reading the adjusted data and simulation in a digital system similar to the real one.

Although we made this division, it is crucial to mention that there was mutual contribution in both tasks, and in practice, we both worked on all objectives.

During the development, some small objectives were defined to guarantee the optimal performance of the program and the desired results. Among them the main ones:

- Extract relevant data from the G-Code
- Have an intuitive interface
- Represent the piece in the interface
- Calculate rotations that orientate the tool at each point parallel to the correspondent edge
- Apply orientation and movement by layer
- Parameterization (select if print infill, limit inclination, rotation by offset)
- Create a generic file
- Represent positions and respective orientations
- Simulate the behavior of the tool between positions with different orientations

1.2. Limitations

As mentioned earlier, the principal purpose is to automatically adjust the tool orientation according to the surface of the parts. That is applicable when printing the surface of the pieces, not for the inside. For this reason, the program needs to apply orientation only when printing surfaces, which in turn, the G-Code defines with perimeter points.

When reading a common G-Code, it is impossible to distinguish a point that belongs to the surface and a point that belongs to the inside.

Slic3r software, has an option that writes notes after the coordinates that indicate the type of point. For the program to get the data correctly from a G-Code file, this file must contain these comments, as this is the only way to distinguish the point types presented.

During the creation of G-Code for the program, the Slic3r's Version 1.3.0 Verbose option was always enabled.

It is important to note that this algorithm is limited with horizontal prints without material below. Since. it defines a vertical orientation that will not print well. The correct approach would be to development an algorithm in the slicing software that would intervene in these situations by changing the methodology used for printing.

1.3. Thesis Structure

The content of each chapter can be explained as follows:

- Chapter 1: Theme of this thesis, the impact on the project and the main objectives and limitations.
- Chapter 2: State of the art on the main themes applied in the project and previous systems / solutions.
- Chapter 3: Presentation of the tools and technologies used through the development of the program.
- Chapter 4: The program interface and explanation of its operation
- Chapter 5: Explanation on the functioning of the program
- Chapter 6: Problems encountered during development and solutions implemented
- Chapter 7: Experimental tests and analysis on the results obtained

- Chapter 8: Conclusions on the final results and future research

2. STATE OF THE ART

As previously mentioned, this chapter contains the state of the art of the technologies applied in this project being the main topics additive manufacturing, the industrial robot, and similar projects.

2.1. Additive Manufacturing (AM)

Additive manufacturing (AM) are processes that build three-dimensional (3D) parts by adding material layer-by-layer, guided by a digital model. This enables the production of complex geometries or customized parts directly from the design. (DebRoy et al. 2018)

Compared to the conventional approach, where parts are created using subtractive techniques, the AM process consumes less material and enables the production of a complete piece where traditional approach parts would need to be produced separately and then joined. (Silva, Assunção, and Almeida 2018)

But even with these advantages, AM parts are not common in the industry outside of prototyping applications. One reason for this is the economy of scale that focuses on the cost advantage that arises when a higher production rate is reached. The cost per part is a good representation of this, reflecting the division of total costs (which include tooling and installation cost) with all units produced. (Additive Manufacturing Delivers Economies of Scale AND Scope - 3DEO - Metal Additive Manufacturing 2017).

The figure below relates the cost per part with the manufactured units:

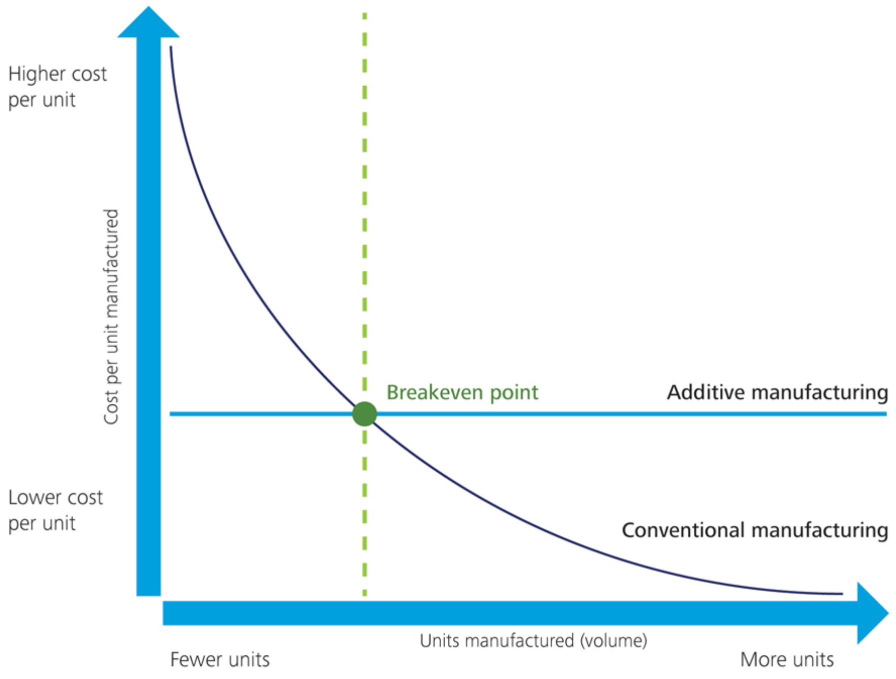


Figure 3- Comparison of cost variation per part with the number of parts manufactured for additive and conventional manufacturing. (Additive Manufacturing Delivers Economies of Scale AND Scope - 3DEO - Metal Additive Manufacturing 2017)

The cost per part is mainly composed by the initial machine costs and will have a low decline throughout the production because of the time it takes to print part. Conventional methods do not have this problem since the initial costs are compensated by the large scale of production achieved.

The economy of scope relates equipment cost to the ability to produce several different products. AM has the best scope economy compared to most conventional methods thanks to the versatility of the process since a printing system can make an infinite variety of parts. (Additive Manufacturing Delivers Economies of Scale AND Scope - 3DEO - Metal Additive Manufacturing 2017)

These properties make AM the indicated process for the production of unique complex pieces. One example of that is the manufacture of the Thales Alenia Space pressure vessel: by switching from the conventional subtractive machining method to AM, Cranfield University printed the 1m high, 8,5kg container with a saving of 200Kg of material and improved manufacturing time by 65% without giving up the required performances. (Titanium pressure vessel for space exploration made using 3D printing 2019)

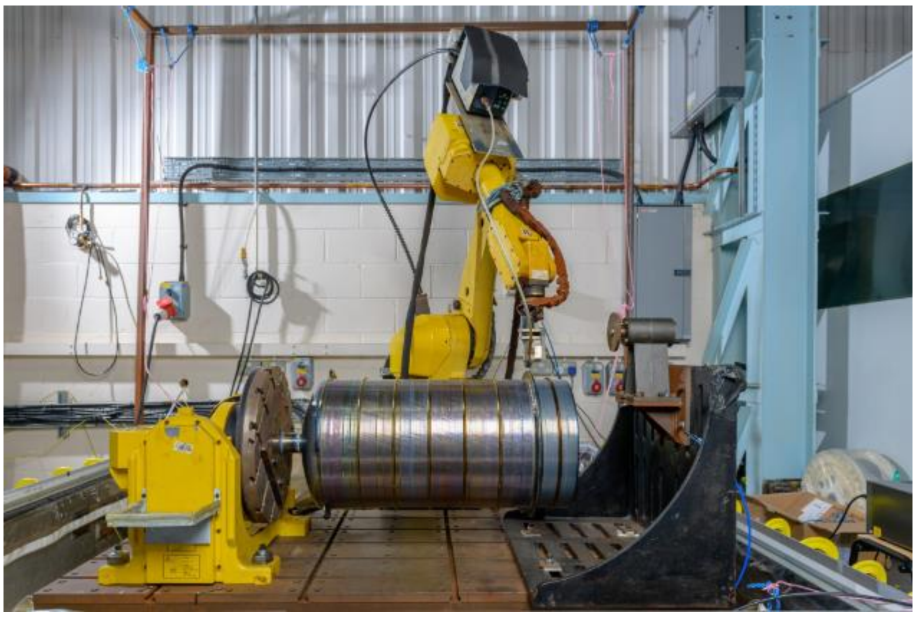


Figure 4- Manufacturing of Thales Alenia's pressure vessel. (Titanium pressure vessel for space exploration built successfully using the Wire + Arc Additive Manufacturing process 2017)

As technology evolves, new additive manufacturing methods are developed to improve certain aspects such as the final material properties of the parts, production, equipment cost and printing speed.

According to the ISO/ASTM 529200:2015, the additive manufacturing of metallic parts is divided by the following methods: (ISO/ASTM 52900:2015(en), Additive manufacturing — General principles — Terminology)

- Binder Jetting: liquid bonding agent is selectively deposited to join powder materials
- Directed Energy Deposition (DED): focused thermal energy is used to fuse materials by melting as they are being deposited
- Material Extrusion: dispense of material through a nozzle or orifice
- Powder Bed Fusion: use of thermal energy to fuse regions of a powder bed
- Sheet Lamination: a part is formed by bonding sheets of material

2.1.1. Wire and Arc Additive Manufacturing (WAAM)

The technology planned for the project was Wire and Arc Additive Manufacturing, a type DED, in which the addition of material is done in form of a wire and an arc welding process is used to melt both the wire and the substrate. (Iválntabernero et al. 2018)

The main advantages of this technology are: (Filomeno and Williams 2015)

- High deposition rate
- Much lower manufacturing cost than powder-bed processes
- Part size limited only by the motion system
- Possibility of in-process machining
- Possibility to work with several different metals

2.2. Industrial Robots

The Robotic Industries Association (RIA) defines industrial robots as programmable mechanical devices used to perform repetitive tasks with a high degree of accuracy. (Defining The Industrial Robot Industry and All It Entails | RIA - Robotics Online)

According to the World Robotics Report 2020 by the International Federation of Robotics, there are currently 2.7 million industrial robots operating in factories around the world. (IFR presents World Robotics Report 2020 - International Federation of Robotics 2020)

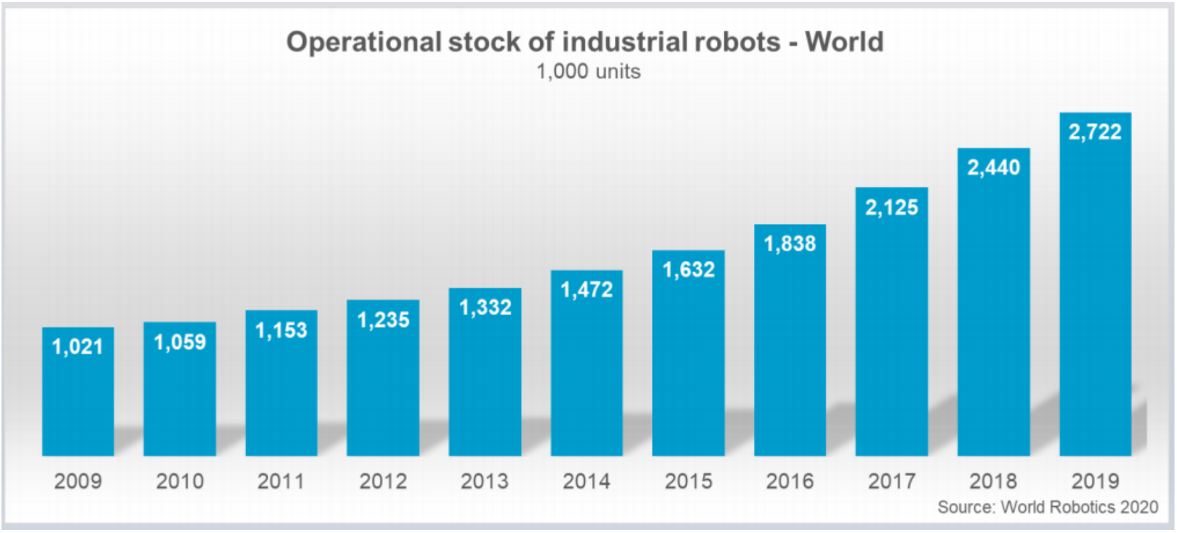


Figure 5- Number of industrial robots in operation since 2009. (IFR presents World Robotics Report 2020 - International Federation of Robotics n.d.)

An industrial robot is composed of a robot manipulator, power supply, and controllers.

A robot manipulator is a mechanism of multiple segments linked by joints with rotational or linear movement. The set of motions presented by each joint positions the robot arm in a specific position. A six-axis robot is a six-jointed robot that, in turn, is capable of positioning a tool mounted at the end, anywhere, and at any angle within its reach. This is called having six degrees of freedom. (Wilson 2014)

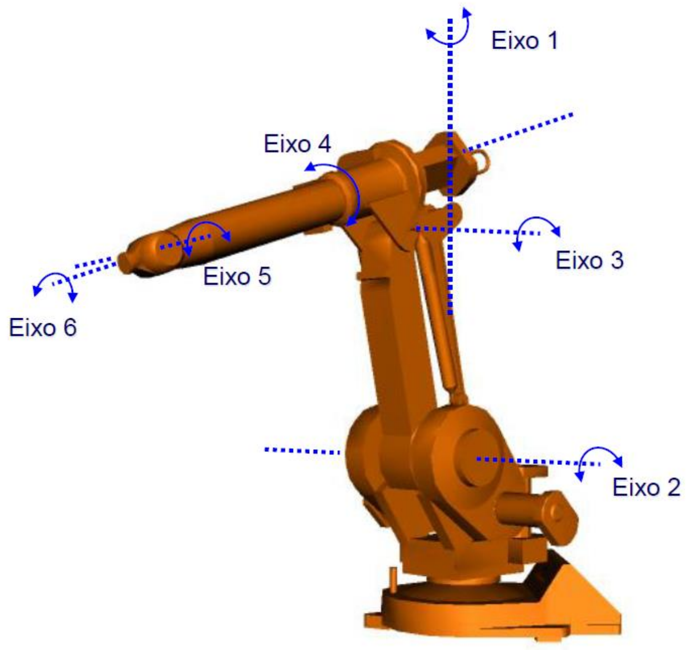


Figure 6- Axes representation on a model ABB RB 1400 M98.(J. Norberto Pires 2020)

2.2.1. Robot movement

The controller is the system responsible for the operation and monitoring of the industrial robot. It is usually a type of computer configured with information about the robot and the work environment, and with the capacity to store and execute the programs that operate the robot.

As explained above, what moves a robot arm to a position is the movement in the joints. To command the robot to perform this, the controller reads the information provided or stored and controls the movement of the joints. During movement, the controller also monitors the robot by receiving the data provided by the servo motors that composes the joints and sensors in the system.

ABB industrial robot controllers are programmed using RAPID language, which can also be applied to a Virtual Controller in the RobotStudio, allowing simulation of the robot behavior.

With RAPID, the command of the robot's movements is done using a function containing the defined destination with coordinates relative to the WorkObject origin established by the user.

2.2.2. Tool orientation

In robotics, a method used to define the orientation of the robot’s tool is Euler Angles. This method achieves a specific orientation by combining three rotations about a three-dimensional axis.

Changing the rotation order applied results in different orientations which, leads to the need to specify their order. In such a way, the Euler Angles are classified in the following ways:

- Tait-Bryan Angles: one rotation applied about each axis (for example, x-y-z or y-x-z)
- Proper Euler Angles: rotations are applied to only two axes (for example, x-y-x or z-y-z)

The reference to which the rotations are applied also results in different orientations, which led to two conventions: The extrinsic rotations uses the original axis as the reference at each rotation, while the intrinsic rotations uses the rotated axis as reference.

2.3. Orientation impact

As mentioned before, by automatically adjusting the tool orientation according to the surface of the parts, it is expected to reduce/remove the necessity for support material.

For understanding the reason for this, it is crucial to introduce the concept of overhang. In AM, geometric shapes in the 3D model that extend outside and beyond the previous layer are called overhangs. They can be defined by the angle and height of the overhang, as shown below: (Jiang et al. 2018)

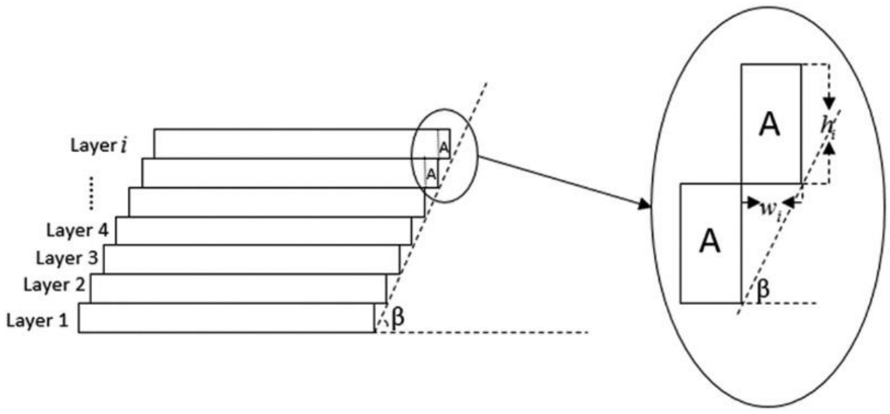


Figure 7- Overhang representation. (Jiang et al. 2018)

As represented in **Figure 7,** a small overhang angle (β) combined with a great height (h_i) means that some material (w_i) is not in contact with the layer below. Also, if there is too much material in the overhang, the part may collapse during printing. Usually, printing support material under the overhang solves this problem.

When it comes to parts made of polymeric materials, the removal of the supporting material is usually done easily. For metallic parts, machining is necessary, and by its nature, the process takes significant time. Also, a known problem with 3D printing is the step-shaped effect caused by the offset between layers, which increases the surface roughness.

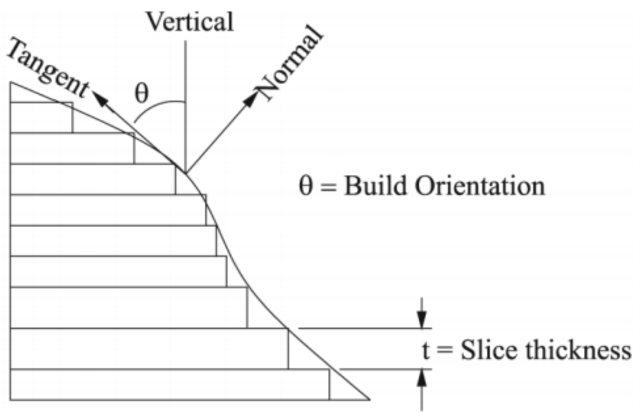


Figure 8- Staircase effect. (Mohan Pandey, Venkata Reddy, and Dhande 2003)

By orienting the print tool so that it is pointed to the printed layer below, represented as Tangent in **Figure 8**, it is possible to increase the contact of the overhang material in 2 successive layers, minimizing the need for support material in certain situations.

This not only reduces machining time but also reduces production cost with less material needed and a possible smoothing effect of the step shape, reducing surface roughness. All these improvements make the process faster and more sustainable.

However, the support material is required in cases where the structural integrity of the layered material is at risk, caused by the weight or geometry of the part.

In 3D printing for metal parts, it has been proven that the decrease in printing time by improving the printing tool speed is an incorrect approach due to the inconsistent properties of the welded bead. (DebRoy et al. 2018)

According to experimental tests where inclined steel walls were printed, using certain welding parameters, it was able to prove that Cold Metal Transfer (CMT), a type of WAAM, can operate in positional welding conditions in a range of angles from 90º to 15º. For a horizontal wall 4mm thick, positioning the tool collinear with the plate and printing 100mm long and positioning the torch with 30º offset from the horizontal plane printing 80mm obtain a diversion of the horizontal plane of 0.1-0.3º. This proves that wall type features could be printed with an orientation from 0º to 180º without the use of support structures. (Kazanas et al. 2012)

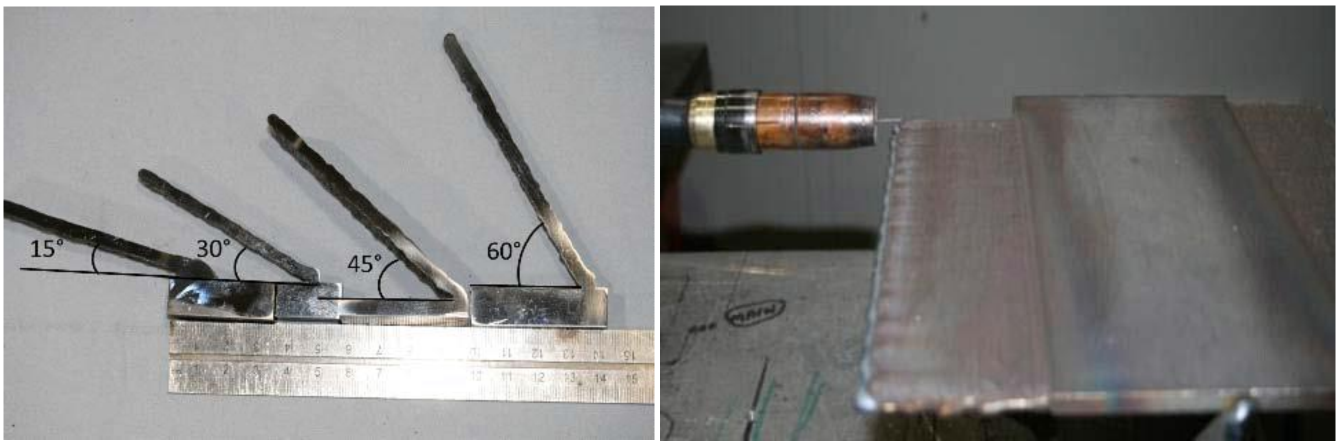


Figure 9- Inclined WAAM walls with different angles and horizontal wall. (Mehnen et al. 2014)

2.4. Previous projects

2.4.1. MX3D Bridge

In 2015, the company MX3D launched a pioneering project in 3D metal printing, proposing the printing of the first full-scale bridge on-site.

From the initiative to print on-site, MX3D had to find a solution that would allow the bridge to be printed, without a surface to support the weight of the part itself or enable the deposition of support material.

Using advanced slicing software, the positioning of some printing layers, perpendicular to each other would enable horizontal printing of certain sections. Thus, the deposition of the layers would be on the part itself, eliminating the problem of gravity, while in a normal situation, it would be necessary to use support materials.

By printing this bridge, it was possible to verify that changing the orientation of the printing tool can eliminate the need for support material, producing satisfactory results.

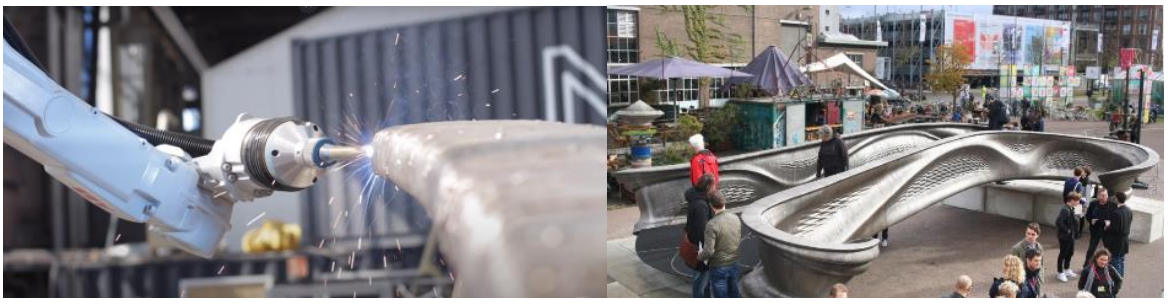


Figure 10- Horizontal print of MX3D Bridge. (MX3D Bridge | MX3D)

2.4.2. Takenaka Connector

Takenaka, one of Japan's largest architectural, engineering and construction companies, has designed a structural steel connector with the partnership of MX3D. This project shows the progress in the production of highly customized steel connectors, designed using 3D metallic printing on robots equipped with Wire Arc Additive Manufacturing (WAAM) technologies. The net weight of the steel connector is 40Kg, reaching 45Kg after filling with mortar.

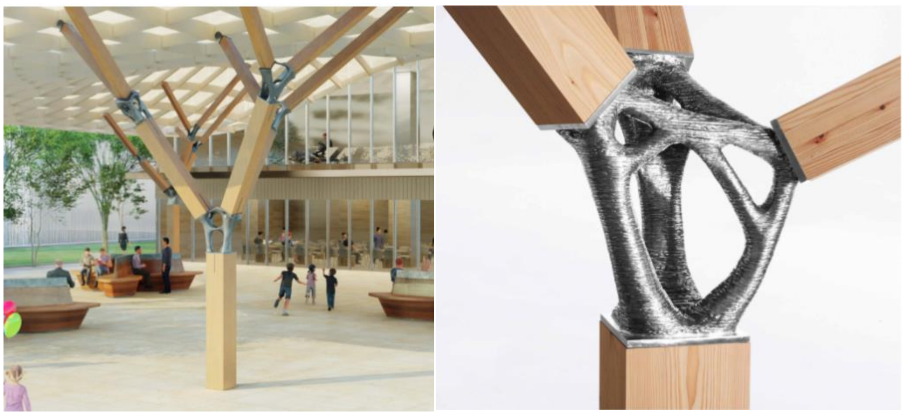


Figure 11- Takenaka Connector. (TAKENAKA CONNECTOR | MX3D)

3. TOOLS AND SOFTWARE

This chapter aims to make known the components present in the printing system, the software, and tools used in the development of the program, and in the different stages between the creation of a part, the simulation in a computer environment, and the subsequent printing in the system.

3.1. Microsoft Visual Studio, C#

Microsoft Visual Studio is an Integrated Development Environment (IDE), where tools to support software development with different programming languages are available.

The programming language used in this work was C # (C Sharp). This language, initially developed by Anders Hejlsberg, runs in the Microsoft .Net structure and is an Object-Oriented Programming (OOP) language. (C Sharp – Wikipédia, a enciclopédia livre n.d.)

The OOP is a computer programming model centered on objects that programmers want to manipulate rather than the logic of the program. This becomes appropriate for this program since the main purpose is to extract and process data stored in a file.

Using C# as the language in the development of the program, it was possible to create a graphical user interface (GUI) application, taking into account the objectives defined.

3.2. Inventor

Autodesk Inventor® is a computer-aided design software developed by the Autodesk company, used to modulate two-dimensional (2D) and three-dimensional (3D) parts of architectural or industrial projects.

For this project, the main use of this software was to model the parts with the predicted dimensions and to convert the models to STL format.

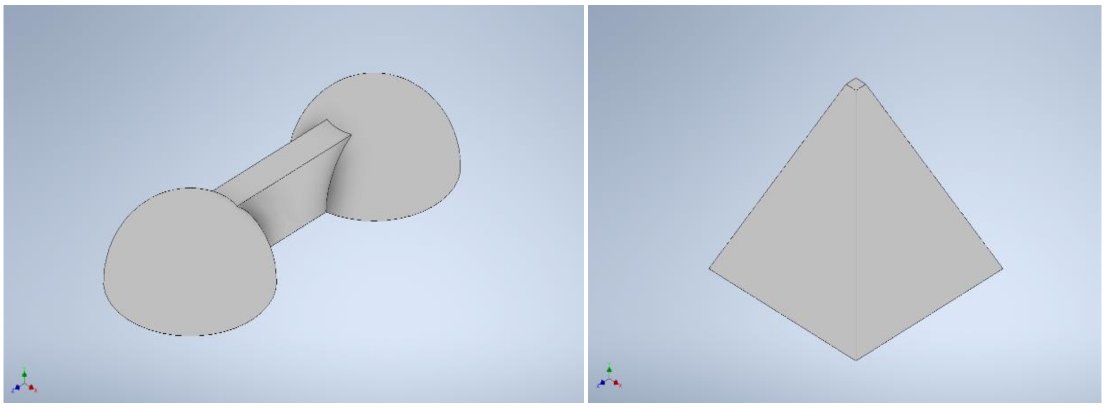


Figure 12- 3d model of dumbbell and pyramid model designed with Inventor.

3.3. Slic3r

Slic3r is a software that converts an STL file, corresponding to a 3D model, into a G-code file. To do this, the software splits a 3D model into horizontal slices that correspond to the 3D printing layers. After slicing, the software generates toolpaths that correspond to the material on each layer. This software also has the functionality to fill the inner gaps of the pieces with a pattern to increase their structural strength and generate support material to support parts that would collapse for being suspended or having no material underneath. (What Is Slic3r? – Simply Explained | All3DP n.d.)

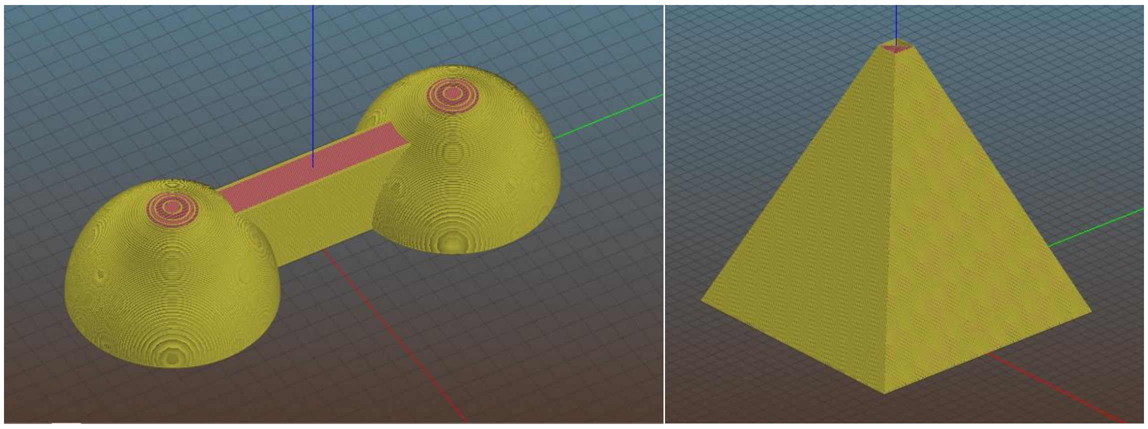


Figure 13- Representation of the layers of the dumbbell and pyramid parts on slic3r.

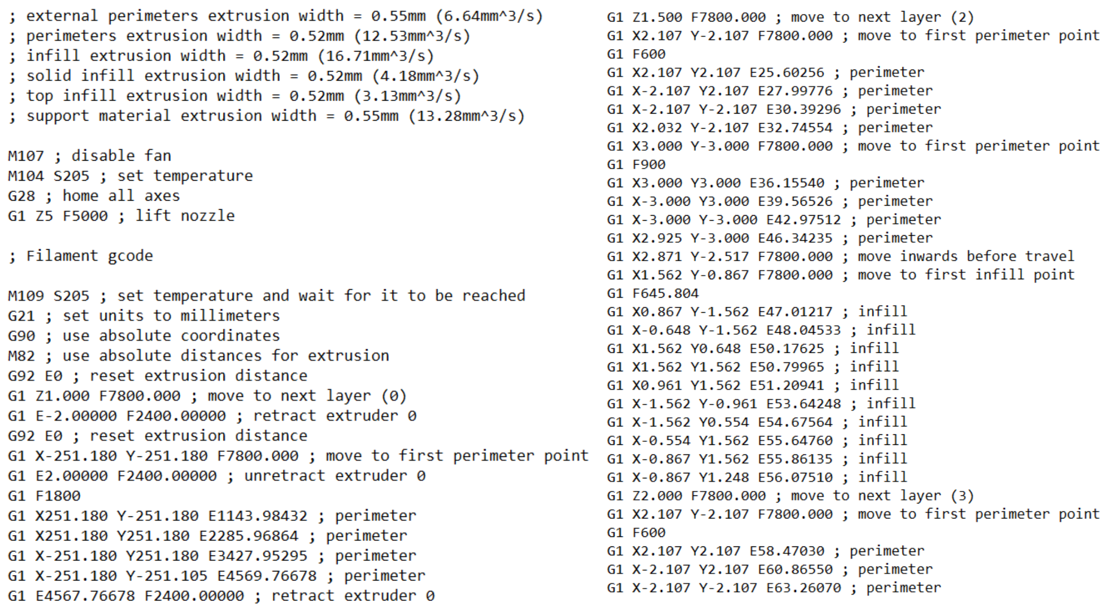
This software has the option to add comments to the G Code, which is necessary for the use of the algorithm since this is the way used to identify the points where the adjustment of orientation must be applied, the perimeter points.

3.3.1. G-Code

The G code is a programming language that originated from the need to create a standardized language in Computerized Numerical Command (CNC) systems. It consists of sequential lines of instructions composed of three-dimensional coordinates (x, y, z) combined with an existing command in the G commands to navigate and instruct the machine.

A simplified version is used in 3D printers since there is not such a wide range of tools and processes, which reduces the variety of commands available.

The figure below shows an example of a G-Code with notes enabled.



```
; external perimeters extrusion width = 0.55mm (6.64mm^3/s)
; perimeters extrusion width = 0.52mm (12.53mm^3/s)
; infill extrusion width = 0.52mm (16.71mm^3/s)
; solid infill extrusion width = 0.52mm (4.18mm^3/s)
; top infill extrusion width = 0.52mm (3.13mm^3/s)
; support material extrusion width = 0.55mm (13.28mm^3/s)

M107 ; disable fan
M104 S205 ; set temperature
G28 ; home all axes
G1 Z5 F5000 ; lift nozzle

; Filament gcode

M109 S205 ; set temperature and wait for it to be reached
G21 ; set units to millimeters
G90 ; use absolute coordinates
M82 ; use absolute distances for extrusion
G92 E0 ; reset extrusion distance
G1 Z1.000 F7800.000 ; move to next layer (0)
G1 E-2.00000 F2400.00000 ; retract extruder 0
G92 E0 ; reset extrusion distance
G1 X-251.180 Y-251.180 F7800.000 ; move to first perimeter point
G1 E2.00000 F2400.00000 ; unretract extruder 0
G1 F1800
G1 X251.180 Y-251.180 E1143.98432 ; perimeter
G1 X251.180 Y251.180 E2285.96864 ; perimeter
G1 X-251.180 Y251.180 E3427.95295 ; perimeter
G1 X-251.180 Y-251.105 E4569.76678 ; perimeter
G1 E4567.76678 F2400.00000 ; retract extruder 0
```

```
G1 Z1.500 F7800.000 ; move to next layer (2)
G1 X2.107 Y-2.107 F7800.000 ; move to first perimeter point
G1 F600
G1 X2.107 Y2.107 E25.60256 ; perimeter
G1 X-2.107 Y2.107 E27.99776 ; perimeter
G1 X-2.107 Y-2.107 E30.39296 ; perimeter
G1 X2.032 Y-2.107 E32.74554 ; perimeter
G1 X3.000 Y-3.000 F7800.000 ; move to first perimeter point
G1 F900
G1 X3.000 Y3.000 E36.15540 ; perimeter
G1 X-3.000 Y3.000 E39.56526 ; perimeter
G1 X-3.000 Y-3.000 E42.97512 ; perimeter
G1 X2.925 Y-3.000 E46.34235 ; perimeter
G1 X2.871 Y-2.517 F7800.000 ; move inwards before travel
G1 X1.562 Y-0.867 F7800.000 ; move to first infill point
G1 F645.804
G1 X0.867 Y-1.562 E47.01217 ; infill
G1 X-0.648 Y-1.562 E48.04533 ; infill
G1 X1.562 Y0.648 E50.17625 ; infill
G1 X1.562 Y1.562 E50.79965 ; infill
G1 X0.961 Y1.562 E51.20941 ; infill
G1 X-1.562 Y-0.961 E53.64248 ; infill
G1 X-1.562 Y0.554 E54.67564 ; infill
G1 X-0.554 Y1.562 E55.64760 ; infill
G1 X-0.867 Y1.562 E55.86135 ; infill
G1 X-0.867 Y1.248 E56.07510 ; infill
G1 Z2.000 F7800.000 ; move to next layer (3)
G1 X2.107 Y-2.107 F7800.000 ; move to first perimeter point
G1 F600
G1 X2.107 Y2.107 E58.47030 ; perimeter
G1 X-2.107 Y2.107 E60.86550 ; perimeter
G1 X-2.107 Y-2.107 E63.26070 ; perimeter
```

Figure 14- G-Code excerpts of a test part.

Inside the G-Code exist five types of points:

- Perimeter correspond to the points which make the piece’s surface
- Infill are the points used to fill the interior gaps of a piece
- Support are not part of the piece and are used to avoid the collapse of suspended parts or without material underneath
- Skirt is part of a contour outside the piece used to establish a smooth flow of filament through the tool
- Brims are used to hold the edges of the piece preventing it from warping and helping with bed adhesion

3.4. RobotStudio

RobotStudio is ABB's offline simulation and programming software that allows the creation of a digital twin, a computational model in a 3D virtual environment that provides the tools to perform tasks such as training, programming, and optimization (RobotStudio - ABB Robotics n.d.).

Using the ABB Virtual Controller in RobotStudio, it is possible to create a digital twin, a digital replica of the physical system, and use an exact copy of the real software running on the system to test the implementation of the adjustments to the orientations. This allows to quickly perform very realistic simulations of the movements using real robot programs and configuration files identical to those used in the system without the risk of damaging the equipment.

For this, it is necessary to configure the digital twin with the identical components projected for the physical system and allow the reading and implementation of coordinates and instructions.

3.5. Printing Cell

The main components selected for the project are an ABB robotic arm IRB 4600, an ABB linear track IRBT 2005, and a printing tool TPS 400i CMT Advanced Fronius.

A photo of the partial assembly of the robotic cell is shown below:

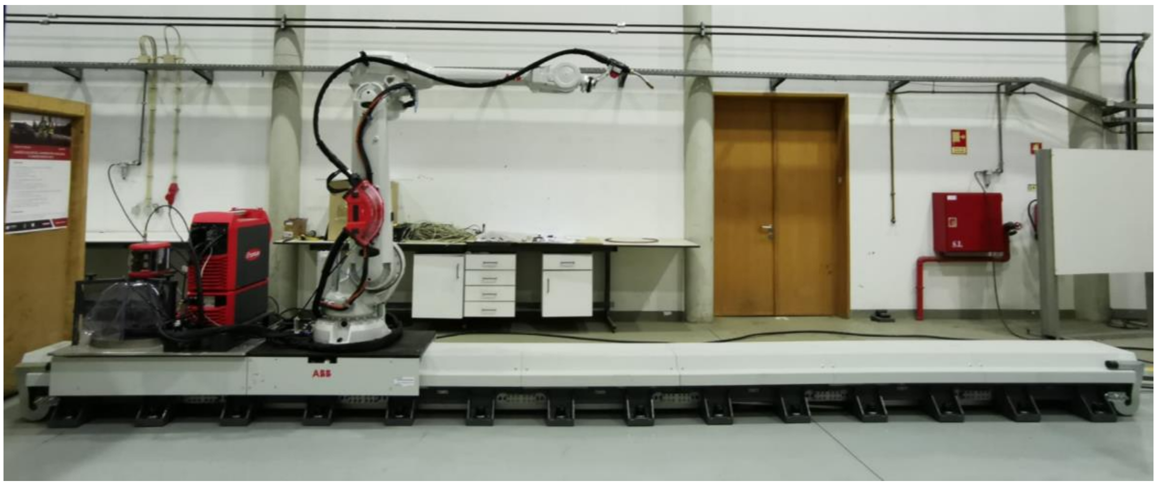


Figure 15- Robot cell assembly.

As mentioned above, the tests will be performed in a double digital cell, similar to the robotic cell, developed in RobotStudio. A photo of the digital twin created for testing is found below:

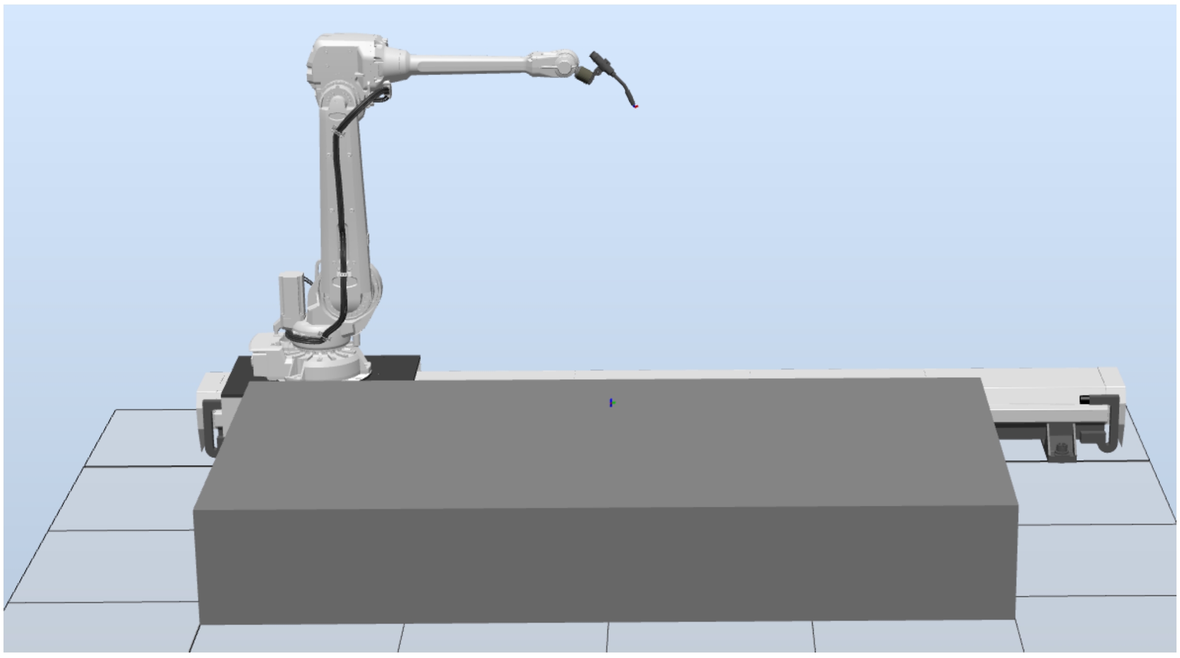
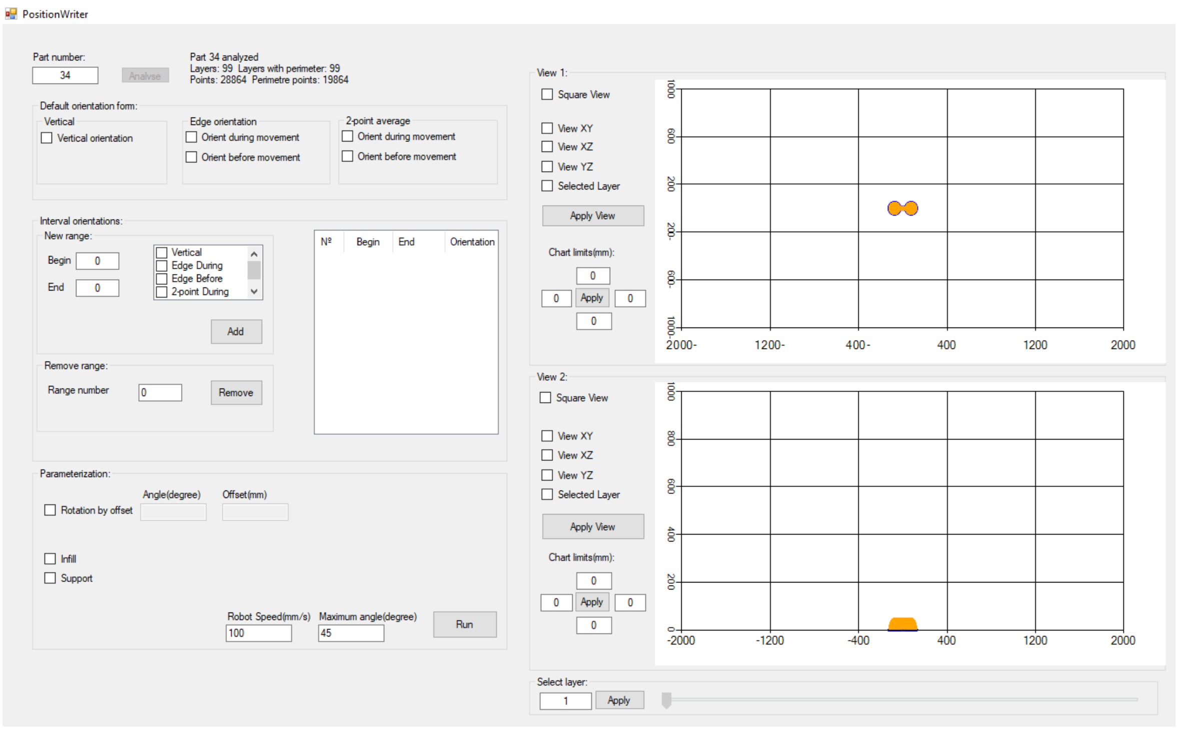


Figure 16- Virtual twin.

4. INTERFACE

This chapter presents the structure of the program interface in C# and the functionalities present in it that configure the file obtained for printing within the robotic cell.

The program interface is divided with the left side constituting the printing functionalities while the right side presents the representations of the piece to be processed. The part number is used to select the piece since, during development and testing, all test files were stored and organized in a folder.


PositionWriter

Part number: 34

Part 34 analyzed
Layers: 99 Layers with perimeter: 99
Points: 28864 Perimetre points: 19864

Robot Speed(mm/s): 100
Maximum angle(degree): 45

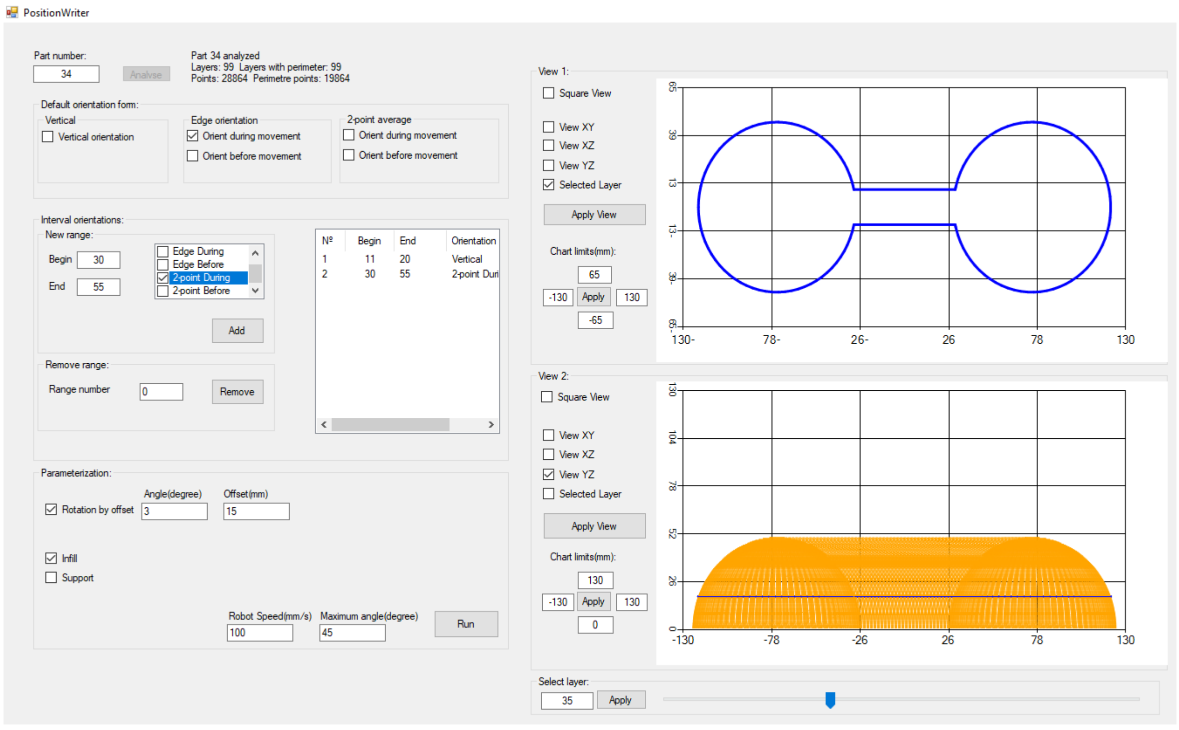
Select layer: 1

Figure 17- Interface after dumbbell file analyses.

When the part is selected and analyzed, a text shows the number of layers, how many of them have perimeter points since, when using Slic3r, there may be layers with only supporting points, the total number of points and perimeter points. Once the analysis is completed, the rest of the program's features are available.

By default, after the file analysis, the top view of the part is shown, the graphic limits being the table dimensions, and the front view, the horizontal axis being the table length, and the vertical axis 1m high. Using these limits, it is possible to have a minimum perception of the size of the part to be printed with the table.

The representations of the piece appeared as an aid in the selection of orientations layer by layer by making it possible to select a layer, view the tool path that will be executed in a window and another view the correspondent horizontal position as shown below (larger version in **Appendix A**):



PositionWriter

Part number: 34

Part 34 analyzed
Layers: 99 Layers with perimeter: 99
Points: 28864 Perimetre points: 19864

Default orientation form:
- Vertical: ☐ Vertical orientation
- Edge orientation: ☑ Orient during movement; ☐ Orient before movement
- 2-point average: ☐ Orient during movement; ☐ Orient before movement

Interval orientations:

New range: Begin 30, End 55; ☐ Edge During, ☐ Edge Before, ☑ 2-point During, ☐ 2-point Before

Remove range: Range number 0

Nº	Begin	End	Orientation
1	11	20	Vertical
2	30	55	2-point Duri

Parameterization:
- ☑ Rotation by offset: Angle(degree) 3, Offset(mm) 15
- ☑ Infill
- ☐ Support
- Robot Speed(mm/s): 100
- Maximum angle(degree): 45

View 1: ☐ Square View; ☐ View XY; ☐ View XZ; ☐ View YZ; ☑ Selected Layer
Chart limits(mm): 65, -130, 130, -65

View 2: ☐ Square View; ☐ View XY; ☐ View XZ; ☑ View YZ; ☐ Selected Layer
Chart limits(mm): 130, -130, 130, 0

Select layer: 35

Figure 18- Fully configured interface for the dumbbell.

The default orientation form is where the default method of orientation and movement is chosen for the tool, while in the Interval orientations section are the tools to add or remove intervals with different methods and view the ones already created. Parameterization includes the option to print infill and support points, apply a limit of rotation by offset (Explained in **Chapter 6**) and define the tool speed and maximum angle.

5. PROGRAM

This chapter explains the operating methodology of the code present in the program. For this purpose, the following division is made according to the main processes:

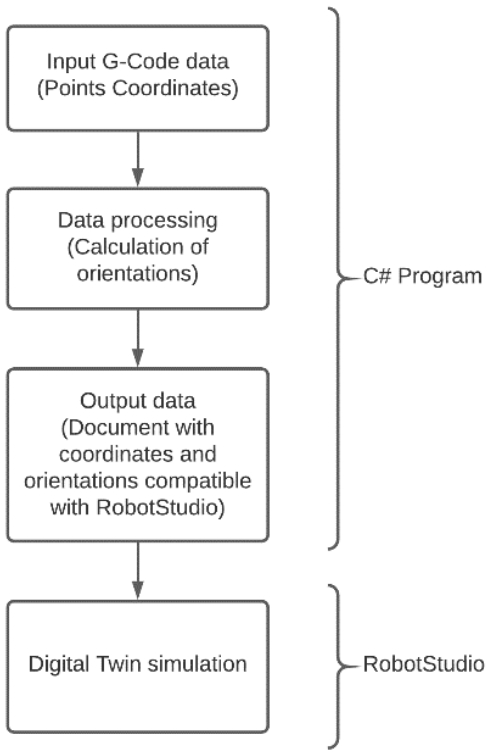

Input G-Code data (Points Coordinates)
Data processing (Calculation of orientations)
Output data (Document with coordinates and orientations compatible with RobotStudio)
Digital Twin simulation
C# Program
RobotStudio

Figure 19- Programs main processes.

This thesis only explains in detail the processes in the C# program, since that was the part which was focused on. The processes of RobotStudio are presented in detail in my colleague Francisco thesis.

5.1. G-Code analyses

The first part of the program analyzes the file, in G-Code format, selected by the user. It reads the file line by line to identify, extract and organize the relevant data present. The points being the main data, are identified by type (Perimeter, Infill, Support, Skirt or Brim), layer and position as well as situations in which the 3D printer retracts or unretract (unretract is the term defined in the Slic3r notes for advancing the tool).

The data is organized as follows:

Dictionary [Sortedlist []] [List (Point)]

A dictionary stores key-value pairs with the keys having to be unique and not null/empty, while the values can be lists of data. This makes it possible to create three dictionaries, one corresponding to each type of point, allow access to the list of points of a layer using its corresponding height as key. Skirt and Brim points are saved in an independent list. An unretract and a retract lists store the coordinates where the tool respectively stops or starts printing.

To ensure a good performance of the program, by allowing access to all points in the correct order, all heights of layers present in the G-Code, were stored in a Sortedlist. This allows to cycle through all the values in the Sortedlist and always check if the dictionaries have pairs using that value as a key.

In turn, points are a type of structure value composed of X, Y, Z coordinates, and a perimeter number. This perimeter number became a necessary adaptation as explained in C**hapter 6.**

To visualize and better understand this process, the flowchart present in **Appendix B** shows the sequential procedure of the program for the collection and cataloging of data.

A MATLAB program developed by my colleague Francisco made it possible to verify that all perimeter points were obtained with the correct coordinates.

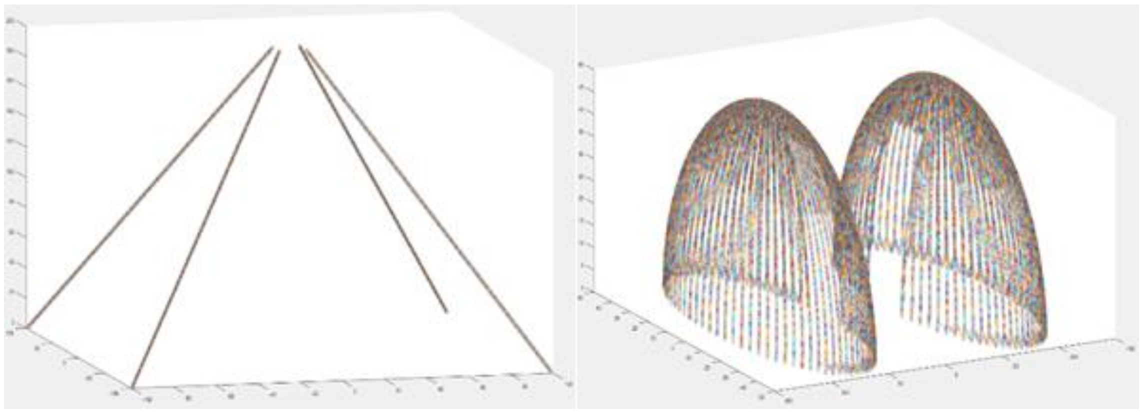


Figure 20- MATLAB representation of data collected for the pyramid and the dumbbell.

5.1.1. Representations

The representations of the piece in the C# interface are made by accessing the data stored in the perimeter dictionary and, using the graphic tool available in the GUI, draw lines to join consecutive points.

What allows changing views are the coordinates of the points used for the representation. For example, in the front view of the part (robot view), the Y coordinate is used for the horizontal axis of the graph and Z for the vertical one while in the top view the Y coordinate is used for the horizontal axis of the graph and X for the vertical one.

By default, when starting the program or changing the type of view, the graphic limits correspond to the maximum measurements of the table or a maximum height of 1m, allowing a minimum perception of the size of the part printed on the table.

The selected layer is highlighted using a different color from the remaining layers to represent it while in the toolpath view, only the highlighted layer is represented from the top view.

5.2. Orientation

The main objective of the program, as explained before, is to define the orientation of the tool parallel to the surface of the parts. This is achieved by defining the orientation at points parallel to the corresponding edge and command the robot to move between these points.

In doing so, the printing process should improve, reducing/removing the need for supporting material and improving the surface finish, as mentioned in **Chapter 2**. Thus, the following analysis becomes valid only for perimeter points, while the tool has vertical orientation on any other type of point.

For the algorithm to be functional with different geometries, the orientation of the tool is based on the comparison between two successive layers with perimeter points.

For the first layer, since all the material deposited is supported, the orientation is always defined as vertical. Whenever there is no perimeter material below a certain point, the orientation is defined as vertical since the right approach is by changing the slicing methodology.

The program's methodology works through an analysis of each point, layer by layer. A flowchart with a schematic representation of the whole process can be found in **Appendix C**

The program starts by going through the values in the Sortedlist and uses them as keys first in the perimeter dictionary, then infill and finally support.

If the perimeter dictionary as values for a certain height, it means that for that layer there are perimeter points. As such, the program finds the closest layer below also with perimeter points. If both layers do not have the same total number of perimeter points, vertical orientation is applied to each point of the current layer, otherwise, the analysis continues. The justification for this adaptation is written in **Chapter 6.**

Starting the analysis of a point, the program runs through all the points of the lower layer with the same perimeter lap, subtracting the X and Y coordinates in order to find the two closest points.

A line is drawn by joining these points. On that line, a virtual point is marked which represents the value of the closest coordinates to our point. If the coordinates of the virtual point are equal to the current point, the orientation is vertical. Likewise, if they are too far apart it means that they do not belong to the same face, being assigned vertical orientation to the current point.

5.2.1. Calculation of rotations

On RobotStudio the orientation of a tool integrated at the end of a robot is defined through the respective Tool Center Point (TCP) Z axis. Therefore, when applying rotations around the TCP X, Y, Z axes, the tool orientation changes.

The methodology chosen for the case study explores a combination of the Tait-Bryan angles and the intrinsic convention to calculate desired orientations for the tool.

Since the tool’s position is on the current points coordinates the orientation should direct the tool to the virtual point’s coordinates this is obtained by intercepting the TCP Z axis with the virtual point. To do so, the program follows the next process outside the main program:

It starts by subtracting the X, Y, Z coordinates of the current point to the coordinates of the virtual point. This results in a set of coordinates that can be defined as a new point. With this point represented in a 3D referential, the rotations needed for the tool are the rotations that intercepts the referential Z axis with the virtual point as represented in **Figure 21**.

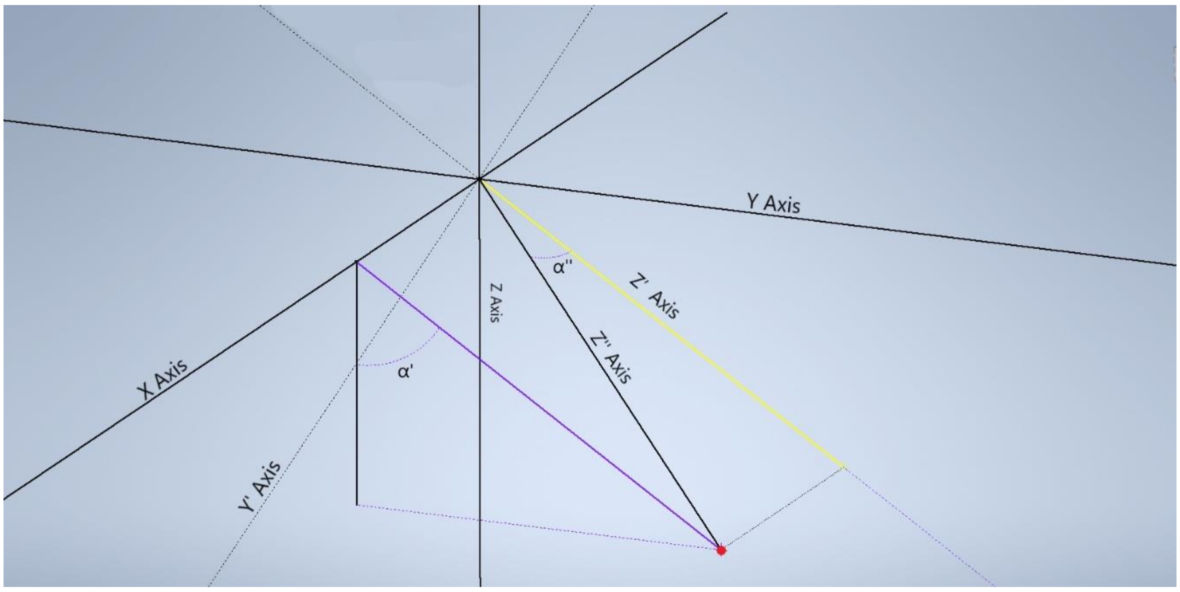


Figure 21- Representation of the rotations applied in the referential.

The new point is marked with red color. As said before the objective is to calculate rotations that intercept the Z axis with this point. This can be done using trigonometric calculations.

The rotation method used is Euler Angles with rotations ordered according to X-Y-Z. The rotation around Z is always null because it does not affect the orientation of the tool, only its direction.

The first rotation, which is around the X axis, causes the Z axis to intersect the y and z coordinates of the point as shown in the example of **Figure 21**. The value for this rotation, α', is obtained by the inverse tangent of y coordinate of the point divided by its z coordinate as shown in the equation (1).

$$\alpha' = tan^{-1}(\frac{y}{z}) \tag{1}$$

The second rotation which is around the rotated Y' Axis, intercepts the Z' axis with the coordinates of the point as represented in the example of **Figure 21**. The value for this rotation, α'', is obtained using the equation below with x, y, z being the coordinates of the point as shown in the equation (1).

$$\alpha'' = tan^{-1}(\frac{x}{\sqrt{y^2 + z^2}}) \tag{2}$$

With this calculation done, the program defines the order of rotations for the current point equal to (α’, α’’, 0).

5.3. Data output

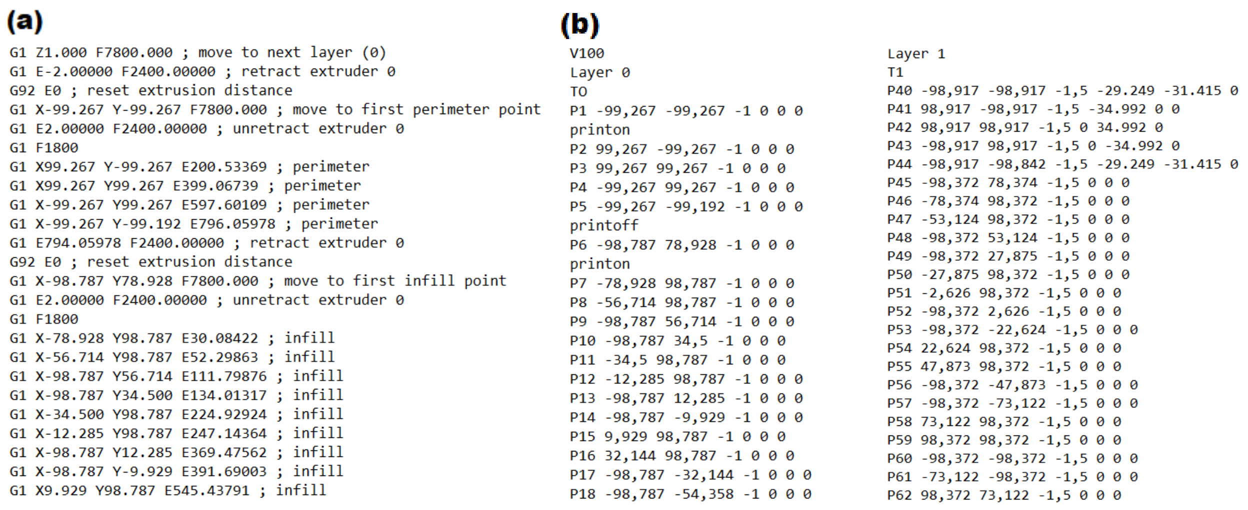
The information is sent to RobotStudio in a Generic text file (.txt) that contains, in a similar way to a G Code, positions and instructions to control the robot. The positions are sent using the point number, its coordinates and rotations. The instructions refer to the method used to apply the orientation to TCP, in other words, the movement of the robot.

As seen in **Chapter 4**, the program allows to define a form of orientation and movement by default and to create layer intervals with different orientations and movements. There are five possible ways to orientate and move the TCP (explained in **Chapter 6**) that correspond to a code:

- TO: Vertical orientation
- T1: Edge with orientation change applied during movement
- T2: Edge with orientation change applied before movement
- T3: 2-point average with orientation change applied during movement
- T4: 2-point average with orientation change applied before movement

At the beginning of the Generic file, a T0 is written referring to vertical orientation as explained before, to the standard method followed by the coordinates of the points and orientations of the first layer. Before start writing the coordinates and orientations of the next layer, the standard T chosen is written. From there, at the beginning of each layer, the program checks which method it's applied. If it differs from the previous, the T corresponding to the new one is written.

Before each point, the program checks whether it’s coordinates are present in the unretract list or the retrack list. If they are present in one of them, the program writes printoff or printon, respectively, before writing the coordinates and rotations of the point. The first lines of the file contain the velocity defined by the user.


(a)

```
G1 Z1.000 F7800.000 ; move to next layer (0)
G1 E-2.00000 F2400.00000 ; retract extruder 0
G92 E0 ; reset extrusion distance
G1 X-99.267 Y-99.267 F7800.000 ; move to first perimeter point
G1 E2.00000 F2400.00000 ; unretract extruder 0
G1 F1800
G1 X99.267 Y-99.267 E200.53369 ; perimeter
G1 X99.267 Y99.267 E399.06739 ; perimeter
G1 X-99.267 Y99.267 E597.60109 ; perimeter
G1 X-99.267 Y-99.192 E796.05978 ; perimeter
G1 E794.05978 F2400.00000 ; retract extruder 0
G92 E0 ; reset extrusion distance
G1 X-98.787 Y78.928 F7800.000 ; move to first infill point
G1 E2.00000 F2400.00000 ; unretract extruder 0
G1 F1800
G1 X-78.928 Y98.787 E30.08422 ; infill
G1 X-56.714 Y98.787 E52.29863 ; infill
G1 X-98.787 Y56.714 E111.79876 ; infill
G1 X-98.787 Y34.500 E134.01317 ; infill
G1 X-34.500 Y98.787 E224.92924 ; infill
G1 X-12.285 Y98.787 E247.14364 ; infill
G1 X-98.787 Y12.285 E369.47562 ; infill
G1 X-98.787 Y-9.929 E391.69003 ; infill
G1 X9.929 Y98.787 E545.43791 ; infill
```

(b)

```
V100
Layer 0
TO
P1 -99,267 -99,267 -1 0 0 0
printon
P2 99,267 -99,267 -1 0 0 0
P3 99,267 99,267 -1 0 0 0
P4 -99,267 99,267 -1 0 0 0
P5 -99,267 -99,192 -1 0 0 0
printoff
P6 -98,787 78,928 -1 0 0 0
printon
P7 -78,928 98,787 -1 0 0 0
P8 -56,714 98,787 -1 0 0 0
P9 -98,787 56,714 -1 0 0 0
P10 -98,787 34,5 -1 0 0 0
P11 -34,5 98,787 -1 0 0 0
P12 -12,285 98,787 -1 0 0 0
P13 -98,787 12,285 -1 0 0 0
P14 -98,787 -9,929 -1 0 0 0
P15 9,929 98,787 -1 0 0 0
P16 32,144 98,787 -1 0 0 0
P17 -98,787 -32,144 -1 0 0 0
P18 -98,787 -54,358 -1 0 0 0
```

```
Layer 1
T1
P40 -98,917 -98,917 -1,5 -29.249 -31.415 0
P41 98,917 -98,917 -1,5 -34.992 0 0
P42 98,917 98,917 -1,5 0 34.992 0
P43 -98,917 98,917 -1,5 0 -34.992 0
P44 -98,917 -98,842 -1,5 -29.249 -31.415 0
P45 -98,372 78,374 -1,5 0 0 0
P46 -78,374 98,372 -1,5 0 0 0
P47 -53,124 98,372 -1,5 0 0 0
P48 -98,372 53,124 -1,5 0 0 0
P49 -98,372 27,875 -1,5 0 0 0
P50 -27,875 98,372 -1,5 0 0 0
P51 -2,626 98,372 -1,5 0 0 0
P52 -98,372 2,626 -1,5 0 0 0
P53 -98,372 -22,624 -1,5 0 0 0
P54 22,624 98,372 -1,5 0 0 0
P55 47,873 98,372 -1,5 0 0 0
P56 -98,372 -47,873 -1,5 0 0 0
P57 -98,372 -73,122 -1,5 0 0 0
P58 73,122 98,372 -1,5 0 0 0
P59 98,372 98,372 -1,5 0 0 0
P60 -98,372 -98,372 -1,5 0 0 0
P61 -73,122 -98,372 -1,5 0 0 0
P62 98,372 73,122 -1,5 0 0 0
```

Figure 22- Pyramid G-Code (a) and created Generic file (b).

This information is compatible with the station developed by my colleague Francisco in RobotStudio, allowing the simulations and analyses of the behavior of the robot.

6. DEVELOPMENT

During the development and testing phase problems were detected and after analysis and reflection, it was determined that they were caused either by G-Code data or by the integration of the algorithm and RobotStudio. To solve these errors, certain adaptations have been implemented in the program.

Below the problems are identified, and their cause and implemented solution explained.

6.1. G-Code

6.1.1. Several perimeters in the same layer

In order to increase the thickness of the piece walls, Slic3r has the option to choose the number of perimeters for the part, that is, how many toolpaths for perimeters are, in any part of the piece, between the exterior of the piece and the interior (infill).

When executing the code for a part with more than one perimeter, certain orientations were calculated using points of different perimeters. When the algorithm looks for the nearest points on the layer below, it can pair points of different perimeters, which leads to incorrect orientations.

To avoid the mixing of points of different perimeters, a new variable has been introduced in the components of the points, which dictates the perimeter in which they are positioned. Using this new variable, the program was able to compare a point only with the corresponding perimeter points, correcting the errors.

When this was introduced, a new error appeared. This error occurs when two consecutive layers have different perimeter numbers. This happens when the part contains separate sections, as shown in the example below:

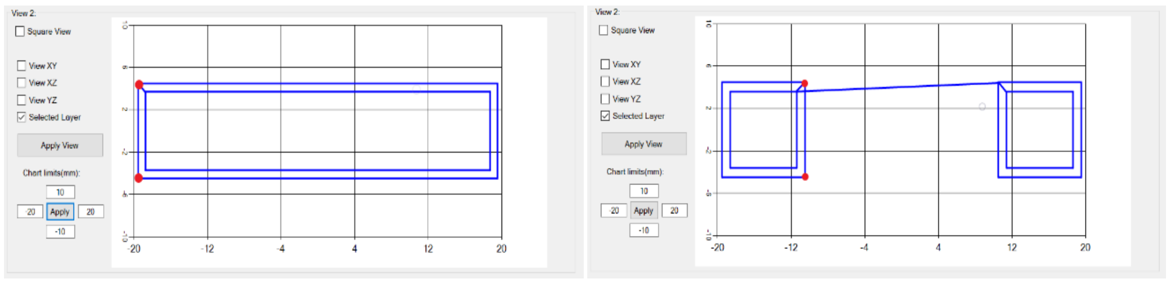


Figure 23- Representation of layers 35 (left figure) and 36 (right figure) of a test part.

In this situation, layer 35 has only one section leading to a total of two perimeters. As layer 36 has two separate sections, it contains a total of four perimeters. As the algorithm looks for the points closest to the two red points marked on layer 36, the result would be the two red points on layer 35. This would create incorrect orientation. Different methods where tested to solved this problem but none proved suited as such, to prevent this problem, when two consecutive layers have a different number of perimeters, the program instructs all points with vertical orientation.

6.1.2. "Fifth point"

A perimeter is completed when the tool returns to the starting point of that perimeter, but in doing so, there would be excessive deposition of material due to the extrusion of material twice in the same place.

To avoid this, the slicing software automatically generates an extra point close to the first one, thus not allowing the tool to pass over the same point twice. This was defined as the fifth point because during the pyramid tests, to complete a square in a layer, the extra point would be the fifth point of the layer as shown below:

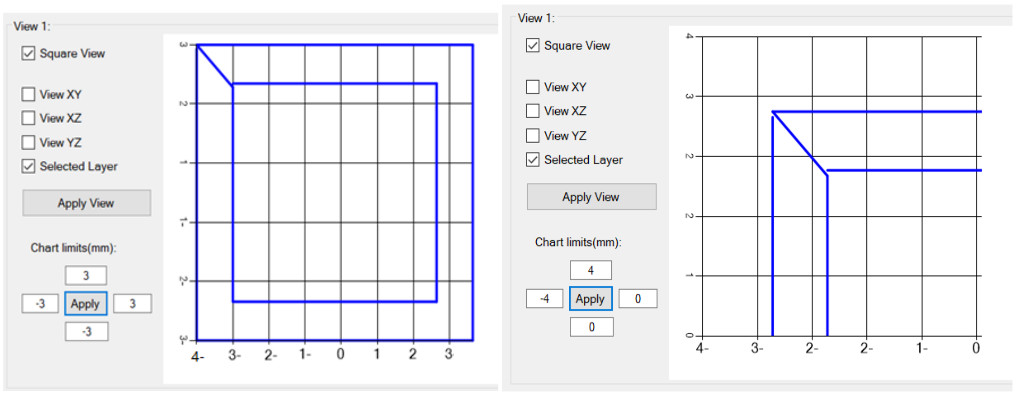


Figure 24- Representation of a pyramid layer that demonstrates the "fifth point".

Since these points are very close (by default the size of the extruder tip defined on Slic3r) it creates a problem: if these points have different orientations, the Robot would have to change the tool orientation to a very small offset which leads to abrupt movements.

The solution found was for the program to define the orientation of the extra point equal to the first and as they represent the same position, this seems to be the right approach.

6.2. Algorithm-RobotStudio integration

6.2.1. Maximum Angle

Depending on the piece geometry, the tool can reach an orientation in which it is working in a horizontal position. This is likely to cause movement problems or even damage the equipment by hitting already printed sections. Therefore, a procedure has been implemented to defines the maximum possible angle.

The user can define the maximum angle but if it is not defined, the program automatically applies a maximum angle of 30º to the horizontal plane since, as shown in **Chapter 2**, this inclination allows the printing of even horizontal walls.

This procedure occurs after obtaining the virtual point and consists of changing its X and Y coordinates, with the respective scale, bringing the virtual point closer to the point to be analyzed, thus reducing the resulting inclination in the tool. With these new coordinates of the virtual point, the program follows the normal procedure.

The following figure shows how the mechanism works: X' and Y' correspond to the initial coordinates of the virtual point that results in the tool being inclined α'. Since α' is larger than the maximum inclination permitted, the program converts this value to the maximum angle, α'', which in turn decreases the coordinates of the virtual point with the respective scale to X'' and Y''.

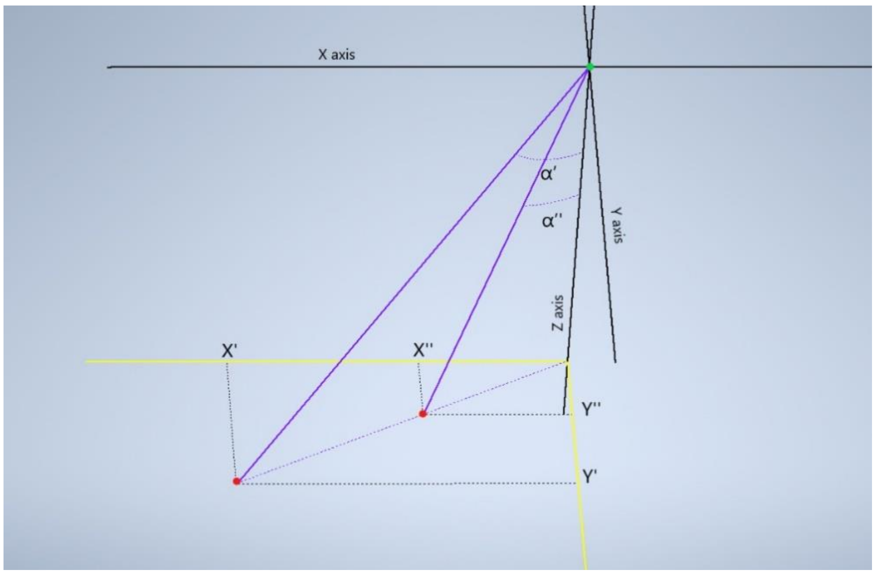


Figure 25- Example of the coordinate's reduction on a virtual point and impact on orientation.

6.2.2. Big variations for small offsets

The first approach of the algorithm was able to work correctly when the part to print did not include curved sections.

By printing a circle with this high number of orientations, the tool was constantly adapting its orientation. Besides, some points can be out of the pattern, which leads the tool to drastically alter its orientation in short distances, leading to meaningless movements.

To avoid this problem, a defense mechanism was introduced: an offset rotation limiter. This mechanism allows the user to limit the maximum degree variation between 2 consecutive points when they are within a defined offset.

This mechanism is applied after the rotations of the current point are calculated by checking whether the distance between it and the previous point is greater than the maximum offset defined by the user. If so, the program compares the current rotations with the rotations of the previous point. If the difference is greater than the maximum defined by the user, the current rotations will be changed so as not to exceed that limit.

This limit is not implemented during changes between perimeters. In the two-point average method, it is the average of both rotations that is used in comparisons.

6.2.3. Types of orientations

To carry out the main objective of this work, which is to orientate the tool according to the part surface, the algorithm created orients the tool by comparing the points of two successive layers. But during the tests, it was noticed that the tool used the orientation of a point during a path.

A better way to define the orientation during a movement would be to use the average orientation between the points that define that trajectory. Thus, the **two-point average** was created.

The difference is applied during the data output, where each method does the following:

For **edge orientation,** the program calculates the rotation to the point and sends the data to a generic file that contains the number of points with their coordinates and rotations.

For **two points average**, there is always stored rotation, which corresponds to the previous point's rotation (if it is the first point of that layer and/or perimeter lap, the stored rotation is equal to 0). To calculate the average, the previous rotation and the one calculated are used for the current point (for the first point this method makes the orientation half the calculated orientation). This average is sent to the generic file. The program stores the original rotation to be applied to the average of the following point.

A **vertical orientation** is also present as an option. This is automatically used by the program in situations where problems are encountered and the implemented solution involves defining the orientation of the points as vertical. This is obtained by assigning the rotations the value zero.

6.2.4. Types of movement

During simulations, two methods for changing the tool orientation were created and both were kept since the most suitable method always depended on the test piece. These methods are applied through configurations in the RobotStudio simulation, so how they are implemented will not be explained, just how they work.

To make the program flexible, the user has the ability to choose which one is applied. For situations where vertical orientation is implemented this concept does not make sense since the orientation does not change.

6.2.4.1. Orient During Movement

This type of movement causes the robot to start a toolpath with the orientation of the starting point and finish it with the orientation of the end point. The change is gradual along the path.

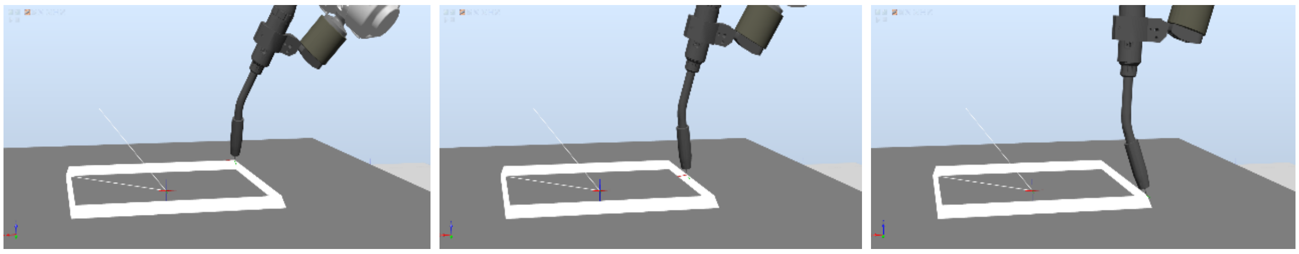


Figure 26- Orientate during movement for the print of the pyramid.

6.2.4.2. Orient Before Movement

This type of movement causes the robot to change the orientation of the tool before starting the movement. The orientation defined for the path is the one calculated for the point of arrival.

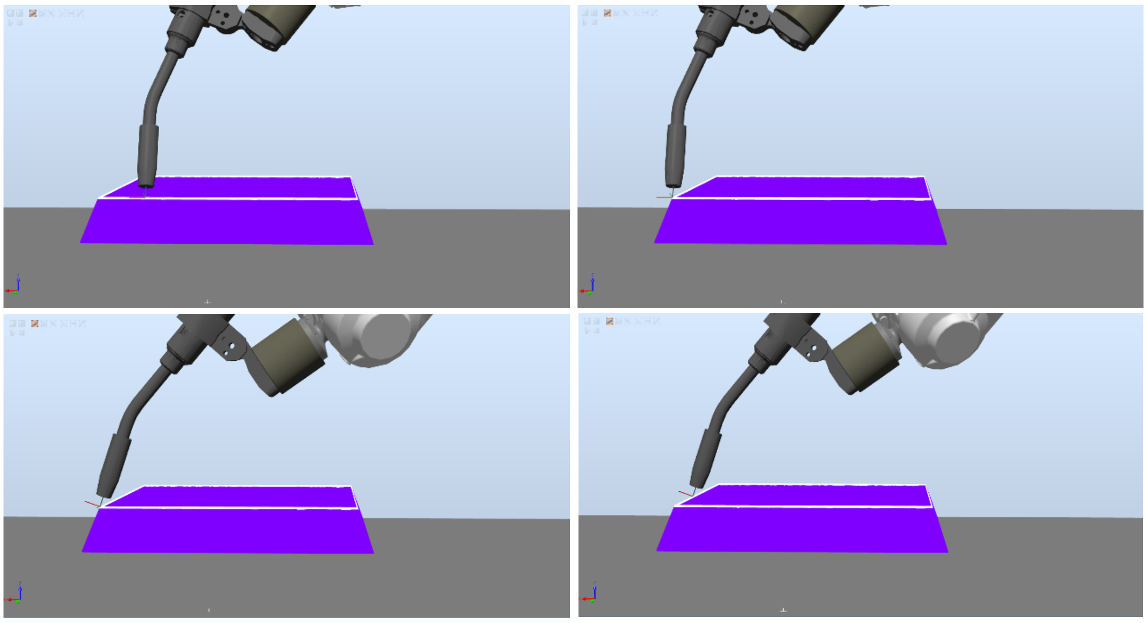


Figure 27- Orientate before movement for the print of the pyramid.

7. SIMULATION RESULTS

After the program development and implementation of all adaptations, two pieces from the test files were selected for the final simulation. The chosen pieces were a simple pyramid and two circles connected by a straight bar, similar to a dumbbell.

The printing of both pieces was simulated using different combinations of movement type and orientation method, which led to the following observations.

7.1. Pyramid

This piece consists of a hollow quadrangular pyramid. It was chosen for the ease of identifying whether the orientation algorithm is working correctly. The 3d model is represented in **Figure 12** and the sliced view in **Figure 13**.

7.1.1. Type of movement

For the pyramid simulations, orienting the tool before moving obtained the desired results by being able to keep the orientation constant along the path.

Orient during movement made the tool change the orientation along a path. For big movements, this may not be ideal as it can impact the deposition of material.

7.1.2. Orientation method

As for the type of orientation, the two-point average perform the best, as expected since the tool always kept its orientation parallel to the surface of the piece.

The edge orientation made the tool move inclined to the side, which does not represent the desired orientation.

7.2. Circular Dumbbell

During testing, circles were undoubtedly the most problematic geometries to print so as a final difficulty test, a part containing spherical and plane geometries was drawn. The 3d model is represented in **Figure 12** and the sliced view in **Figure 13**.

7.2.1. Type of movement

For this piece, orienting before the movement had bad results. As said before, circular paths are composed of numerous points. Thus, the orientation of the printing tool is constantly changing, which is a problem since this type of movement requires a small amount of time to readapt to the orientation of the next point.

This causes an inconstant movement, certainly not applicable to this part. To solve this situation, orientation during the movement was applied and obtained good results.

7.2.2. Orientation method

The edge orientation, although movements were possible for both parts, there were some uncertainty about the practical application. The two-point average performs the best, as expected since the tool always kept its orientation parallel to the surface of the piece.

As mentioned earlier, the control of the orientation by offset, solved the inconsistent pattern of point placement. However, the user should be aware that the distance between successive points is crucial to define the correct offset to choose and the corresponding maximum rotation to apply. In the simulations, 3 degrees for 15mm offset obtained good results. For the pyramid piece, this feature is not relevant since, there are only 4 different orientations per layer (one for each face), which is a considerably low number of orientation changes and as the points are significantly apart, there is no need to smooth the movements of the robot.

8. CONCLUSION

This thesis presents the solution developed together with my colleague Francisco that aims to implement a methodology to adapt the orientation of the printing tool connected to an industrial robot. The main objective of this solution would be to improve the printing process of metallic components, by reducing the printing time and the material needed. For this purpose, an algorithm was developed, based on the geometry of the pieces, implementing an orientation parallel to its surface.

All the tests were carried out on the RobotStudio interface since the actual system was not fully operational. Although there are studies that demonstrate the benefits of orienting a print tool parallel to the surface to be printed, it is important to perform tests on the real system to ensure the correct functioning of the system and verify the properties of the printed part.

Throughout the development of the program, it was necessary to implement several methods of orientation and movement of the robot to ensure the adaptation for different part geometries.

In conclusion, although all the objectives have been achieved, the program is not fully automatic because it cannot automatically identify the best methodology or combination of methodologies to apply to each piece, which leads the user to intervene in the selection of methods to apply. Also, to print horizontal surfaces without material below is still necessary to use support material. The solution for this would have to involve changing the printing methodology created in the slicing program.

8.1. Future Work

This solution should be interpreted as an initial approach that opens up possibilities for the following developments and/or improvements:

- Testing in the real system the applicability of the orientation adjustment during printing
- Study the mechanical behavior of the printed parts
- Complement the program with an adapted slicing system

- Automate the selection of the best method

BIBLIOGRAPHY

"3D Printer Build." https://www.3dprinterbuild.com/what-is-3d-printing/ (October 23, 2020).

"Additive Manufacturing Delivers Economies of Scale AND Scope - 3DEO - Metal Additive Manufacturing." https://www.3deo.co/strategy/additive-manufacturing-delivers-economies-of-scale-and-scope/ (October 23, 2020).

"C Sharp – Wikipédia, a Enciclopédia Livre." https://pt.wikipedia.org/wiki/C_Sharp (October 25, 2020).

Chalvin, Maxime, Sébastien Campocasso, Vincent Hugel, and Thomas Baizeau. 2020. "Layer-by-Layer Generation of Optimized Joint Trajectory for Multi-Axis Robotized Additive Manufacturing of Parts of Revolution." *Robotics and Computer-Integrated Manufacturing* 65(February): 101960. https://doi.org/10.1016/j.rcim.2020.101960.

DebRoy, T. et al. 2018. "Additive Manufacturing of Metallic Components – Process, Structure and Properties." *Progress in Materials Science* 92: 112–224.

"Defining The Industrial Robot Industry and All It Entails | RIA - Robotics Online." https://www.robotics.org/robotics/industrial-robot-industry-and-all-it-entails (October 23, 2020).

Filomeno, Martina, and Stewart Williams. 2015. "Wire + Arc Additive Manufacturing vs. Traditional Machining from Solid: A Cost Comparison. Available Online: Http://Waammat.Com/Documents/Waam-vs-Machining-from- Solid-a-Cost-Comparison (Accessed on 1 October 2019)." http://waammat.com/documents/waam-vs-machining-from-solid-a-cost-comparison.

"IFR Presents World Robotics Report 2020 - International Federation of Robotics." https://ifr.org/ifr-press-releases/news/record-2.7-million-robots-work-in-factories-around-the-globe (October 23, 2020).

"ISO/ASTM 52900:2015(En), Additive Manufacturing — General Principles — Terminology." https://www.iso.org/obp/ui/#iso:std:iso-astm:52900:ed-1:v1:en (October 23, 2020).

Ivántabernero, Amagoia Paskual, Pedro Álvarez, and Alfredo Suárez. 2018. "Study on Arc Welding Processes for High Deposition Rate Additive Manufacturing." *Procedia CIRP* 68(April): 358–62. http://dx.doi.org/10.1016/j.procir.2017.12.095.

J. Norberto Pires. 2020. "Robótica (e Biónica)."

Jiang, Jingchao, Jonathan Stringer, Xun Xu, and Ray Y. Zhong. 2018. "Investigation of Printable Threshold Overhang Angle in Extrusion-Based Additive Manufacturing for Reducing Support Waste." *International Journal of Computer Integrated Manufacturing* 31(10): 961–69. https://doi.org/10.1080/0951192X.2018.1466398.

Kazanas, Panagiotis et al. 2012. "Fabrication of Geometrical Features Using Wire and

Arc Additive Manufacture.” *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 226(6): 1042–51.

Mehnen, Jörn, Jialuo Ding, Helen Lockett, and Panos Kazanas. 2014. “Design Study for Wire and Arc Additive Manufacture.” *International Journal of Product Development* 19(1–3): 2–20.

Mohan Pandey, Pulak, N. Venkata Reddy, and Sanjay G. Dhande. 2003. “Slicing Procedures in Layered Manufacturing: A Review.” *Rapid Prototyping Journal* 9(5): 274–88.

“MX3D Bridge | MX3D.” https://mx3d.com/projects/mx3d-bridge/ (October 21, 2020).

Pires, J. Norberto, Germano Veiga, and Ricardo Araújo. 2009. “Programming-by-Demonstration in the Coworker Scenario for SMEs.” *Industrial Robot* 36(1): 73–83.

“RobotStudio - ABB Robotics.” https://new.abb.com/products/robotics/robotstudio (October 25, 2020).

Silva, Maria Inês Castro e, Eurico Gonçalves Assunção, and Maria Luísa Coutinho Gomes de Almeida. 2018. “Study of Deposition Strategies of a Wire + Arc Additive Manufactured Component Materials Engineering Examination Committee.” (May).

“TAKENAKA CONNECTOR | MX3D.” https://mx3d.com/projects/takenaka-connector/ (October 21, 2020).

“The Industrial Revolution: From Industry 1.0 to Industry 4.0.” https://www.seekmomentum.com/blog/manufacturing/the-evolution-of-industry-from-1-to-4 (October 23, 2020).

“Titanium Pressure Vessel for Space Exploration Built Successfully Using the Wire + Arc Additive Manufacturing Process.” https://www.cranfield.ac.uk/press/news-2019/titanium-pressure-vessel-for-space-exploration-built-successfully-using-the-waam-process (October 23, 2020).

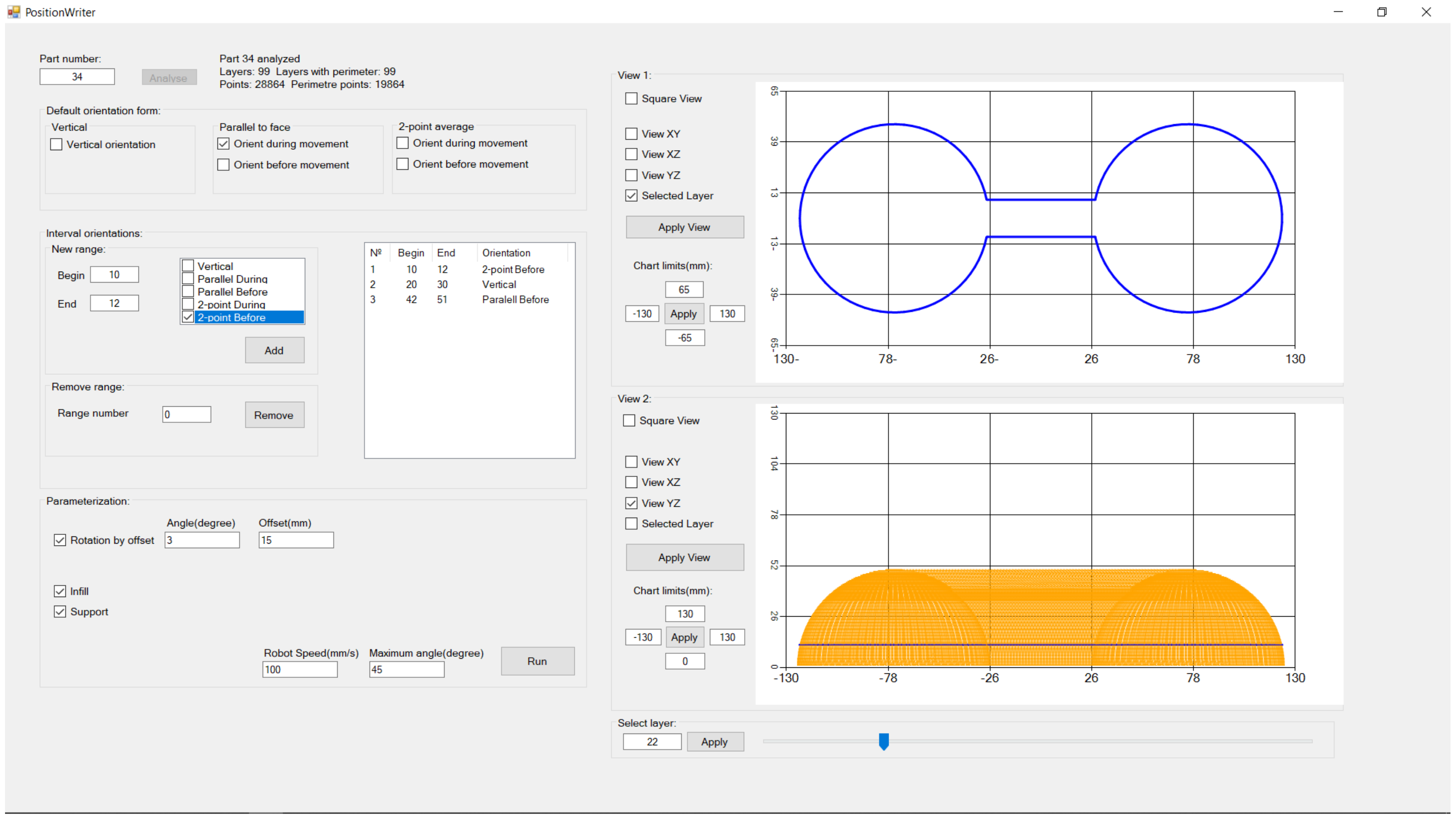
“Titanium Pressure Vessel for Space Exploration Made Using 3D Printing.” https://www.pesmedia.com/titanium-pressure-vessel-space-exploration-additive-manufacturing/ (October 21, 2020).

“What Is Slic3r? – Simply Explained | All3DP.” https://all3dp.com/2/what-is-slic3r-simply-explained/ (October 25, 2020).

Wilson, Mike. 2014. Implementation of Robot Systems: An introduction to robotics, automation, and successful systems integration in manufacturing *Implementation of Robot Systems: An Introduction to Robotics, Automation, and Successful Systems Integration in Manufacturing*. Elsevier Inc.

Wu, Bintao et al. 2018. “A Review of the Wire Arc Additive Manufacturing of Metals: Properties, Defects and Quality Improvement.” *Journal of Manufacturing Processes* 35(August): 127–39. https://doi.org/10.1016/j.jmapro.2018.08.001.

APPENDIX A



Part number: 34

Part 34 analyzed
Layers: 99 Layers with perimeter: 99
Points: 28864 Perimetre points: 19864

Default orientation form:

Vertical
- [ ] Vertical orientation

Parallel to face
- [✓] Orient during movement
- [ ] Orient before movement

2-point average
- [ ] Orient during movement
- [ ] Orient before movement

Interval orientations:

New range:

Begin 10
End 12

- [ ] Vertical
- [ ] Parallel During
- [ ] Parallel Before
- [ ] 2-point During
- [✓] 2-point Before

Nº	Begin	End	Orientation
1	10	12	2-point Before
2	20	30	Vertical
3	42	51	Paralell Before

Remove range:

Range number 0

Parameterization:

- [✓] Rotation by offset Angle(degree) 3 Offset(mm) 15
- [✓] Infill
- [✓] Support

Robot Speed(mm/s) 100
Maximum angle(degree) 45

View 1:
- [ ] Square View
- [ ] View XY
- [ ] View XZ
- [ ] View YZ
- [✓] Selected Layer

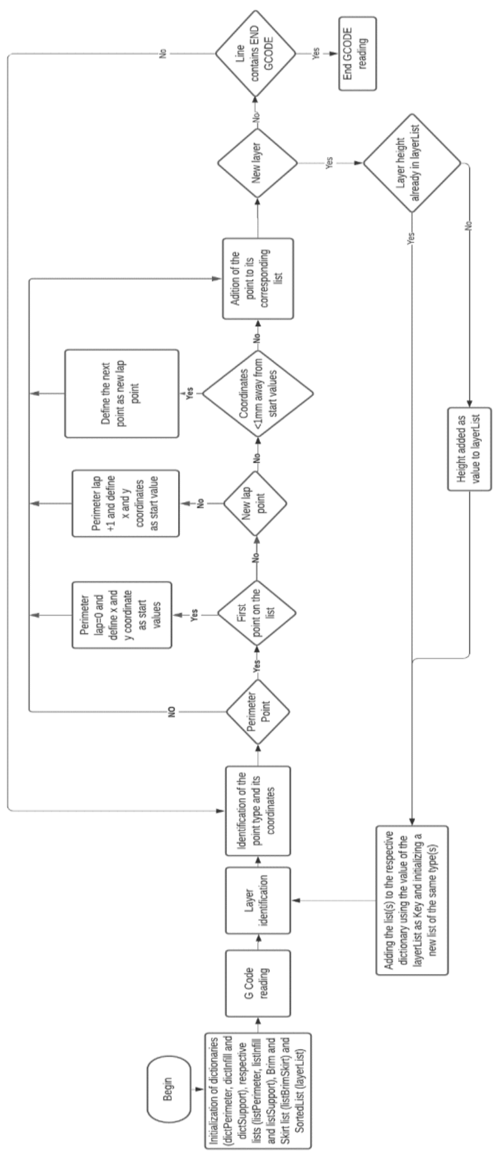
Chart limits(mm): 65, -130, 130, -65

View 2:
- [ ] Square View
- [ ] View XY
- [ ] View XZ
- [✓] View YZ
- [ ] Selected Layer

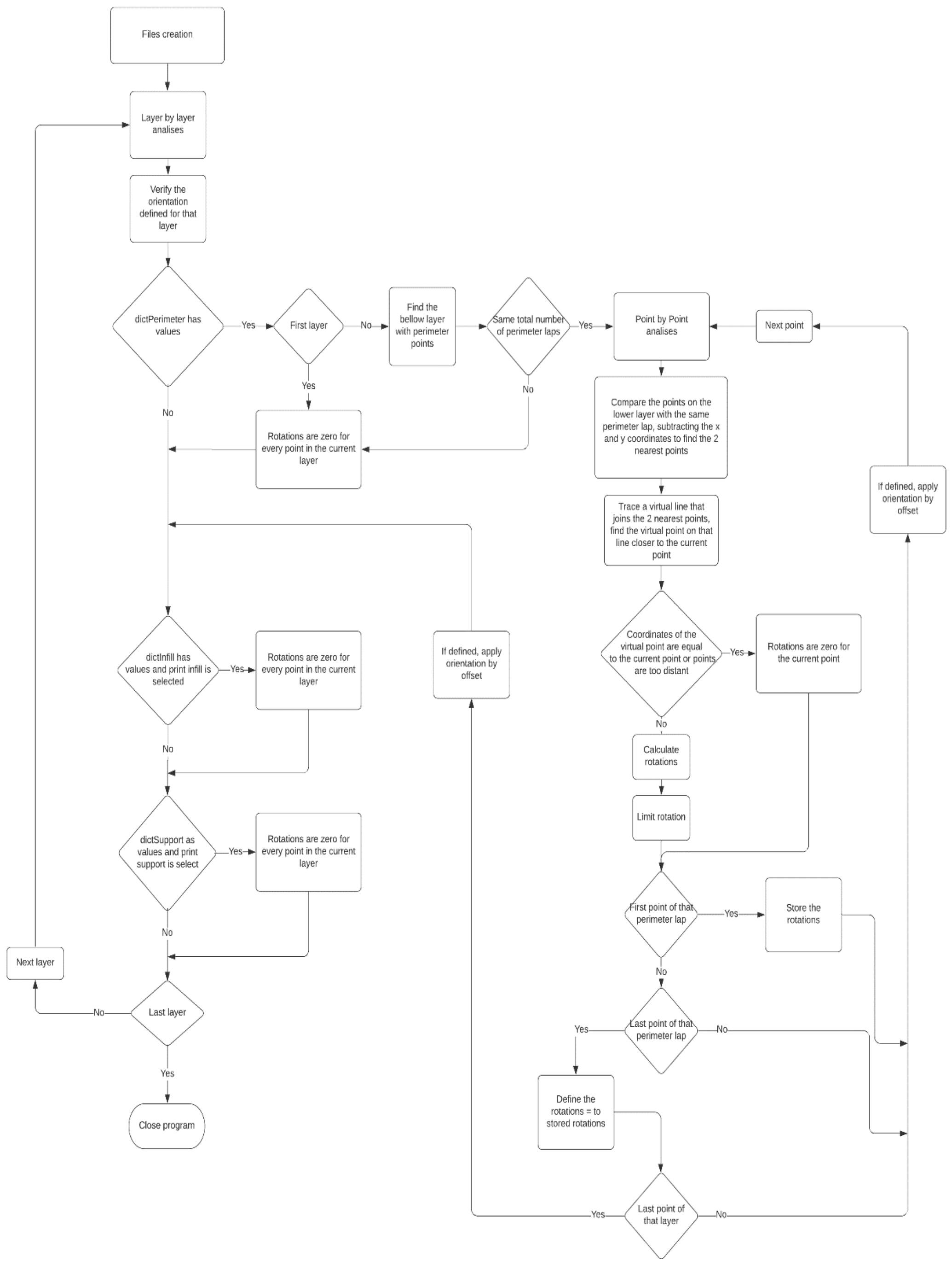
Chart limits(mm): 130, -130, 130, 0

Select layer: 22

APPENDIX B


Begin
Initialization of dictionaries (dictPerimeter, dictInfill and dictSupport), respective lists (listPerimeter, listInfill and listSupport), Brim and Skirt list (listBrimSkirt) and SortedList (layerList)
G Code reading
Layer identification
Identification of the point type and its coordinates
Perimeter Point
NO
Yes
First point on the list
Yes
Perimeter lap=0 and define x and y coordinate as start values
No
New lap point
No
Perimeter lap +1 and define x and y coordinates as start value
No
Coordinates <1mm away from start values
Yes
Define the next point as new lap point
No
Adition of the point to its corresponding list
New layer
No
Line contains END GCODE
Yes
End GCODE reading
No
Yes
Layer height already in layerList
Yes
No
Height added as value to layerList
Adding the list(s) to the respective dictionary using the value of the layerList as Key and initializing a new list of the same type(s)

APPENDIX C


Files creation
Layer by layer analises
Verify the orientation defined for that layer
dictPerimeter has values
Yes
First layer
No
Find the bellow layer with perimeter points
Same total number of perimeter laps
Yes
Point by Point analises
Next point
Yes
No
No
Rotations are zero for every point in the current layer
Compare the points on the lower layer with the same perimeter lap, subtracting the x and y coordinates to find the 2 nearest points
If defined, apply orientation by offset
Trace a virtual line that joins the 2 nearest points, find the virtual point on that line closer to the current point
dictInfill has values and print infill is selected
Yes
Rotations are zero for every point in the current layer
If defined, apply orientation by offset
Coordinates of the virtual point are equal to the current point or points are too distant
Yes
Rotations are zero for the current point
No
No
Calculate rotations
Limit rotation
dictSupport as values and print support is select
Yes
Rotations are zero for every point in the current layer
First point of that perimeter lap
Yes
Store the rotations
No
Next layer
No
No
Last layer
Last point of that perimeter lap
Yes
No
Yes
Define the rotations = to stored rotations
Close program
Yes
Last point of that layer
No