Trajectory Generation for Robotic Applications using Point Cloud Data

Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Jorge Beltrá Fuerte
Tutor:
José García Rodríguez

Mayo 2023

Trajectory Generation for Robotic Applications using Point Cloud Data

Autor
Jorge Beltrá Fuerte

Tutor
José García Rodríguez
Tecnología Informática Y Computación

Grado en Ingeniería Robótica

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Mayo 2023

Abstract

EN

Throughout this project, the aim is to develop a program that, based on the processing of three-dimensional point clouds belonging to objects or shapes, achieves the creation of trajectories through route generation algorithms. These trajectories can be subsequently followed by a manipulator robot. The motivation for this project arises as a contribution to the current development of industrial applications involving robotics, where the use of computer vision techniques is increasingly relevant. This implementation can be beneficial in industrial settings such as welding of metal parts, where trajectory planning along the object's surface is required.

ES

En este proyecto se plantea el desarrollo de un sistema, que a partir del procesamiento de nubes de puntos tridimensionales pertenecientes a objetos o figuras, defina trayectorias mediante algoritmos de generación de rutas, que podrán ser seguidas posteriormente por un robot manipulador. La motivación de este proyecto surge como aportación al desarrollo de aplicaciones industriales en las que está presente la robótica, donde cada vez toma más relevancia el uso de técnicas de visión por computador e inteligencia artificial. Los resultados del proyecto pueden ser aplicados a nivel industrial, en trabajos como soldadura de piezas metálicas donde se requiere de una planificación de trayectorias a lo largo de la superficie del objeto con el que se trabaja.

Acknowledgments

This project would not have been possible without the support of my parents, my sister and all my friends that have stayed during the whole process until this moment.

Thanks to José García, for helping me during this project.

Thanks to all these people who stayed in the first steps of this project in Sweden.

Also thanks to all my teachers of these years for giving the knowledge during the degree that have allowed me to accomplish this goal.

Finally, thanks to everybody that has been part of my life through all these years.

The best way
to predict your future
is to create it.

Abraham Lincoln and Peter Drucker.

Contents

List of Figures

List of Tables

Listings

1 Introduction

In this chapter, the project will be introduced along with its background, highlighting its relevance in addressing current issues and its objectives. Additionally, this chapter contains a part dedicated to the sustainability of the project, including environmental and social impacts, as well as an overview of the document, providing a guide with the structure and contents of the report.

1.1 Background

In today's world, robotics has experienced a significant increase in its applicability and presence in various domains. Additionally, the number of robots being created and used in industries, services, and other fields is continuously increasing. According to World Robotics (2022), this growth and expansion are predicted to continue, making robotics part of everybody's daily life.

Moreover, robotics is opening new frontiers, particularly in industry, where it is an important component of Industry 4.0, which is a field that is improving using technological elements such Internet of Things (IoT), Artificial Intelligence (AI), Virtual Reality (VR) and robotics, among others (Goel & Gupta, 2020). In this ambit, it is possible to differentiate between industrial robots that are used to automatize hazardous or repetitive tasks, and collaborative robots, which are able to work in the same space with humans reducing the payload of work that they could have.

However, actual investigations are also focusing on more techniques that could be implemented in order to support and make more efficient the work that robots are doing in order to let the worker the most specific tasks that can be difficult to program or that need human supervision. Some of these fields are computer vision or AI, which are giving good results (Kakani et al., 2020) because they make it easier to detect faults and it is possible to do evaluations faster than a human can do without the need of being supervised by an operator.

For this reason, there is an increasing trend in using a combination of these techniques with robotics, which provides feedback to production, improves task efficiency, and enables the connection of visual systems to robots, allowing for more accurate decision-making through automated analysis (Hägele et al., 2016).

Following this trend, the objective of this project is to use computer vision to develop a system that generates trajectories for manipulators by creating a designated path. The project wants to achieve this goal through computer vision techniques (focused on three-dimensional) and path planning methodologies.

1.2 Problem statement

Nowadays, industries are focused on increasing production and enhancing efficiency. The companies are looking for sustainable transportation, distribution, and environmentally-friendly monitoring, goals that are emphasized by the persistent economic growth of many countries (Sharma et al., 2023). Hence, automating processes and production lines is a crucial task as it can save time and money. This is because automation can lead to more benefits with less wasted energy and products, because the intervention of a human is not always needed. Moreover, the introduction of technology in industry has also significantly improved production efficiency, being able to produce more in less time.

To contribute to this search for improving productivity and getting a less harmful impact on the environment produced by the industry, this project aims to create an efficient program that can analyze an object and follow a path that is compounded by the key points of an object. The project is not focused on a particular case because it aims to be applied to different problems.

This project can be useful in different areas such as welding, where it is necessary to go from one point to another in a three-dimensional space; drawing over a surface of an object; cutting different shapes on diverse materials. These are some examples where the project could fit, but there are more applications for it.

Finally, the ambition behind this work is to contribute to current research with more documentation and a new project that could be used in future research to improve all the mentioned issues and help to make more efficient work.

1.3 Sustainability

During the process of developing a project is important to consider the impact that it will have in the environment, how it can contribute to the world and more aspects in order to be as beneficial to the rest of the world as possible.

For that reason, the Agenda 2030 (*Figure* 1.1) exists, which is a number of purposes that are pretend to be achieved by the year 2030 to control the development of the world to make it sustainable.

In this project these goals have been taken into account, considering that it contributes in the following ones:

- **3. Good health and well-being**: The project aims to reduce the workload of workers in various areas. Therefore, it is considered that the project could help prevent injuries and fatigue.

- **8. Decent work and economic growth**: In terms of industry, the project can reduce workers' workload and increase factory production, thereby conserving resources. Thus, it can improve the economy of industries while creating a better work environment for the operators.

- **9. Industry, innovation and infrastructure**: The project aims to contribute to the industry by providing new ways to program production lines and adapt them to the current state of the industry using modern technologies.

- **12. Responsible consumption and production**: When an industry is automated, it tends to conserve resources, improve production, and reduce waste.

- **13. Climate action**: Although the project does not directly influence the environment, it can contribute to reducing pollution and mitigating the factors that cause it.

SUSTAINABLE DEVELOPMENT GOALS
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

Figure 1.1: Global goals of Agenda 2030. Retrieved from: `https://www.un.org/`

1.4 Overview

This section provides a summary of the distribution of the different points into which the thesis is divided:

- **Chapter 2. Theoretical frame of reference**: This chapter contains information about various concepts that need to be clarified for a full understanding of the project.

- **Chapter 3. Objectives**: This chapter will explain the aim of this project and the different tasks that must be undertaken to achieve the final goal.

- **Chapter 4. Literature review**: This section will present literature and previous works in the fields that this project encompasses.
- **Chapter 5. Methodology**: This chapter will explain the process followed to complete the project.
- **Chapter 6. Development**: This chapter outlines the process and the steps that comprise the project.
- **Chapter 7. Results and discussion**: In this part of the project the results that have been obtained during the experiments are shown.
- **Chapter 8. Conclusions**: A brief summary of the achieved objectives and some results if it is needed.
- **Chapter 9. Future works**: This chapter presents some ideas for the expansion of the project and potential ways to improve the current project.
- **References**: This section lists the sources that were used to obtain information and conduct research during this project.

2 Theoretical frame of reference

This chapter contains relevant notions to understand the project and the different parts that compose it. Explaining some concepts related to the specific tasks that conform the general purpose.

2.1 Industrial robotics

There are many ways to define an industrial robot. One is the term that defines a programmable, mechanical device that is designed to execute tasks with precision, speed, and efficiency within an industrial area.

The International Organization for Standaridization (ISO) describes these devices as an "automatically controlled, reprogrammable multipurpose manipulator, programmable in three or more axes, which can be either fixed in place or fixed to a mobile platform for use in automation applications in an industrial environment" ((ISO 8373:2021),2021).

The most remarkable characteristics to define this kind of robots by International Federation of Robotics (IFR) in 2022 are:

- **Reprogrammable**: Is the capability of being reconfigured to accomplish new rules or commands without the need of interacting with the hardware.
- **Multipurpose**: Refers to the capacity of the machine to fulfill different functions and adapt it to different applications.
- **Manipulator**: It is a mechanism consisting of an arrangement of segments, jointed or sliding relative to one another that is capable of grasping, moving and manipulating objects.
- **Axis**: Means the direction used to specify the robot motion in a linear or rotary mode.

These robots are typically used for automating manufacturing processes, such as assembly, welding, handling or painting, among others. They are equipped with multiple axes of motion, enabling them to move in various directions and perform complex operations.

Moreover, during recent years sales of manipulators have increased, reflecting the growing demand for robotic systems in industry. The interest of improving the efficiency and productivity of factories and the intention to modernize the industry have been an essential point of this increment. As a result, the sales of manipulators have experienced substantial growth as IFR shows in the *Figure* 2.1 in recent years, highlighting their essential role in the future of industrial automation.

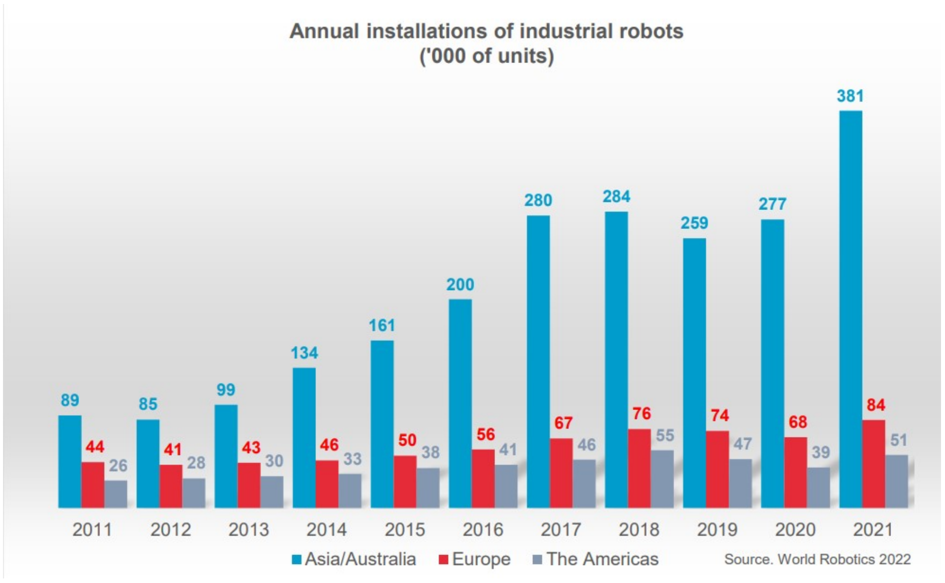


Figure 2.1: Installations of industrial robots. Retrieved from: `https://ifr.org/img/worldrobotics/Executive_Summary_WR_Industrial_Robots_2022.pdf`

Currently, industrial robots are being integrated with various technologies, including cameras for visual feedback and task execution, AI, and other devices to augment their capabilities. By combining these technologies, the performance and functionality of these machines are significantly improved.

2.2 Open 3D library

Open3D is an open-source library that is designed to work with 3D data, providing various tools for processing and visualization of point clouds, meshes, and Red, Green, Blue and Depth (RGB-D) images. It offers a wide range of algorithms and methods that facilitate the development of applications in the field of computer vision and robotics. The library provides functionalities such as 3D data input/output, registration, filtering, segmentation, and visualization. It includes various improvements, such as new algorithms for feature extraction and matching, enhancements in the visualization module, and support for 3D data formats (Zhou et al., 2018). Its use in this project will allow for efficient and accurate processing of 3D models, contributing to the creation of trajectories from objects. The version that will be used is the 0.17.0.

2.3 Three-dimensional vision system

This section pretends to clarify important concepts of vision system focused on three-dimensional treatment of images. By exploring different methodologies that are used in three-dimensional vision system to get characteristics and work with point clouds.

2.3.1 Ways of capturing

An important aspect of three-dimensional imaging is the depth of the scene. Depth can be captured in several ways: one method involves a specialized camera that obtains depth information using a laser. Another option is to use multiple cameras, with each camera knowing the position of the others. Alternatively, depth information can be gathered using a single camera that takes several photos, knowing the transformation between each position where the photos are taken.

Depth information in three-dimensional imaging can be acquired through various methods. In addition to laser scanning and stereo vision, depth sensors that use structured light or time-of-flight principles offer real-time solutions for capturing depth. Moreover, advancements in AI have significantly enhanced these techniques.

2.3.2 Point clouds

They are a data structure that can represent point in the three-dimensional space. Each point has the information of its position with the coordinates x, y and z. These structures can contain more information such as color or intensity of the point (Point Cloud Library, 2023).

The used data for tasks with this three-dimensional structure is compound by an amount of points, an example can be the *Figure* 2.2 (Open3D, 2018).

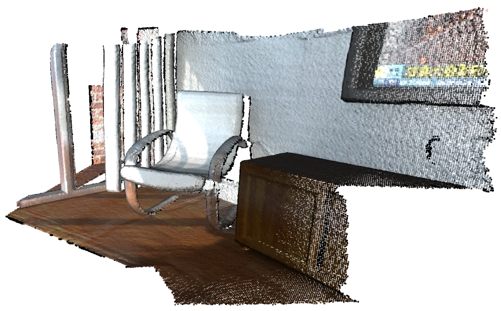


Figure 2.2: Point cloud example. Retrieved from: `http://www.open3d.org/docs/0.11.1/tutorial/geometry/pointcloud.html`

As can be seen in the previous figure, the displayed image consists of visible points. These points form a cloud that allows for the differentiation of the three-dimensional aspects of these structures.

2.3.3 Voxel downsampling

Normally, point clouds are heavy data structures due to they contain a lot of information. Working with them can suppose high computational costs, making it essential to minimize the quantity of data processed in applications. This reduction is necessary to improve the efficiency and speed of algorithms, aiming for more effective and expedient processing.

One method to achieve this data reduction is through voxel downsampling. This can be understood as creating grids in two-dimensional data, or boxes in space, that contain a number of these points. The process reduces the information by calculating a relevant characteristic, such as the centroid of the point cloud, for example. This centroid is saved as a single point that represents all the points within the box. (Hacinecipoglu et al., 2020).

This method results in a reduced point cloud that retains relevant information for representation (*Figure* 2.3), but it has minor data than the complete point cloud. Consequently, it can be used in algorithms that will work more efficiently.

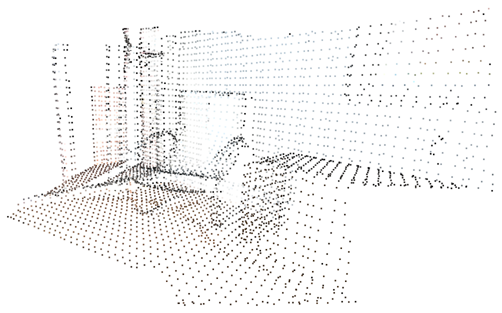


Figure 2.3: Voxeled point cloud example. Retrieved from: `http://www.open3d.org/docs/0.11.1/tutorial/geometry/pointcloud.html`

The figure shown is compared with the unfiltered one from the previous section. Although the scene is still identifiable, it has fewer points. This reduction in points was the principal purpose of using voxel downsampling..

2.3.4 Segmentation

Segmenting an image, or objects within an image, involves identifying which regions belong to it and grouping the pixels that share certain characteristics into an area.

To perform this separation, it's important to understand that the process involves more than just working with a single image; it starts by looking for possible matches or similarities between multiple images.

There are different techniques that allow to perform these matches, such as Sum of Squared Differences (SSD) that can have problems if the light is not uniform, Normalized Cross Correlation (NCC)and MI.

Once the matching process is done, the next step is to group the pixels that belong to the object of interest. This can be achieved by different segmentation techniques such as region growing, clustering, or thresholding.

Region growing is a technique that starts with a single pixel and iteratively adds neighboring pixels that satisfy certain criteria. Clustering is a technique that groups pixels based on their similarity in terms of color, texture, or other features. Thresholding is a technique that separates pixels based on their intensity values, such that pixels above a certain threshold are grouped together.

The choice of segmentation technique depends on the specific application and the characteristics of the objects to be segmented.

2.3.5 Key points

An essential aspect of trajectory planning is determining the points that make up the path to be followed. Since this project focuses on working with geometric pieces for industry, the path needs to be created such that it connects the vertices of the object. In other words, the trajectory is determined by the edges, but the specific points along this trajectory are dictated by the vertices.

In the case of three-dimensional objects, key points are typically detected by using a technique called 3D feature detection. This involves analyzing the object's shape and geometry to identify important landmarks or features, such as corners or edges. One popular method for 3D feature detection is based on the Harris corner detector, which is a well-known algorithm used for 2D feature detection. The Harris corner detector works by analyzing the intensity variations in the image to identify regions where there is a high gradient, indicating a change in the image's intensity. These regions correspond to corners or edges in the image, which can then be used as key points for path planning. It can be observed in the *Figure* 2.4.

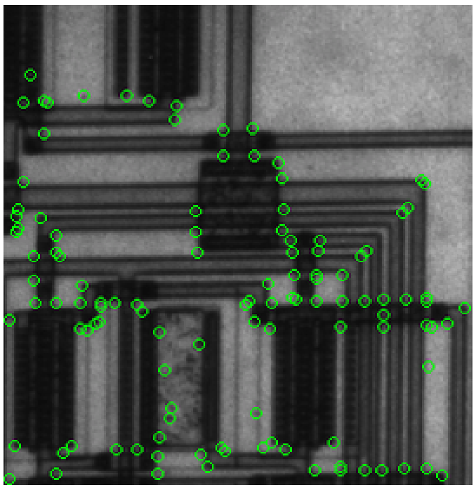


Figure 2.4: Detection of corners with FAST. Retrieved from: `https://es.mathworks.com/help/vision/ug/local-feature-detection-and-extraction.html`

To apply the Harris corner detector to 3D objects, researchers have developed various techniques based on multi-scale analysis and local shape descriptors. These methods involve analyzing the object's shape and geometry at different scales and using local descriptors to identify key points that are invariant to scale and orientation. Examples of local shape descriptors used for 3D feature detection include Scale-Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF), which have been shown to be effective for detecting key points in complex 3D shapes.

Overall, the use of computer vision techniques for detecting key points in 3D objects is a crucial step in path planning for manipulator trajectories. By accurately identifying key points and using them to generate a path, robots can perform precise and efficient movements, which is particularly important in industrial applications where speed and accuracy are critical.

2.3.6 KDTree

A KDTree is a data structure used for organizing points in a k-dimensional space. It is a type of binary search tree, where each level of the tree splits the space into two parts along one of the k dimensions, one example of three-dimensional partition of the space to apply this algorithm is the *Figure* 2.5.

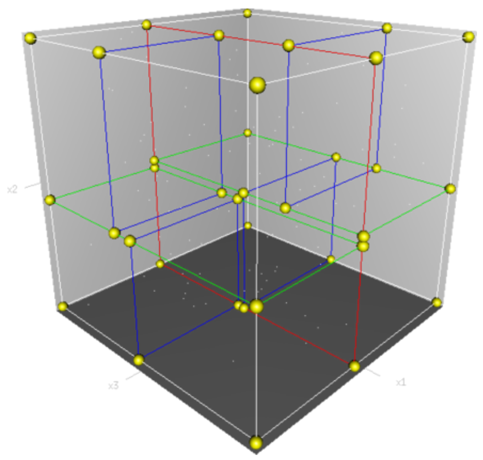


Figure 2.5: Representation of division of a three-dimensional space by KDTree algorithm. Retrieved from: `https://en.wikipedia.org/wiki/K-d_tree`

The primary use of KDTree is to enable efficient queries, such as finding the nearest neighbor or searching for points within a specified range.The performance of a KDTree depends on its balance, so it is crucial to use balanced partitioning methods when the tree is being constructed.

Once the tree is created, there needs to be a method to group points from the resulting cloud for various purposes such as reducing the number of points, grouping areas of similarities to define certain characteristics, among others. One of the most frequently used functions for point grouping is `query_pairs`, which is often associated with the KDTree. This function identifies all pairs of points within a specified distance from each other in a set of points. This can be useful in spatial data analysis, pattern recognition, and clustering.

Given a KDTree containing a set of points and a distance threshold, the `query_pairs` function returns a list of pairs of points such that the distance between each pair is less than or equal to the given threshold. This function takes advantage of the efficiency of the KDTree search. It traverses the tree and compares the distances between nodes in different branches to identify pairs that are within the specified distance threshold. By avoiding unnecessary comparisons and capitalizing on the spatial organization of data within the tree, the `query_pairs` function can identify pairs of points much faster than a naive approach.

2.4 Type of robots

To facilitate optimal movements, it is important to consider how these movements are executed, their limitations, and how they are controlled.

A robot is essentially comprised of links connected by joints (Barrientos et al., 2007). Typically, the links are rigid components that move in accordance with the action of the joints. Conversely, the joints are mobile elements that require an actuator to operate.

Joints can be categorized into different groups based on certain features. These characteristics include the DOF, which signifies the number of independent movements a joint can perform through an axis in space (Morales et al, 2014), and the nature of these movements. In robotics, joints typically have one DOF, and the most prevalent types of joints are prismatic (allowing translation along an axis) or revolute (allowing rotation around an axis) (Adolfsson & Schmidt, 2001).

Based on the type of joint and its DOF, distinctions can be drawn between: 1 DOF joints that can be either prismatic or rotational; 2 DOF joints that can be either cylindrical or planar; and 3 DOF joints, often referred to as spherical (Craig, 2005). These distinctions are illustrated in *Figure* 2.6.

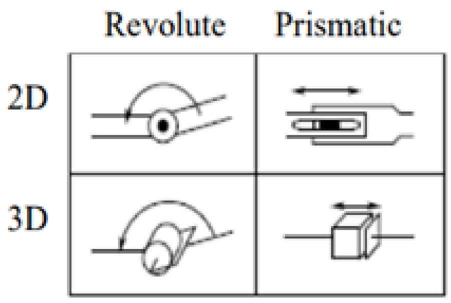


Figure 2.6: Symbolic representation of different joints (Wilson, 2006). Retrieved from: `https://www.researchgate.net/`

The classification of a robot can depend on the arrangement of its joints, that is, how they are connected, their types, and the sequence in which the links form the robot. Different configurations, as depicted in *Figure* 2.7, can make a robot more suitable for specific tasks.

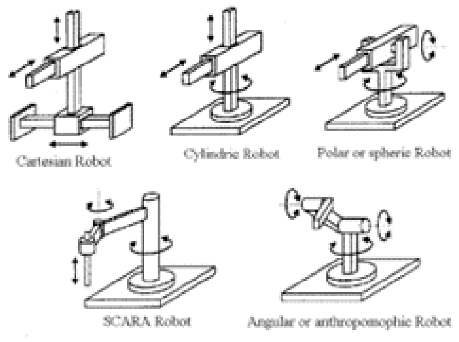


Figure 2.7: Most common configurations of robots (Rosales et al., 2002). Retrieved from: `https://www.researchgate.net/`

A robot with 7 DOF is considered redundant, as it possesses more DOF than necessary to reach any point in its operational space (Chirikjian, 1992). In this context, it operates in a 3D space where 6 DOF would suffice. However, this additional degree can help the robot overcome issues or achieve its goals, even if one of the other degrees fails.

2.5 Planning of trajectories

One essential part of this project is planning trajectories because the final goal is to create a path that the robot must follow, for this reason is necessary to explain concepts that will be used during the explanation of the project.

2.5.1 Movements

This section will demonstrate the distinctive movements that a manipulator robot can execute, as they differ from those of other types of robots, such as mobile robots.

During the execution of tasks, it is crucial to avoid collisions and joint-related issues. One of the challenges that may arise is the occurrence of singularities, which are positions where the robot loses some DOF because there are no feasible solutions (Cheng, 1997) or the robot does not have the possibility of performing a movement due to its own morphology.

Due to the limitations on allowed movements, robots must operate within a defined range of joint and link motions. In this section, we will explore three different types of manipulator movements.

The first type of movement involves the motion of a single joint. *Figure* 2.8 illustrates examples of each joint in Sawyer and the corresponding movements they can perform. In this movement, a specific joint is selected and performs the motion, either by rotation for a rotational joint or along an axis for a translational joint. The remaining parts of the robot remain static or adjust their configuration according to the movement of this particular joint. This type of movement is beneficial for avoiding singularities or positioning the robot with a specific configuration.

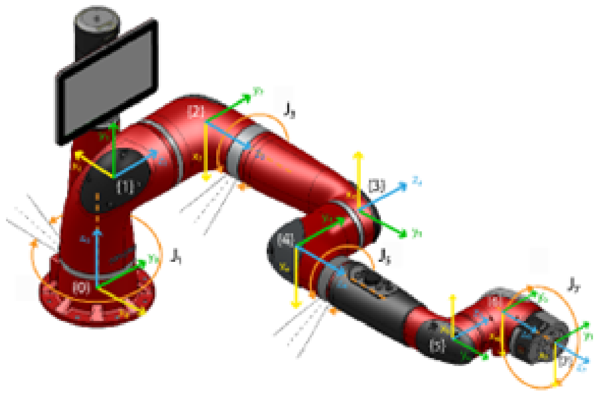


Figure 2.8: Representation of the movement of all joints of Sawyer (Cornejo et al., 2018)). Retrieved from: `https://www.semanticscholar.org/`

Manipulators have the capability to perform linear movements, which involve coordinating the motion of their joints to move the end-effector from one point to another in space, creating a straight line trajectory between these points (*Figure* 2.9). However, it's important to note that certain configurations may be unreachable due to obstacles or joint limitations.

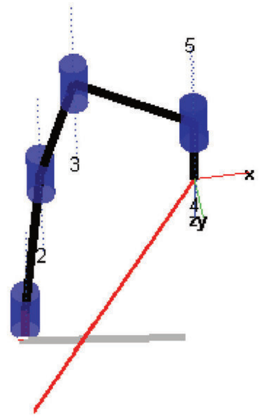


Figure 2.9: Representation of a linear movement (Pérez Ruiz et al., 2017). Retrieved from: `https://www.researchgate.net/`

In addition to linear movements, manipulators can also perform circular movements (*Figure* 2.10). This type of movement can be seen as a generalization of linear movement, as it involves transitioning from one point to another while following a curved path. Circular movements can be useful for avoiding obstacles or reaching joint positions that are otherwise unreachable.

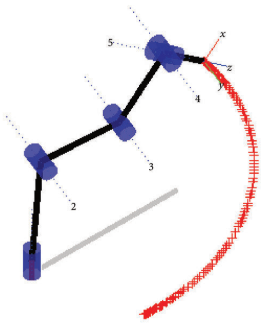


Figure 2.10: Representation of a circular movement (Pérez Ruiz et al., 2017). Retrieved from: `https://www.researchgate.net/`

2.5.2 Trajectories

Trajectory planning is a vital component in robotics that deals with the computation of motions to be executed by a robot. Trajectories play a fundamental role in robot control and consist of a geometric trajectory and a temporal law(Human Robotics, 2023). The planning of trajectories involves several important considerations :

- **Cartesian Space and Joint Space**: Planning in Cartesian space enables better visualization and adaptation but requires inverse kinematics computations at runtime to determine the corresponding joint configurations. On the other hand, planning in joint space using polynomials is suitable for point-to-point interpolations but may introduce unwanted oscillations in trajectories with many intermediate points.

- **Dynamic constraints**: These are restrictions due to the forces and torques involved in moving the robot. It is important to consider its effects to not provoke damage in the robot.

- **Use of Splines**: Splines offer a solution with minimal curvature among all interpolation functions that have continuity in the second derivative. They are particularly useful in pick-and-place trajectories, where concatenated polynomial schemes like 4-3-4 or 3-5-3 can be employed.

- **Application in Articular and Cartesian Spaces**: Trajectory planning methods proposed in joint space can also be applied in Cartesian space. However, Cartesian trajectories may encounter challenges due to singularities in the workspace.

- **Interpolation in Cartesian Space**: In Cartesian space, the number of points to be interpolated is typically low, allowing for the use of simple interpolation trajectories such as straight lines or circular arcs.

- **Velocity and Acceleration Considerations**: In some cases, it may be necessary to recalculate the trajectory result to adjust the joint motion to its velocity and acceleration limits, which can reduce the task execution time.

In summary, trajectories in robotics involve the planning and generation of smooth and precise robot motions. Considerations such as trajectory density, choice of planning space (Cartesian or joint), techniques like double normalization and spline interpolation, as well as limitations and challenges associated with each trajectory type, are essential for achieving efficient and safe control of robots in various applications.

2.6 Path planning

Once the image has been processed, the next step is to implement the route that the robotic arm will follow. This path planning is carried out by software that uses the detected key points of the shape. Various algorithms can assist in determining the fastest way to join these points.

The creation of the path involves searching for an accurate route based on the given example and finding it in an efficient manner. This process is executed by software, and among the possible routes, it seeks to find a good approximation that aligns with the mentioned characteristics.

This process is usually divided into two parts: graph generation and pathfinding algorithm implementation (Abd Algfoor et al., 2015). Since graphs are mathematical elements, various operations can be performed on them, some of which relate to finding the shortest or most efficient path between different points. Some of the algorithms that are used include Dijkstra, Floyd-Warshall, or Prim, among others.

However, these algorithms are not the most computationally efficient, leading to the exploration of new approaches that enhance this feature, with most of them based on AI. One of the best-known algorithms is Backtracking, a solution to optimization problems. It requires minimal computation time and a small number of parameters to achieve the best result in a reasonable time frame (Civicioglu, 2013). Backtracking consists of navigating through problems in various ways, comparing many possible solutions, evaluating each response, and choosing the most suitable and efficient method (Schmidt & Druffel, 1976).

Currently, more methods of improving these algorithms are being researched, and efforts are being made to enhance existing ones for use in larger problems. These algorithms use new structures such as trees, as seen in Rapidly Exploring Random Trees, or graphs, as in Probabilistic Roadmap Methods or Probabilistic Cell Decomposition. These structures organize information and employ probabilistic methods to calculate the most efficient path planning for the robot (Aarno et al., 2005).

Another methodology to create the path is by creating a list of continuous points along a contour, studying the surrounding area until all the contour pixels are included in the list (Alhusin Alkhdur et al., 2012). Alternatively, the Freeman chain code can be used in closed contours and organized according to their lengths (Jaquier, 2016).

2.6.1 Convex Hull

A convex hull is the smallest convex polygon or polyhedron that completely encloses a set of points in a Euclidean space. In 2D space, the convex hull can be visualized as a rubber band stretched around the outermost points of the set, an example is the *Figure* 2.11, a result of extracting it of the base of the point cloud of cube's base.

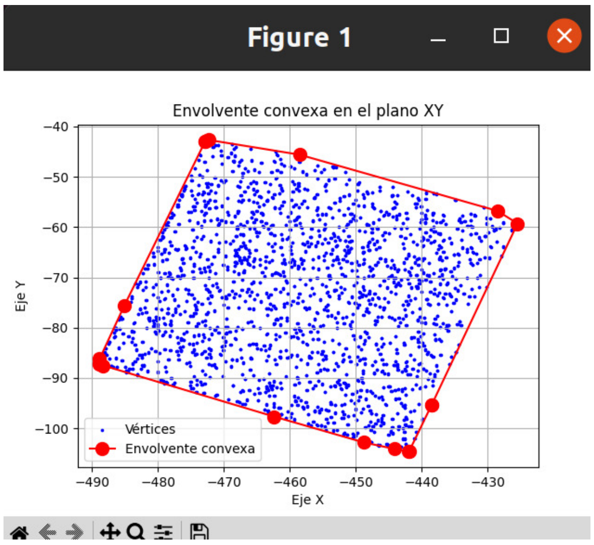


Figure 2.11: Extraction of the convex hull of the base of a point cloud cube.

In 3D space, it can be thought of as the smallest convex polyhedron that contains all the points. Convex hulls are important in various fields such as computational geometry, computer graphics, and optimization. Common algorithms to compute the convex hull of a set of points include the Gift Wrapping algorithm, Graham's Scan, and QuickHull.

2.6.2 Naive approach

A naive approximation or approach refers is a simplistic method used to estimate or solve a problem, often without taking into account relevant factors. It involves making simplified assumptions or using basic techniques to arrive at a solution or estimate, typically sacrificing accuracy for simplicity or ease of implementation.

Naive approximations or approaches are often used as initial or baseline methods before more sophisticated or refined techniques are applied. Some naive methods used in programming can be look for the closest neighbour in order to find the shortest path to arrive from one point to another, comparing a point with the rest to find the nearest point (Cheng et al., 2021).

This methods are usually really expensive computationally and are used in a few problems, that requires low resources and has an easy implementation.

2.6.3 Depth-First Search (DFS)

DFS is a graph traversal algorithm used to explore all the vertices and edges of a graph. It starts at an initial vertex and explores as far as possible along each branch before backtracking. DFS can be implemented using recursion or an explicit stack data structure. It is particularly useful in solving problems that involve searching for a specific path or condition in a graph, such as finding connected components, topological sorting, and maze-solving.

2.7 Control of robots

An essential part of generating trajectories is to define the movement and control of the robot. It will condition the way that the robot performs the movements, so it is necessary to know about the controllers and how they work.

2.7.1 Controllers

Controllers are an essential part of any robotic system that states the rules to communicate the operator's orders through an algorithm to the robot in a way that it can execute them (Kickert & Mamdani, 1993). It is the physical part that allows communication between the user and the robot. The controller receives the instruction and sends the orders to the robot to be executed.

To work correctly, the controller must have essential parts and the communication must follow some rules. First, it is necessary to have input signals, due to that they will be the orders that it is going to communicate to the robot, it internally has the software to send the outputs that are the executed actions. However, during the communication, the robot must be able to make decisions and check the environment while it is working. The system needs a part to measure if the robot has reached the goal and send corrections if it has not happened.

For that reason, the control of an activity can be performed in different ways and with different equipment. Depending on the environment and the tasks to perform the used system to control can change, one of the most used systems is a vision controller (camera, IR sensors...) that allows the system to obtain different kinds of external information (Sogo et al., 2001) helping in the performance of the tasks and being the part that measures the error between the reference and the current state.

There are more ways to perform the control, depending on the method that will be used, it can be composed of different agents. If it is a simple control that does not need to continuously measure the error cannot have included some equipment and if it is necessary to check, it needs equipment like sensors or other accessories that measure the necessary conditions.

2.7.2 Loops of control

In an automatic machine control, two schemas are typically employed to provide a simple illustration of the system's operation. The first configuration is open-loop control, where a signal is sent, processed by the controller, followed by execution of the motion or program, and finally, an output is created, as represented in *Figure* 2.12. This control scheme is beneficial for machines tasked with periodic and continuous functions, and for machines that must not be affected by environmental changes or external disturbances, thereby ensuring stable conditions throughout the process. Systems with this type of control commonly rely on simple input signals to mitigate any potential disturbances (Borovic et al., 2005). Although open-loop control is not utilized in certain applications today due to the necessity for superior control over interferences and changes for complex tasks, it remains prevalent in tasks that require performance of an order.

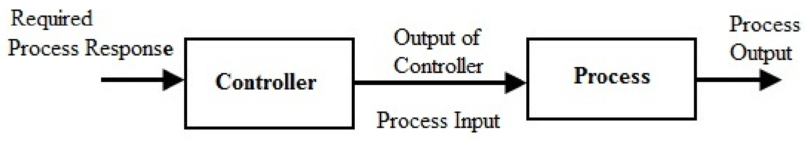


Figure 2.12: Schema of a general open-loop control system. Retrieved from: `https://www.elprocus.com/what-is-an-open-loop-control-system-its-working/`

The other one is the closed-loop control or feedback control. It consists of the same as the open-loop control, but it also includes one or more sensors that measure the difference between the goal and the current state and it sends a signal that is processed by the controller until it reaches the objective, represented in *Figure* 2.13.

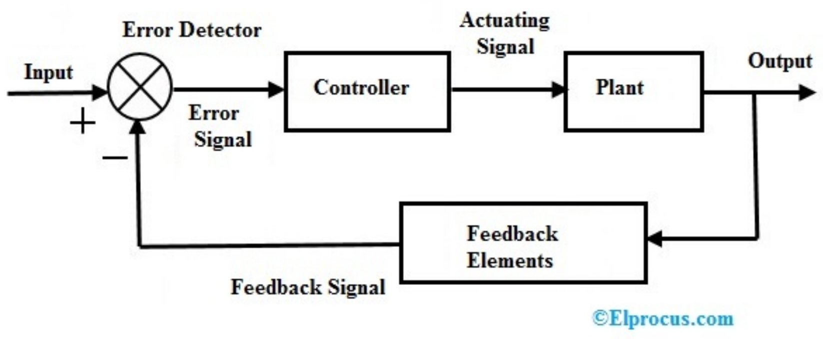


Figure 2.13: Schema of a general closed-loop control system. Retrieved from: `https://www.elprocus.com/what-is-a-closed-loop-control-system-its-working/`

How it is possible to observe in the image, there is an input signal that is processed, and it generates an output signal that is measured, this signal is compared with a reference, it sends a signal that is processed indicating the next step, and this cycle finishes when the goal is reached.

This control system avoids external interferences measuring the new state and making corrections. Moreover, with closed-loop control, the machine can be exposed to a changing environment or activities that do not always follow the same pattern. For that reason, it is the chosen method of control for changing tasks or works that need continuous analysis. Besides, closed-loop control is used because this method of control requires, in most cases, less wiring, minor installations and lower maintenance costs and it gives flexibility. Although it can increase the response time and has some weaknesses regarding the communication (Casado-Vara et al., 2019).

As it has been indicated before, the closed-loop control requires a sensor or measuring equipment guided by the proper algorithm. For example, one way to control is by the vision system, equipping a camera in the environment (internal or external) that sends corrections between the aim and the current state, passing through software it indicates the following step (Abt et al., 2011).

2.8 Matrix transformation

A matrix transformation, also known as a linear transformation, is a mathematical operation that applies a matrix to a vector or set of vectors to produce a transformed vector. It represents a fundamental theoretical framework in robotics, allowing for precise geometric operations on objects and vectors within a mathematical framework.

It allows to represent an object in the point of view of an external factor in order to let the other system to know how to get the points of the figures in its own reference, transforming the points to its comprehension. The *Figure* 2.14 shows the idea of the application of these matrix.

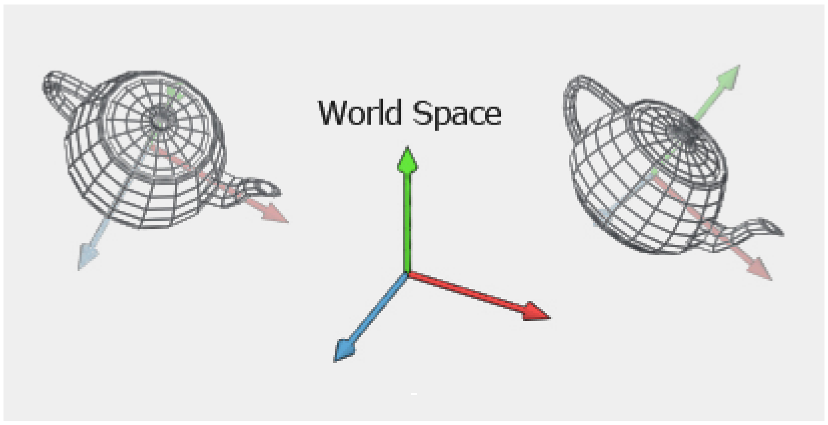


Figure 2.14: Application of a transformation matrix. Retrieved from: `https://docs.hektorprofe.net/graficos-3d/25-matriz-de-vista-y-camara/`

Matrix transformations find extensive use in robotics due to their versatile applications. Firstly, they play a crucial role in robot kinematics by modeling the spatial relationships between different components of a robot's mechanism. Through the utilization of transformation matrices, based on conventions such as Denavit-Hartenberg, the position and orientation of each robot link relative to its neighboring links can be accurately defined. By concatenating these transformations, the overall pose of the robot's end effector can be determined, enabling precise control of robot movements.

Secondly, matrix transformations are essential in solving both forward and inverse kinematics problems. The forward kinematics equations employ transformation matrices to calculate the pose of the robot's end effector given specific joint angles. Conversely, the inverse kinematics problem involves determining the joint angles required to achieve a desired end effector pose. By manipulating transformation matrices, robots can accurately position themselves in desired configurations, enabling tasks such as path planning and motion control.

Matrix transformations also form the foundation for tasks such as trajectory planning. Through their utilization, robots can perceive and interact with their environment, enabling tasks such as object manipulation, mapping, and localization.

3 Objectives

3.1 Aim and objectives

This project aims to create a program to process point clouds and create an efficient path that must be followed by a robot. To complete this purpose, some points are proposed:

1. **Create a point cloud**: The object to analyze is a three-dimensional entity, usually industrial pieces. For that reason, it is necessary to recreate it with point clouds in stead of images.
2. **Detection of keypoints**: Once the point cloud has been created, to create a path is necessary to detect the most relevant points to follow.
3. **Creation of the path**: With the obtained points it is possible to create the way that the robot must follow with efficient algorithms.
4. **Path Performance**: Once the path has been created and all the coordinates have been referenced to the robot, the robot should be able to execute it.

3.2 Extent and delimitation

The objective of the project is to create a path in the three dimensions starting with the capture of images or with different kind of files that could be uploaded.

This work could be extended adding an interface, improving the used methods or testing it in other kind of activities. However, due to the current unavailability of a robot, these extensions are not possible at the moment.

Moreover, the absence of a physical robot for this stage of the project limits the types of tests that can be performed to obtain results. Additionally, it limits the ability to assess the robot's reaction to varying conditions and object interactions, and to observe its potential.

4 Literature review

This section of the document is dedicated to analyzing previous researches related to this project and the areas that it pretends to contribute to. The goal of this section is to understand the current state of three-dimensional analysis of objects, path planning and performing of trajectories with robots. This understanding will be achieved by reviewing other works and research. Additionally, open questions will be identified for discussion during this project.

Path planning is a fundamental task in robotics, particularly in the area of manipulators. Traditionally, path planning has been performed using mathematical models and algorithms, but recent advancements in computer vision technology have made it possible to use three-dimensional data for path planning. This has led to increased interest in using 3D computer vision for path planning in manipulators, particularly for applications such as object recognition, grasping, and manipulation. In this literature review, it es explored the current state of research in path planning for manipulator trajectories with 3D computer vision.

Grasping is a common task for manipulator robots, but it can be complex depending on the shape of the object and other features. Bone et al. (2008) developed an automated modeling process of three-dimensional objects by computer vision to facilitate this task. This project combined the modeling of three-dimensional objects with the planning of trajectories to grasp the object. By integrating different technologies to extract relevant information from the object, they were able to plan a trajectory in four seconds, making an advancement in object manipulation through computer vision feedback.

Following this trend, Mousavian et al. (2019) used a neural network that analyzed the three-dimensional object by computer vision and chose the best way to grasp it. Combining both technologies, they were able to have a good percentage of success in their task. During this project, the network they used was trained to grasp unknown objects depending on the feedback detected by the robot and the outcomes predicted by the neural network, which was trained on data derived from object textures and shapes..

Therefore, the use of neural networks for extracting characteristics of a three-dimensional object is a trend. One example of this area is the work presented by Dasari et al. (2019), which is a neural network that tries to approximate any movement of manipulators. It has been trained with using a substantial volume of data from different videos of robots performing trajectories and grasping. The aim of this project is to generalize the actions of manipulators and create a network that allow them to learn different abilities to adapt the robot to new tasks, objects to grasp and different variables that can be changed. It tries to obtain a global architecture to solve manipulating problems.

Furthermore, computer vision is growing up and the methodology that is used is also evolving, for example, Abdollahi et al. (2020) present a method where improves the clustering to obtain more accurate relationships. That is important because, it can help to get better matches between real objects and their point clouds, improving the result of tasks that use it. This contribution allows to improve the results that can be obtained in the comparison between a detection and a point cloud, allowing to approximate better a result.

Computer vision is being applied to different areas, giving advances to the creation and interpretation the point clouds, such as Yaddaden et al. (2021) do, analyzing point clouds with local descriptors for applications that must act in real-time. This contribution is relevant due to it presents a way to analyze three-dimensional objects with low-computational cost. It allows to perform faster detection, making low-cost programs that can be applied to different purposes, but in more areas due to it could be supported by different kind of processors even with lower computational cost.

One example that contains image analysis and planning trajectories is the work of Rodriguez Baidez & Beltrá Fuerte (2022), which is dedicated to perform a trajectory to draw a sketch from a picture. In this project, a robot is used to take a photography, then the characteristics points of the image are taken to extracted and a path is generated through software. Once the points have been obtained, it creates a path with an algorithm that allows the robot to recreate the image in a bi-dimensional plane. It is a project that uses computer vision for a different purpose, but it combines the computer vision with planning trajectories in order to move a manipulator robot that draws a processed image.

As it has been said, AI is a reality and many of the mentioned projects contain it. Another example of this is the implementations that McMahon et al. (2022) show in their article, where try to improve the efficiency of Sampling-Based Motion Planners (SBMPs) to perform trajectories, planning and consider different aspects of this kind of planners for motion. It contributes to the generation of trajectories through neural networks, a tool that currently is common and improves the efficiency of projects.

Another task performed by a robot that analyzes a surface is the proposed project by Lizhe Qi & Sun (2023). In this project a robot equipped with a RGB-D camera extracts characteristics of the surface that it must clean and generates the necessary trajectories with neural network to perform the maintenance. It shows another way that robotics can be applied to, it uses a robot to clean surfaces extracting information with a camera that also computes the depth of the surface, having more information that allows the neural network to perform the task.

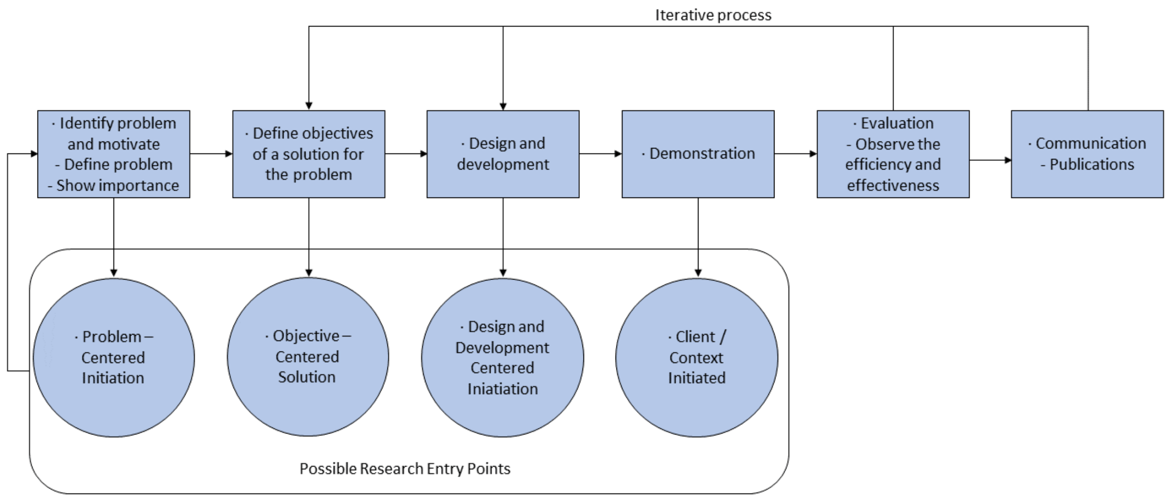
Regarding the all mentioned works, there is a trend that emerges of using neural networks for extracting characteristics of different objects. Moreover, the planning of trajectories is used in different purposes. Therefore, the idea of combining computer vision and path planning is something that has been studied and it is used currently. Efforts in this field focus on improving feature extraction and obtaining good results in trajectory planning to accomplish tasks that involve analyzing three-dimensional objects.

5 Methodology

In this section, it is described the used methodology to achieve the objectives of the project. The aim of this section is to provide a clear and concise description of the approach taken to carry out the work. The methodology consists of several steps, including data collection, data preprocessing, analysis, and evaluation.

Design Science Research Methodology (DSRM) is a research methodology used in the field of Information Systems to develop and evaluate Information Technologies (IT) artifacts. It involves a cyclical process of design, development, evaluation, and refinement, with a focus on creating practical solutions to real-world problems.

The methodology is characterized by its problem-solving focus, and its use of both qualitative and quantitative methods. The first step involves identifying a problem or opportunity and developing a set of design requirements. Next, the artifact is designed and developed, followed by an evaluation process that assesses the effectiveness of the solution in meeting the design requirements. Based on the evaluation results, the artifact is refined and improved, and the process repeats until the design requirements are met. The schema that is follow to accomplish the objectives through this methodology is the shown in the *Figure* 5.1.



Iterative process
· Identify problem and motivate - Define problem - Show importance
· Define objectives of a solution for the problem
· Design and development
· Demonstration
· Evaluation - Observe the efficiency and effectiveness
· Communication - Publications
· Problem – Centered Initiation
· Objective – Centered Solution
· Design and Development Centered Iniatiation
· Client / Context Initiated
Possible Research Entry Points

Figure 5.1: DSRM schema. (Adapted from Peffers et al., 2007)

Each step represents a stage of the process of an investigation that uses this methodology:

- Identify problem and motivation: In this initial stage, the problem and its context are identified and defined. The motivation behind the problem and the reasons for solving it are also determined.

- Define objectives for a solution for the problem: This step involves defining the specific objectives that the solution should achieve. These objectives should be clear, concise, and measurable.

- Design and development: Once the objectives are defined, the design and development of the solution can begin. This stage involves identifying possible solutions, selecting the best one, and designing and implementing it.

- Demonstration: After the solution is developed, it needs to be demonstrated to the stakeholders to ensure that it meets their requirements and objectives.

- Evaluation: This stage involves evaluating the effectiveness of the solution in achieving the defined objectives. It may involve collecting data and analyzing it to determine if the solution was successful.

- Communication: This stage involves communicating the results of the evaluation to stakeholders, such as clients or users. It may involve presenting reports or other forms of communication.

- Problem-centered initiation: This step involves starting the problem-solving process with a specific problem in mind, rather than a solution.

- Objective-centered solution: This step involves identifying the objectives that the solution should achieve, and then designing and implementing the solution to meet those objectives.

- Design and development center initiation: This step involves starting the problem-solving process by focusing on the design and development of a solution.

- Client/Context initiated: This step involves initiating the problem-solving process based on the needs and requirements of the client or the context in which the problem is occurring.

However, during the development of the project it was necessary to adapt the previous steps to how the project is performed, matching each step with the stages of the project, this relation is shown in the *Figure* 5.2.

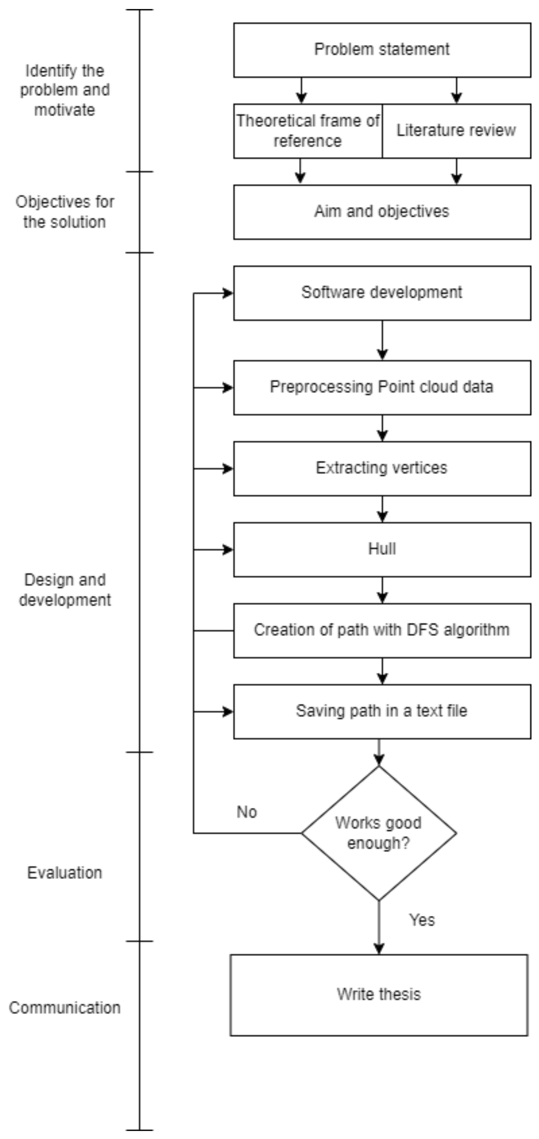

Identify the problem and motivate
Problem statement
Theoretical frame of reference
Literature review
Objectives for the solution
Aim and objectives
Software development
Preprocessing Point cloud data
Extracting vertices
Hull
Design and development
Creation of path with DFS algorithm
Saving path in a text file
No
Works good enough?
Evaluation
Yes
Write thesis
Communication

Figure 5.2: Flowchart of the development of the project.

As shown in the previous flowchart, the methodology followed in this project involves a systematic approach to problem-solving. The first step involves identifying the problem and conducting a thorough investigation using relevant documentation and previous works. Once the problem is identified, the objectives are defined, along with their aim. The design and development phase constitutes the largest part of the project, where all the necessary configurations and programming take place. During the evaluation stage, if the project does not work as planned, previous steps are checked until the problem is resolved, and the loop is repeated until the project works correctly. Finally, once the project is deemed successful, the next step is to write the thesis and demonstrate how the project works.

6 Development

Trough this chapter it will be exposed the different process that this project follows, as it can be seen in the *Figure* 5.2, it is divided into different sections. The parts that belong to the section of design and development will be explained along this section.

6.1 Software development

This project is composed of various scripts, each utilized in different stages of development. The parts into which the development is divided include: the assembly of a set of point clouds used for extracting key points, creating the path, and planning trajectories. These point cloud data has been extracted from a previous work of Abellan et al. (2014), which provides different pieces of geometrical figures suitable for the mentioned purposes.

Moreover, to perform the rest of steps of this project it was necessary to divided the tasks into scripts that are connected to operate in order. This steps will be shown in the flowchart of the *Figure* 6.1, but each step will be expanded in its own section.

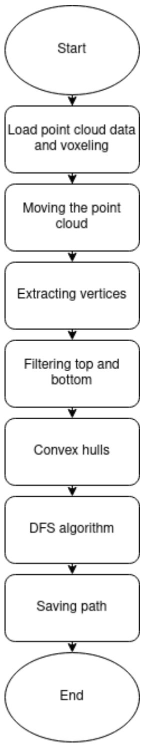


Figure 6.1: Flowchart of the application's process.

6.2 Point clouds

As it has been said, the data that this application uses is structured such as points, which are combined into point cloud data. These objects will be used to obtain the trajectories of the edges and contour of the shapes.

6.2.1 Loading of the data and voxeling

The representation of this step is shown in the *Figure* 6.5:

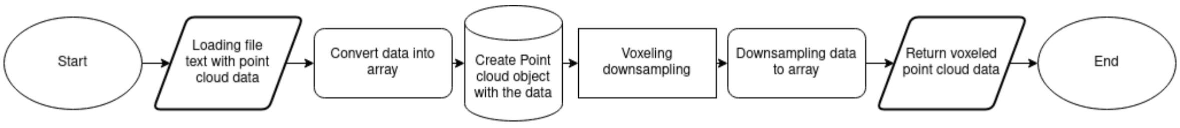

Start
Loading file text with point cloud data
Convert data into array
Create Point cloud object with the data
Voxeling downsampling
Downsampling data to array
Return voxeled point cloud data
End

Figure 6.2: Flowchart of loading data and downsampling stage.

First stage for the creation of the path is to read the point cloud. To do this task, the point clouds are loaded from the previous data set that contain the figures as a file text (*Figure* 6.3).

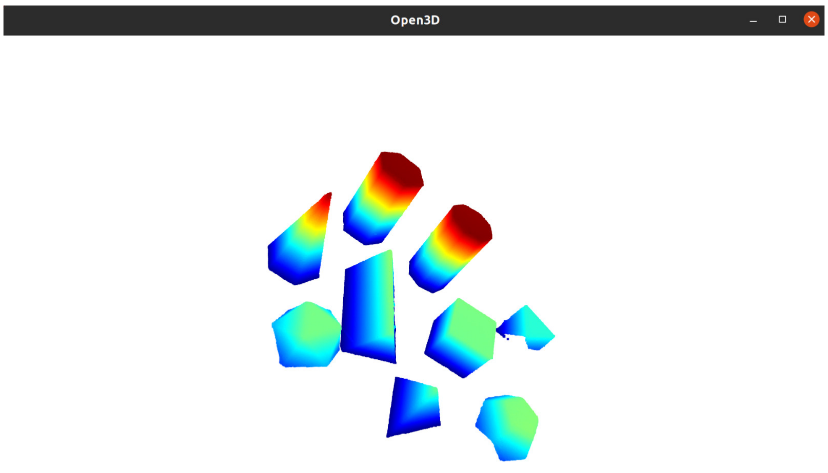


Figure 6.3: All the figures available of the mentioned data set.

The data is presented as such a text and it must be transformed into point cloud data to use the Open3D library, it is made using some functions that this library already contains.

Working with this data is computationally expensive and to develop the task of this project so many points are unnecessary and generate noise that can provoke to have wrong trajectories. To solve it, the first step is to reduce the number of points in this shapes, it is made using a voxel subsample, the size of the used box for the reduction depends on the shape and must be adapted to the task. Finally, the result of the point cloud is transformed into a numpy array to work in the following steps.

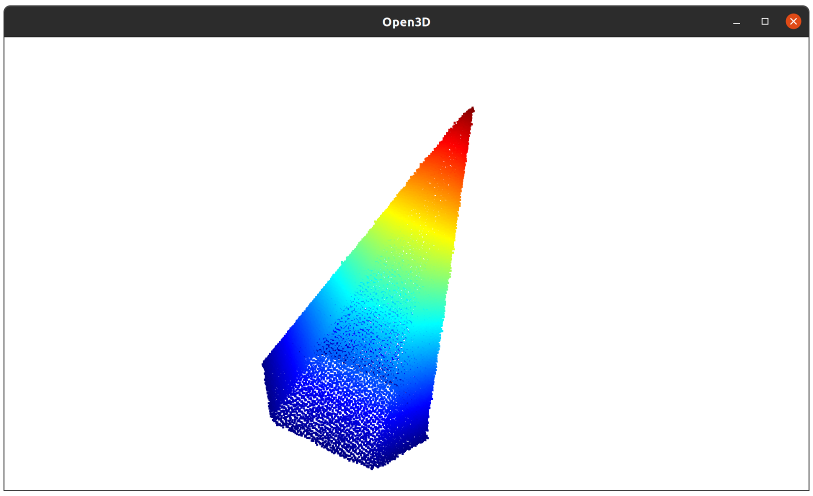


Figure 6.4: Point cloud of pyramid after voxel downsampling.

After downsampling the point cloud, it has reduced its number of points. In this case, the original point cloud had 37994 points and with a 0.4 of size of voxel it has been reduced to 32032, it has been deleted almost six 6000 of points.

After this reduction, it is still many points, they will be reduced afterwards, but until this moment the achieved reduction improves the efficiency of the intermediate steps.

6.3 Moving the point cloud

The data is located in different points of the space and to work properly in a simulation, it is necessary to move the point cloud to a place where it can be used to work. It will be done in three steps:

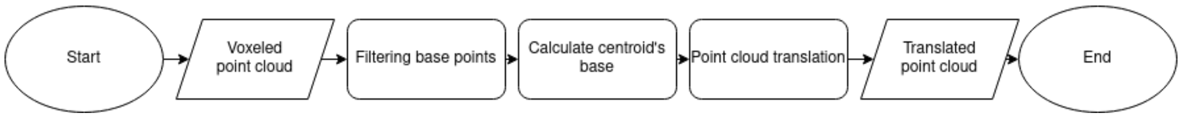


Figure 6.5: Flowchart of translation of the pointcloud.

1. Definition of the base: It has been established a threshold to differentiate the bottom of a figure from the rest of the figure. The points that are in this base are used in the following steps.

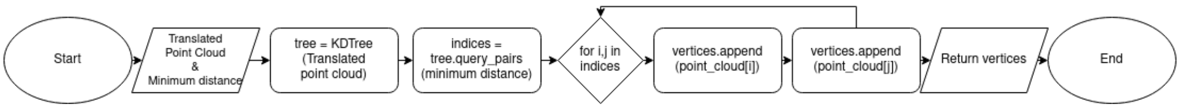
2. Calculation of the centroid of the base: Once the base is created, the next step is to calculate the middle of this base and move the whole structure to the origin of the system of reference, so the centre of the base will be approximately in the 0,0,z coordinate.

3. Moving in the z-axis: The piece is initially in the coordinate z that is established in the file text, to ensure the point cloud is in the coordinate 0,0,0 approximately, it is possible to move it doing a translation substracting the point that is on bottom, which means, the lowest value of z in the figure.

Once the point cloud is in the reference to work, it is possible to make different transformations to the structure that will be connected to achieve the final goal.

6.4 Extracting vertices

With the reduced point cloud data is difficult to work if it is not organized, for that reason, the data is transformed into a KDTRee. After that, with the function of Open3D query_pairs looks for pairs of points that are in a range of distance. Once the pairs have been created, the points are append into an array to extract the uniques vertices of this vector. This process can be shown in the *Figure* 6.6.



Start
Translated Point Cloud & Minimum distance
tree = KDTree (Translated point cloud)
indices = tree.query_pairs (minimum distance)
for i,j in indices
vertices.append (point_cloud[i])
vertices.append (point_cloud[j])
Return vertices
End

Figure 6.6: Flowchart of extracting vertices.

Such as result of the implementation of this flowchart is obtained the point cloud of the *Figure* 6.7.

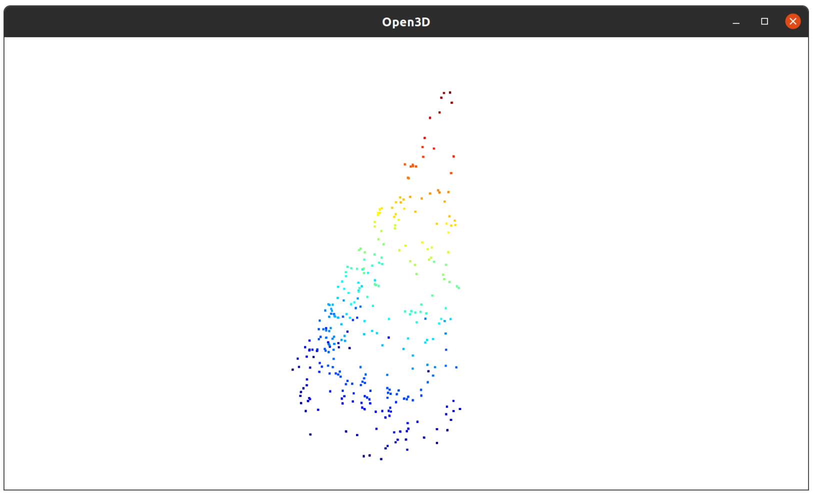


Figure 6.7: Pyramid after extracting the vertices key points.

As it is shown in the previous figure, there has been a great reduction of points, now the point cloud is compounded by 620 points. However, the pyramid has points that are unnecessary for the trajectories that can make sneaky paths that are not interesting for the project.

Next step is to differentiate between these points, which belongs to the top and the bottom of the structure to join them in order to obtain the corners and edges of the figure.

6.5 Filtering of bottom and top of the figure

During the planning of trajectories the most relevant points to generate the path are the vertices of the figure, which also defines the edges of the shape, but how it has been said in the previous section, the extraction of vertices is not completely successful. For that reason, it is important to look for this points that will define the structure of the figure.

The followed process to obtain them is shown in the *Figure* 6.8:

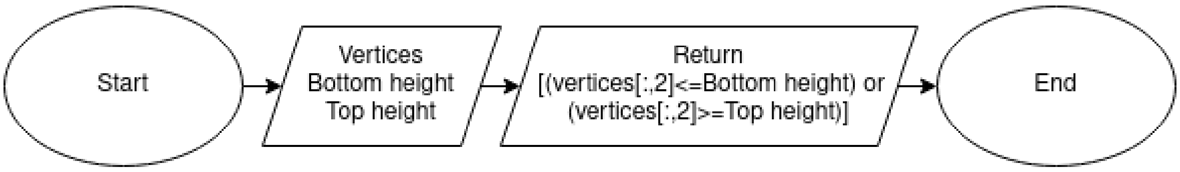

Start
Vertices
Bottom height
Top height
Return
[(vertices[:,2]<=Bottom height) or
(vertices[:,2]>=Top height)]
End

Figure 6.8: Flowchart of Filtering top and bottom.

To obtain it, two thresholds are stated in order to avoid the noise of the middle of the figure. with this threshold bottom and top are differentiated and the points are saved in an array to plan the trajectories later. The *Figure* 6.9, shows the plane YZ, where it is possible to differentiate both parts.

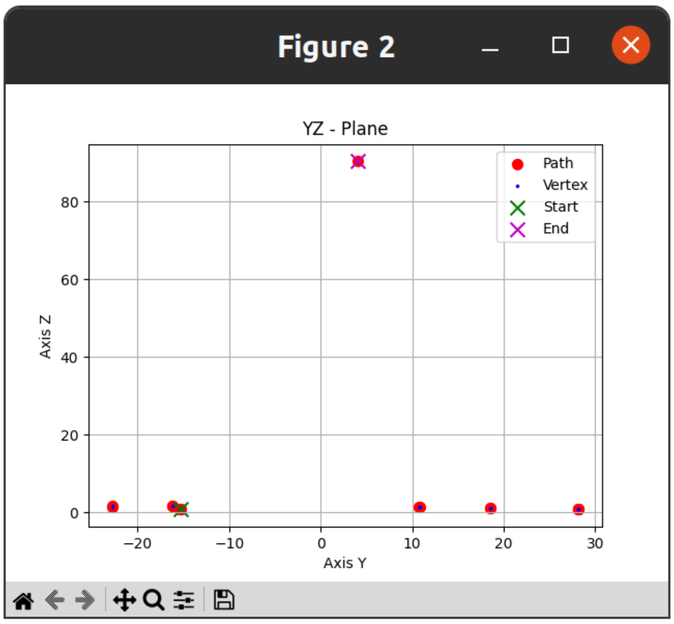


Figure 6.9: Plane YZ of the pyramid with only base and top.

6.6 Hull

Having both parts, in some figures it is possible to have points that conflicts with the top or bottom of the figure due to they are in the threshold. However, the important part is the contour of the shape, so it is possible to obtain it using the convex hull.

To obtain it, the used function is `ConvexHull` from `scipy.spatial` and the flowchart is the *Figure* 6.10:

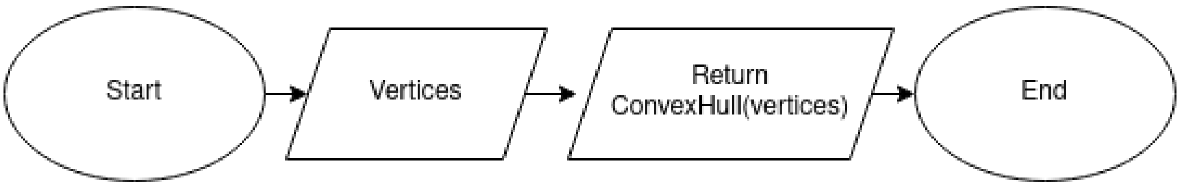
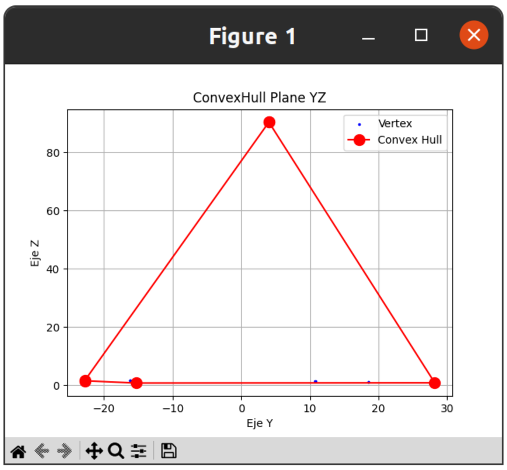
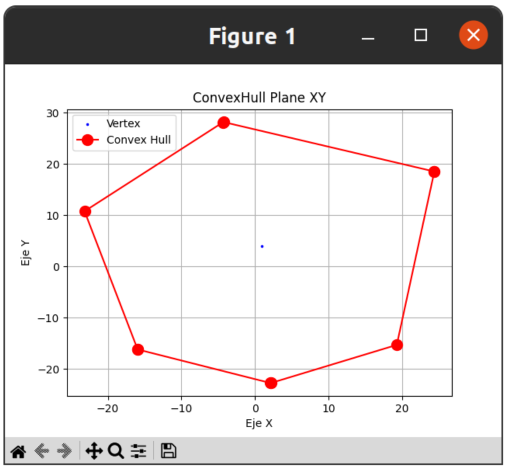


Figure 6.10: Flowchart of Obtaining Convex hull.

For the pyramid, it has been used twice times to obtain the hull of the YZ-Plane (*Figure* 6.11a), according to the *Figure* 6.9 and the XY-Plane's hull (*Figure* 6.11b).



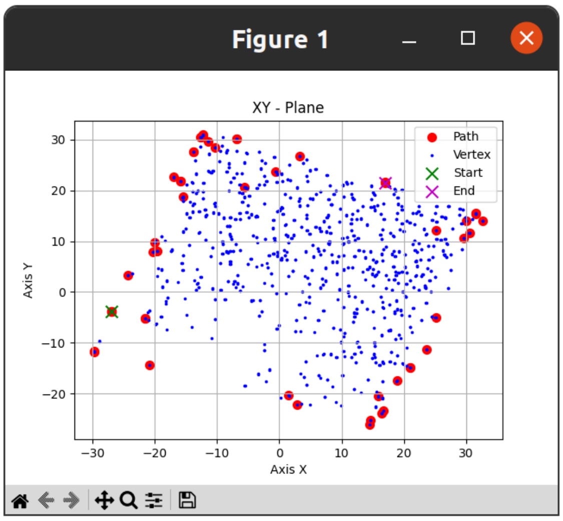
(a) Convex hull YZ of the pyramid.



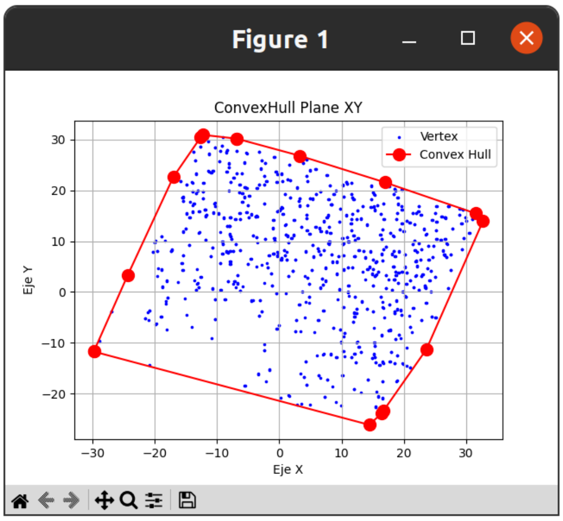
(b) Convex hull XY of the pyramid.

Figure 6.11: Visualization of convex hull from two points of view.

It has more relevance when some of the planes has noise,, in this case, the convex hull has more effect due to it discriminates and shows the contour without the noise, one example is the the plane XY of the cube (*Figure* 6.12a), which has noise in the top and the convex hull avoid it .



(a) Top of the cube (Plane XY).



(b) Convex hull XY of the cube.

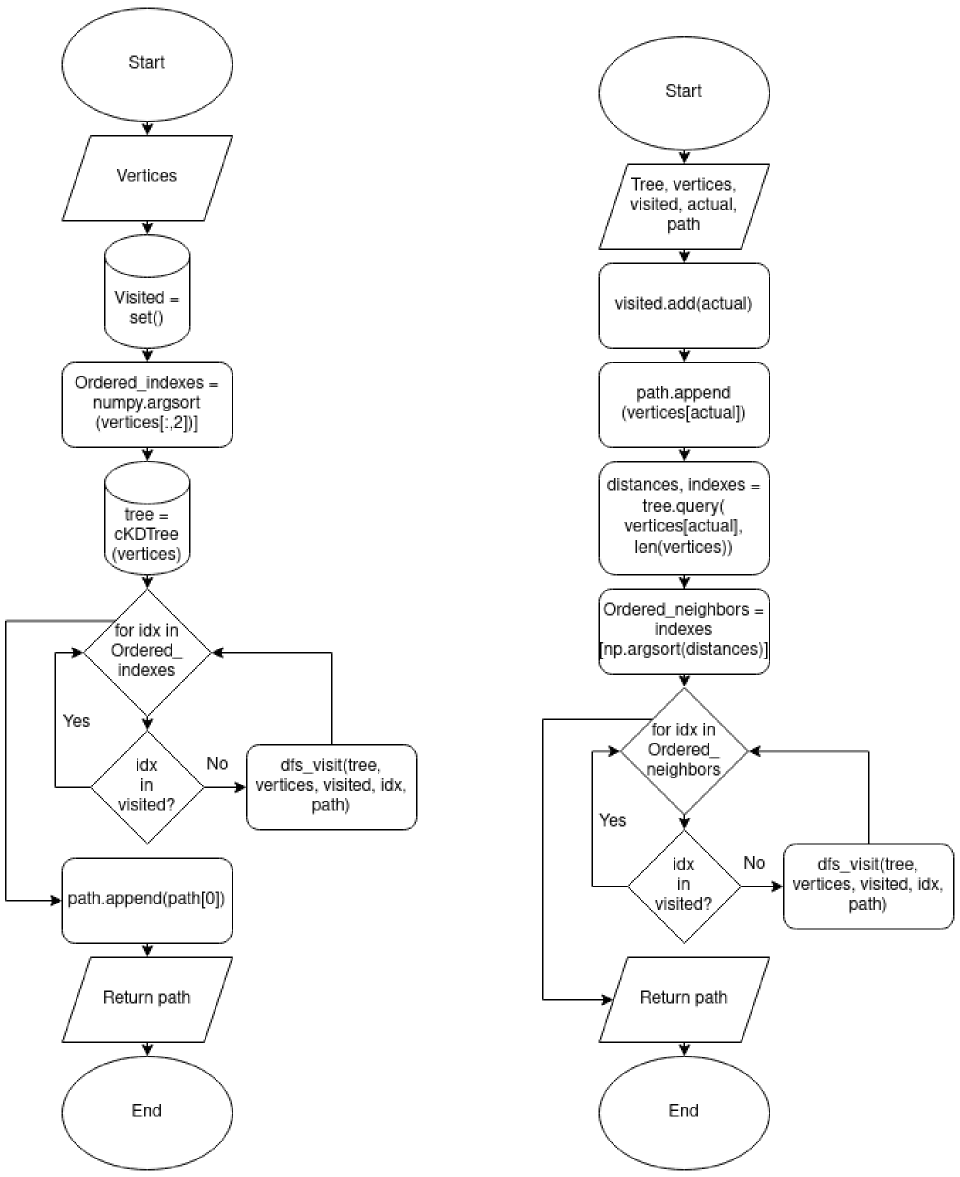
Figure 6.12: Visualization of convex hull of the cube.

Now, the path can be created, because the key points have been detected and joining them it is possible to create a route that follows them in a certain order.

6.7 DFS

To obtain the path, the used algorithm is DFS that goes trough a tree structure compounded by the vertices of the figure. It is divided into two functions in the development of the application.

Both flowcharts are presented in the *Figure* 6.13, `dfs` algorithm is respresented in the *Figure* 6.13a and the `dfs_visited` is the shown function of *Figure* 6.13b:


Start
Vertices
Visited = set()
Ordered_indexes = numpy.argsort (vertices[:,2])]
tree = cKDTree (vertices)
for idx in Ordered_ indexes
Yes
idx in visited?
No
dfs_visit(tree, vertices, visited, idx, path)
path.append(path[0])
Return path
End
Start
Tree, vertices, visited, actual, path
visited.add(actual)
path.append (vertices[actual])
distances, indexes = tree.query(vertices[actual], len(vertices))
Ordered_neighbors = indexes [np.argsort(distances)]
for idx in Ordered_ neighbors
Yes
idx in visited?
No
dfs_visit(tree, vertices, visited, idx, path)
Return path
End

(a) Flowchart of DFS algorithm. **(b)** Flowchart of DFS_visit algorithm.

Figure 6.13: Flowcharts of recursive algorithm dfs.

There is a first recursive function, which is used to explore the tree that is created, adding to a new list when a point is visited and its order. The used structure makes easier to find the nearest neighbors, using the the distance between each one. Moreover it is sorted by according this distances. This function calls the other one to create a path recursively and finally add the first point of the path to close de cycle.

The second function, that is called in the main creates an empty array to save the visited nodes. The vertices are organized based on their z coordinates to ensure that the route is created by layers using a KDTree structure to make the exploration more efficient. For each vertex the previous function is called to explore which points are reachable and more suitable from there. Finally, when the route has included all the points, the cycle is closed adding again the last point.

This algorithm ensures to visit all the points recreating the final shape and allows to change the order of following the points changing a few lines of the code.

6.8 Saving path

In order to use the path in following applications, once the path has been created it is exported to a text file, where the points are saved with the format XYZ coordinates and with the order of the path. It follows a basic flow to save it in a file text in the computer as it is shown in *Figure* 6.14:

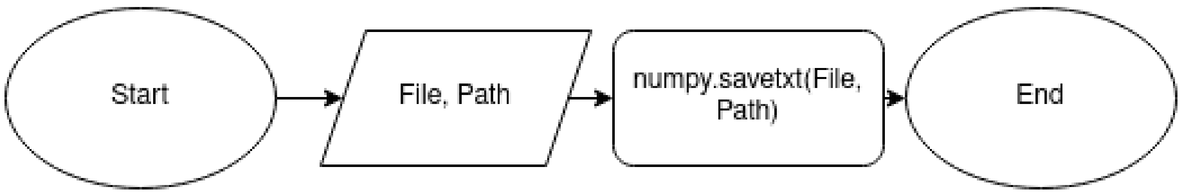


Figure 6.14: Flowchart of Saving the path.

The path will be saved in a text file, being possible to export it in future operations where it is needed or being read from another application to use the calculated path.

6.9 Additional steps

There are some steps that are not essential during the process of calculating the path or the planning trajectories. However, they can be an useful tool to visualize how the path is going to be performed or it the shape that is being analyzed is right.

6.9.1 Visualization of hulls

As it has been shown in previous figures such as 6.11a, 6.11b and 6.12b the contour of the figure is calculated with the convex hull. However, it shows some planes of the shape of the figure, so it has been implemented a three-dimensional version of this calculus (*Figure* **??**) to observe the whole structure.

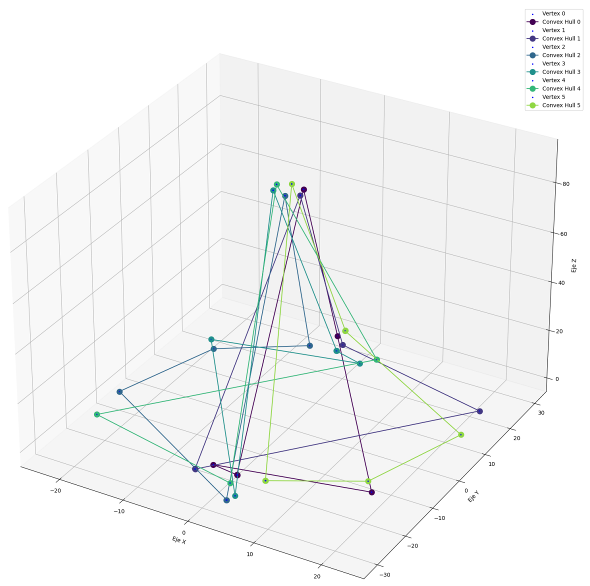


Figure 6.15: Visualization of the hull from different angles.

It is possible to distinguish the structure, it is not the real path, but it gives an idea about the relation that the points have among them an if this is an approximation of the figure that is wanted.

To obtain this figure it was used the matplotlib library and the code is the shown in the *listing* 6.1:

Listing 6.1: Visualize rotated convex hulls

```
# Function to visualize rotated convex hulls
def visualize_rotated_convex_hulls(paths):
    # Create a new figure
    fig = plt.figure()
    # Add a 3D subplot
    ax = fig.add_subplot(111, projection='3d')
    # Generate a color for each path
    colors = [plt.cm.viridis(i/len(paths)) for i in range(len(paths))]
    # For each path
    for i, path in enumerate(paths):
        # Scatter plot each vertex
        ax.scatter(path[:, 0], path[:, 1], path[:, 2], marker='.', color='b', s=10, label=f'Vertex {i}')
        # Line plot the path (convex hull)
        ax.plot(path[:, 0], path[:, 1], path[:, 2], marker='o', color=colors[i], linestyle='–', markersize=10, label=↩
            ↪ f'Convex Hull {i}')
    # Set labels for each axis
    ax.set_xlabel('X Axis')
    ax.set_ylabel('Y Axis')
    ax.set_zlabel('Z Axis')
    # Show the legend
    plt.legend()
    # Display the plot
    plt.show()
```

6.9.2 Visualization of paths

Moreover, the generated path can be observed before being used in other applications to check if it is the correct path, if it needs some improvement or if it was necessary to apply any change. To see if the route corresponds to the shape that is being studied, it is reflected in the *Figure* 6.16, where it generates lines joining the points in the order that they will be followed.

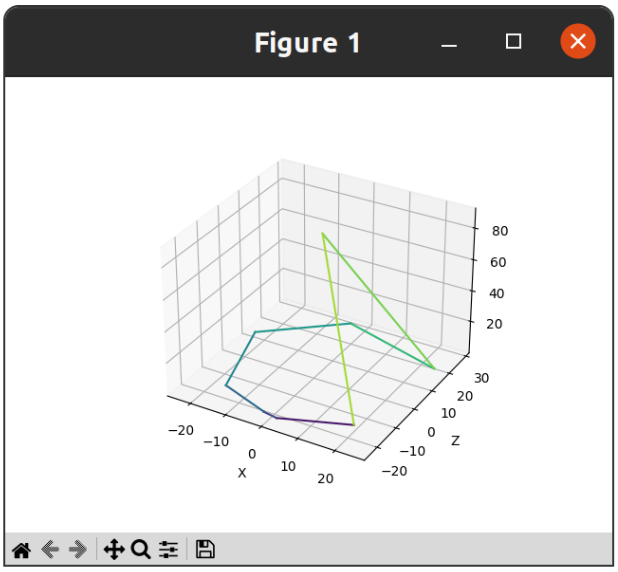


Figure 6.16: Visualization of the path created for the pyramid.

It is obtained with the code in the *listing* 6.2:

Listing 6.2: Plot DFS Path Points

```
# Function to plot the points of a DFS path
def plot_dfs_path_points(path):
    # Create a new figure
    fig = plt.figure()
    # Add a 3D subplot
    ax = fig.add_subplot(111, projection='3d')

    # Convert the path to a numpy array for easier handling
    path = np.array(path)

    # Create a list of colors based on the position of each point on the path
    colors = [plt.cm.viridis(i/len(path)) for i in range(len(path))]

    # Scatter plot the points with their corresponding color
    ax.scatter(path[:,0], path[:,1], path[:,2], c=colors)

    # Set labels for each axis
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    # Display the plot
    plt.show()
```

6.10 Planning of trajectories

Once the path has been obtained, the following step is to plan the order of the points that the robot will have to follow to complete the task. It is performed with the reconstruction of the path that the algorithm has created. During the performance of this path, the points of the closest neighbors were kept in order to reconstruct it.

This information is saved in a file where the points are written in the order that must be followed.

After that, the next program that uses this information, must perform the inverse kinematics in order to send the correct orders to the robot.

7 Results and discussion

In this section of the project, the results obtained from the research, experiments, or simulations are presented and analyzed. The discussion focuses on interpreting the findings and addressing problems.

Using different procedures, the most relevant characteristics of this project will be compared and the results of these experiments will also support the final implementation of this application.

Finally, the discussion section will show the strengths and weakness of this project, analyzing possible future implementations to improve the aspects that are weaker and enhance the features that work correctly.

7.1 Results

The results are presented using visual aids such as tables, graphs, and figures. Following the presentation of results, a comprehensive discussion is conducted, linking the findings to relevant literature, theories, and previous studies. This analysis highlights the implications of the results and uncovers novel contributions and suggestions for future research.

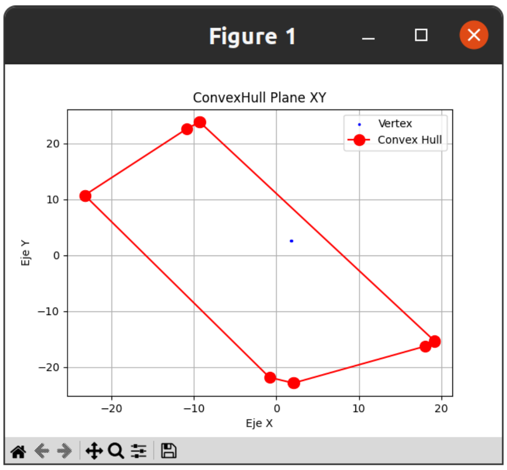
The section will be divided into two differentiate parts: Extraction of vertices, where it will be observed how the parameters affect the shape of the figure and the points that are taken; evaluation of the path generating, where different algorithms to plan the path are compared.

7.1.1 Extraction of vertices

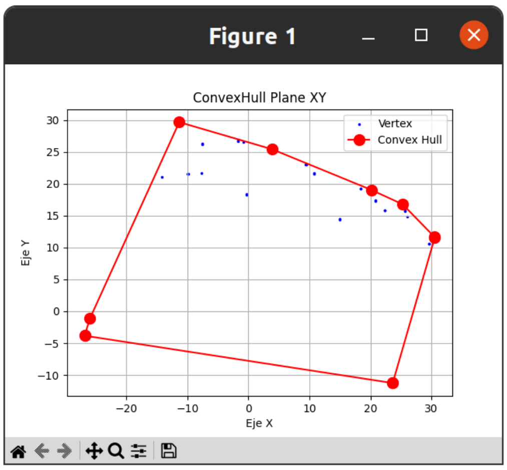
Regarding the extraction of key points, some aspects will be considered such as the reduction of the number of them and how they define the contour of the path that the robot must follow.

The process is compounded by different steps, which reduce the number of points to process less data. The first reduction is made by a downsampling, the most relevant element is the size of the voxel, it determines the radius that is compressed, so it is necessary to find a balance between reducing data and not lose information that can be important to preserve the figure's shape.

Testing different sizes and combining with following parameters, the most appropiatte voxel size to perform the downsampling is 0.15 because it reduces without losing relevant information. With minor numbers, there were points that generated noise in the trajectories and with bigger numbers some points did not appear, even changing other parameters, which affected to the shape (*Figures* 7.1a and 7.2c).



(a) Convex hull of the pyramid losing information.



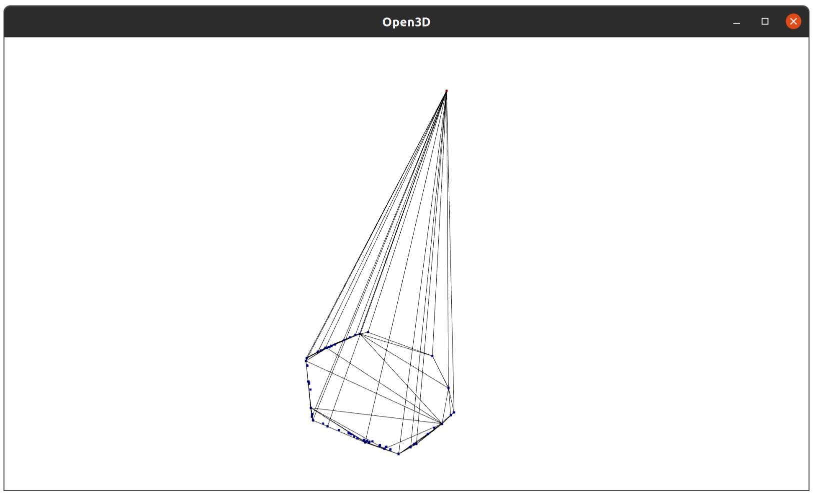
(b) Convex hull of the cube losing information.

Figure 7.1: Visualization of convex hull of two figures with a high voxel size.

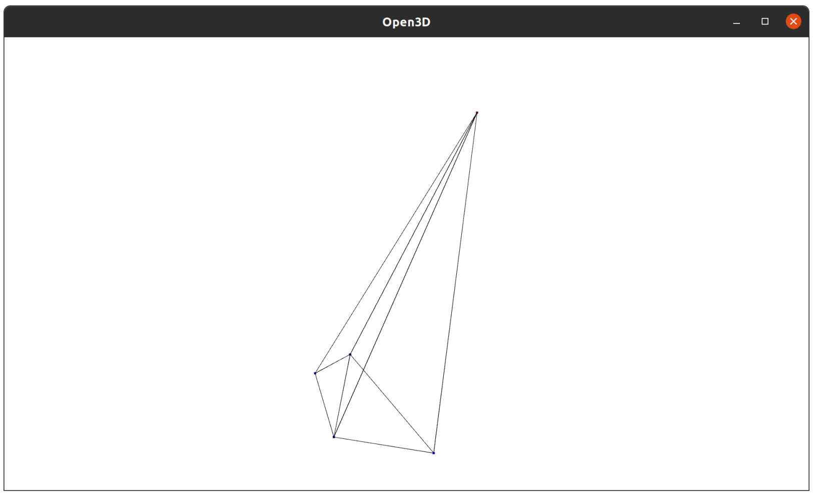
The result of changing the voxel size can be compared between the *Figure* 7.1a with the *Figure* 6.11b and *Figure* 7.2c with the *Figure* 6.12b. Both comparisons are perfomed changing only the voxel size, as it is possible to observe it generates great differences that provokes deformations in the contour of the shape.

Next step is identifying the vertices, in this stage, the important parameter is the minimun distance between a pair to create them. Variating this distance, most points are considered such as pair, so, depending the value of it, the return will be more or less accurate.

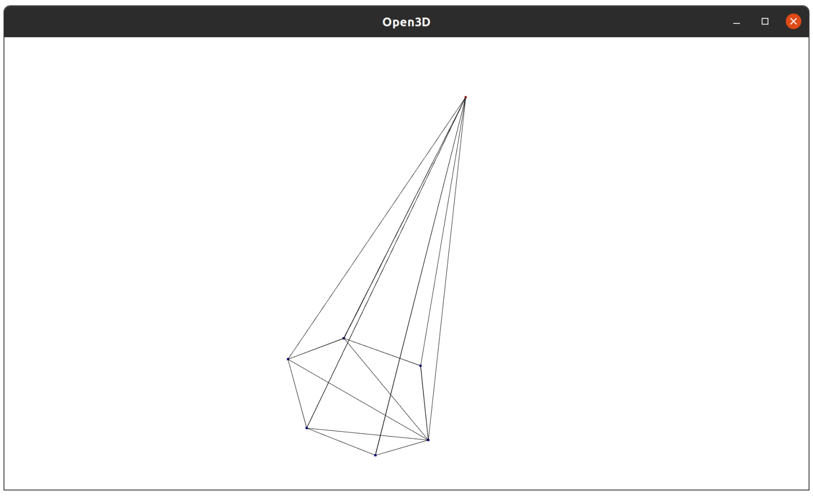
If the minimum distance is increased, the obtained shape will show many points, which will affected to the creation of the path (*Figure* 7.2a). On the other hand, if the number is smaller, it is possible that it does not find pairs or make deformations in the figure (*Figure* 7.2b).



(a) Result of minimum distance 0,23.



(b) Result of minimum distance 0,06.



(c) Result of minimum distance 0,09.

Figure 7.2: Visualization of convex hull of two figures with a high voxel size.

Next, it is employed a height-based filter that separates the structure into top and bottom components, or in the case of more complex shapes, generates intermediate zones (*Figure* 7.3). The determination of zones are essential to identify the specific regions of the shapes, so they must been stated depending on the figure to work with.

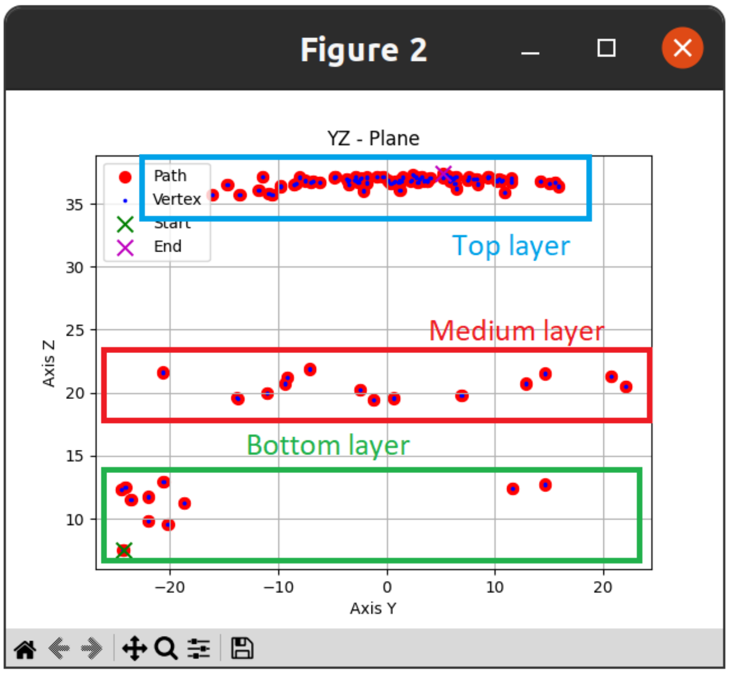


Figure 7.3: Different layers of a more complex shape.

The precision of these thresholds have an important impact on the result. Incorrect filtering of certain parts of the figure may lead to visible distortions, as illustrated in *Figure* 7.4.

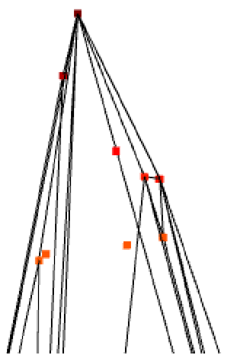
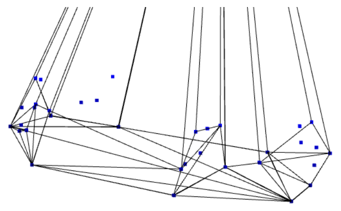


Figure 7.4: Deformation of the top of the pyramid.

Another problem that can appear if the threshold is not selected properly it that more points can appear to configure the path, introducing noise in the trajectory like in the *Figure* 7.5a or remove important points that provokes deformations in the structure, as it is shown in *Figure* 7.5b.


(a) High threshold that provokes noise for the trajectory.

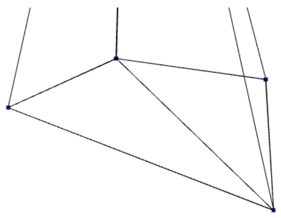

(b) Low threshold that loses information.

Figure 7.5: Comparison between high and low threshold.

The stated problems of the thresholds depend on the shape that is being used in this moment, so it is a parameter that must be adjusted before each work.

Last step is the convex hull, it does not remove points, but selects which one is consider such as contour. It has two compute a few number of points due to the previous steps has removed the most irrelevant. Nonetheless, this step just take these points that belong to the contour and has no parameters to change.

7.1.2 Creation of the path

Different algorithms to create the path have been tested in order to choose which one gets a lower time. The tested algorithms have been Dijkstra's, greedy and DFS. They have been run ten times and the *Table* 7.1 shows the mean of the ten iterations.

Number of points	Dijkstra (ms)	Greedy (ms)	DFS (ms)
18	1,916	1,057	0,528
25	2,512	2,072	0,739
39	9,359	4,986	1,778
63	22,995	13,195	1,956
87	47,748	24,534	3,493
122	83,24	47,967	5,243
186	193,93	110,473	9,338
293	481,996	273,175	19,592
362	725,863	422,069	28,313
423	1021,407	563,409	36,669
490	1382,859	735,364	46,845
566	1797,768	998,03	65,025
618	2148,252	1242,839	76,663

Table 7.1: Comparison of time of the different algorithms to create the path.

Moreover, to visualize the data it has been created a graphic (*Figure* 7.6) that shows the different algorithm's execution time and are compared with typical functions to measure the efficiency of an algorithm.

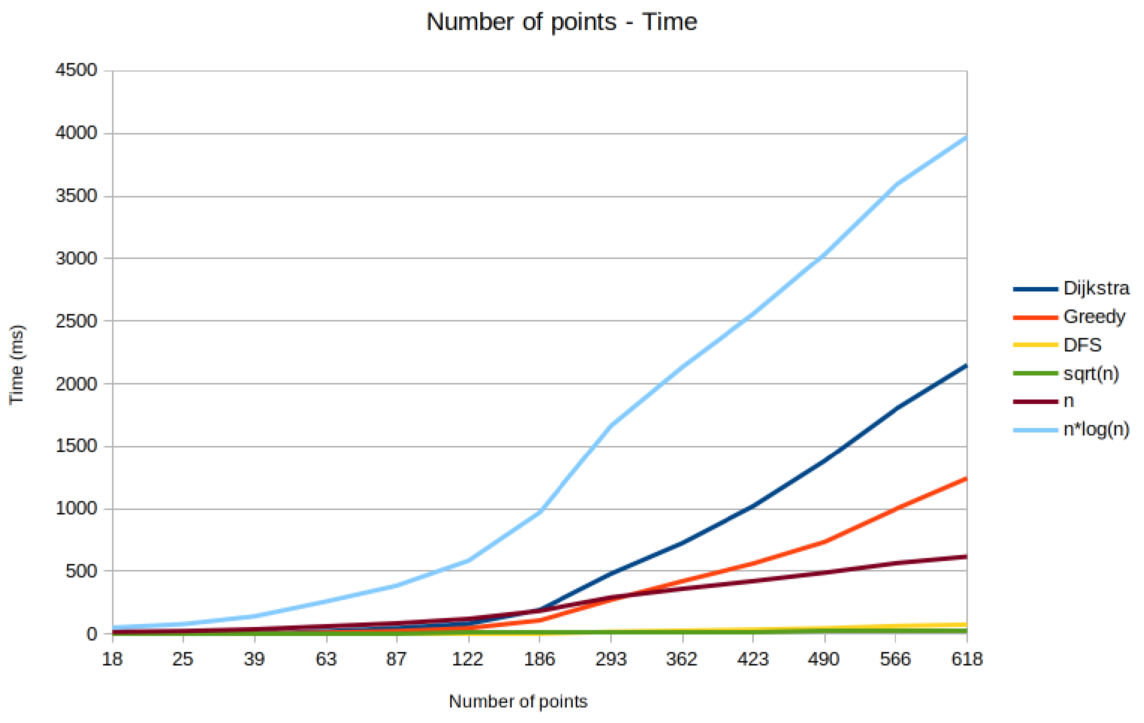


Figure 7.6: Visualization of the results and comparison with other function.

Looking at this data is possible to observe that the algorithm with best results is the DFS algorithm, which has approximately an efficiency of $\sqrt{n}$, being n the number of points. The next studied algorithm is the greedy, which increases the its time really fast, being less efficient than a linear function, it is because of the comparison of each point with the rest of the point cloud that have not been visited to obtain the shortest distance. Finally, the less efficient algorithm is the Dijkstra's algorithm, it needs to create a graph, update all the weights and go through it to obtain the cost of the path, which makes it inefficient to create a path, obtaining times of more than two seconds.

7.2 Discussion

Throughout the discussion, any limitations or weaknesses in the methodology, data, or interpretations will be adressed.

This project can be divided into two different stages: the use of point cloud data and the creation of algorithms. Both are critical in the execution of a trajectory and it is important to analyze in detail.

7.2.1 Point cloud data transformation

The aim of this project is to generate trajectories. However, it is not possible without previous transformations of the point cloud data. During the development of the project and the application's flow, there are some steps that requires to modify the data to work with it easier.

Regarding point cloud data transformation steps, all of the steps are essential and look for the highest efficiency to the following steps in order to reduce the number of points that the path generator has to consider and analyze, but conserving the information to represent the shape properly.

This objective is accomplished, it reduces the points of the data to manage it easily and keep the key points of the figure, which represent it. However, the program is not too generalizable, different parametres, variables, even some functions must be modified to adapt the program to the shapes.

It could be solved with more complex procedures or AI to obtain the most accurate extraction of points and adaptable to different figures or complex shapes.

7.2.2 Trajectory generation

As it has been shown in the results section, different algorithms have been tried to compare the results among them and finally, to choose the algorithm that makes the best path in the lowest time.

According to the algorithms to generate the paths, the most efficient as was expected is the DFS algorithm that uses the tree structure to get faster results finding the closest points. Moreover, it preserves the shape of the figure and creates a path that would work.

The generation of paths has a huge dependence of the previous step, where it is very important to filter correctly. However, the implementation of this path generation works as it should do and it is easily modifiable for the different purposes.

8 Conclusions

The aim of this project is to develop an application that generates paths from three-dimensional point cloud data. The project uses data reduction techniques to eliminate irrelevant information and detect vertices to define the structure's shape. Additionally, it employs routing algorithms to create paths, which preserves the contour of the figure, saving the points of the path in the order that must be followed in a text file to use.

The obtained shapes correspond to the point cloud objects utilized, accomplishing their intended function. However, as it stands, the program is not fully generalizable as it relies on several variables that must be adjusted depending on the specific piece. Despite this, it operates efficiently, and the potential for standardization represents a direction for future expansion letting a scope for enhancing the performance through the adoption of varied strategies or technologies, such as AI.

Regarding the implementation of path creation, the achieved efficiency accomplish the expectations, serving as a testament to the proper implementation of the algorithms. The results indicate that the chosen algorithms are effectively generating paths achieving the expected goals. The efficiency of the path creation process ensures that the generated trajectories meet the requirements within a reasonable timeframe.

In conclusion, the project has successfully achieved its initial objectives, demonstrating the viable application of three-dimensional data for generating accurate trajectories. This successful development has resulted in the creation of a solid foundation that can be built upon in the development of future projects.

9 Future work

One potential extension for this project could be to explore the use of AI to optimize the path planning process. It would be interesting to specifically investigate the use of reinforcement learning or deep learning techniques. These techniques can automatically generate optimal trajectories for the robot based on its environment and task requirements. This could involve training a neural network or another machine learning model on a dataset of existing trajectories. The model could then be used to predict the optimal path for new objects.

Additionally, neural networks could be employed for the identification of geometry or specific figures. This would help in selecting faces with higher resolution and obtaining relevant points from known figures. This method could improve the path creation for the shapes and make it more accurate.

Another potential extension could explore the use of real-time feedback to adjust the robot's trajectory as it moves. This could involve the use of sensors or cameras to monitor the robot's progress and make adjustments to the trajectory in real-time. This could be based on changes in the environment or the presence of unexpected obstacles. This improvement could enhance the accuracy and efficiency of the robot's movements, especially in dynamic or unpredictable environments.

This project could also be used for VR, applying the calculations of trajectories and path planning to visualize how robots move in a virtual world. This would be a contemporary adaptation of the project, given the increasing use of VR in various fields.

In summary, this project demonstrates considerable potential for further advancement in various fields, from the use of AI and machine learning for optimization to real-time adjustments of robot trajectories. Standardizing the program could represent a significant advancement and could be a valuable resource for future tasks.

References

Aarno, D., Lingelbach, F., & Kragid, D. (2005, 08). Constrained path planning and task-consistent path adaptation for mobile manipulators. In (p. 268 - 273). doi: 10.1109/ICAR.2005.1507423

Abd Algfoor, Z., Sunar, M. S., & Kolivand, H. (2015, 04). A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology*, *2015*, 1-11. doi: 10.1155/2015/736138

Abdollahi, R., Amjad Seyedi, S., & Reza Noorimehr, M. (2020). Asymmetric semi-nonnegative matrix factorization for directed graph clustering. In *2020 10th international conference on computer and knowledge engineering (iccke)* (p. 323-328). Retrieved from `https://ieeexplore.ieee.org/document/9303649` doi: 10.1109/ICCKE50421.2020.9303649

Abellan, A., Jaboyedoff, M., Tomás, R., & Riquelme, A. (2014, 01). *3d point clouds of simple figures.* Retrieved from `https://www.researchgate.net/publication/358461509_3D_point_clouds_of_simple_figures` doi: 10.13140/RG.2.2.21511.47528

Abt, F., Heider, A., Weber, R., Graf, T., Blug, A., Carl, D., … Tetzlaff, R. (2011, 12). Camera based closed loop control for partial penetration welding of overlap joints. *Physics Procedia*, *12*, 730-738. doi: 10.1016/j.phpro.2011.03.091

Adolfsson, J., & Schmidt, B. (2001). *Introduction to manipulator kinematics.*

Alhusin Alkhdur, A., Givehchi, M., & Wang, L. (2012, 06). Interfacing image processing with robotic sketching..

Barrientos, A., Cruz, A., Peñín, L., & Balaguer, C. (2007). *Fundamentos de robótica.* McGraw-Hill. Retrieved from `https://books.google.es/books?id=ArEMPAAACAAJ`

Bone, G. M., Lambert, A., & Edwards, M. (2008). Automated modeling and robotic grasping of unknown three-dimensional objects. In *2008 ieee international conference on robotics and automation* (p. 292-298). doi: 10.1109/ROBOT.2008.4543223

Borovic, B., Liu, A.-q., Popa, D., Cai, H., & Lewis, F. (2005, 10). Open-loop versus closed-loop control of mems devices: Choices and issues. *J. Micromech. Microeng*, *15*, 1917-1924. doi: 10.1088/0960-1317/15/10/018

Casado-Vara, R., Chamoso, P., De La Prieta, F., Prieto, J., & Corchado Rodríguez, J. (2019, 01). *Non-linear adaptive closed-loop control system for improved efficiency in iot-blockchain management* (Vol. 49). doi: 10.1016/j.inffus.2018.12.007

Cheng, S., Chen, X., He, X., Liu, Z., & Bai, X. (2021). PRA-Net: Point relation-aware network for 3d point cloud analysis. Retrieved from `https://arxiv.org/pdf/2112.04903.pdf`

Chirikjian, G. S. (1992). Theory and applications of hyper-redundant robotic manipulators. *ProQuest Dissertations and Theses*, 211. Retrieved from `https://www.proquest.com/dissertations-theses/theory-applications-hyper-redundant-robotic/docview/304032414/se-2` (Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Última actualización - 2023-02-23)

Civicioglu, P. (2013). Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation*, *219*(15), 8121-8144. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0096300313001380` doi: https://doi.org/10.1016/j.amc.2013.02.017

Cornejo, J., Denegri, E., Vasquez, K., & Ramos, O. E. (2018). Real-time joystick teleoperation of the sawyer robot using a numerical approach. *2018 IEEE ANDESCON*, 1-3.

Craig, J. J. (2005). *Introduction to robotics: Mechanics and control* (3rd ed.). Pearson.

Dasari, S., Ebert, F., Tian, S., Nair, S., Bucher, B., Schmeckpeper, K., ... Finn, C. (2019). Robonet: Large-scale multi-robot learning. *CoRR*, *abs/1910.11215*. Retrieved from `http://arxiv.org/abs/1910.11215`

Goel, R., & Gupta, P. (2020). Robotics and industry 4.0. In A. Nayyar & A. Kumar (Eds.), *A roadmap to industry 4.0: Smart production, sharp business and sustainable development* (pp. 157–169). Cham: Springer International Publishing. Retrieved from `https://doi.org/10.1007/978-3-030-14544-6_9` doi: 10.1007/978-3-030-14544-6_9

Hacinecipoglu, A., Konukseven, E., & Koku, A. (2020, 01). Pose invariant people detection in point clouds for mobile robots. *International Journal of Mechanical Engineering and Robotics Research*, 709-715. doi: 10.18178/ijmerr.9.5.709-715

Hägele, M., Nilsson, K., Pires, J. N., & Bischoff, R. (2016). Industrial robotics. In B. Siciliano & O. Khatib (Eds.), *Springer handbook of robotics* (pp. 1385–1422). Cham: Springer International Publishing. Retrieved from `https://doi.org/10.1007/978-3-319-32552-1_54` doi: 10.1007/978-3-319-32552-1_54

Human Robotics. (2023). *Control de robots: Theme 2.* (Grado en Ingeniería Robótica)

International Organization for Standardization. (2021). *Iso 8373:2021.* Retrieved from `https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-3:v1:en` (Robots and robotic devices – Vocabulary (6th ed.))

Jaquier, N. (2016). Eeg-based control of a 7-dof upper-limb prosthesis.

Kakani, V., Nguyen, V. H., Kumar, B. P., Kim, H., & Pasupuleti, V. R. (2020). A critical review on computer vision and artificial intelligence in food industry. *Journal of Agriculture and Food Research*, *2*, 100033. Retrieved from `https://www.sciencedirect.com/science/article/pii/S2666154320300144` doi: https://doi.org/10.1016/j.jafr.2020.100033

Kickert, W., & Mamdani, E. (1993). Analysis of a fuzzy logic controller. In *Readings in fuzzy sets for intelligent systems* (pp. 290–297). Elsevier.

Lizhe Qi, Z. H. D. D. W. J., Zhongxue Gan, & Sun, Y. (2023, 04). Cleaning of object surfaces based on deep learning: a method for generating manipulator trajectories using rgb-d semantic segmentation.

Lynch, K. M., & Park, F. C. (2018, May 21). *9.1 and 9.2 point-to-point trajectories (part 1 of 2).* Cambridge University Press. Retrieved from `https://modernrobotics.northwestern.edu/nu-gm-book-resource/9-1-and-9-2-point-to-point-trajectories-part-1-of-2/`

McMahon, T., Sivaramakrishnan, A., Granados, E., & Bekris, K. E. (2022). A survey on the integration of machine learning with sampling-based motion planning. *Foundations and Trends® in Robotics*, *9*(4), 266-327. Retrieved from `http://dx.doi.org/10.1561/2300000063` doi: 10.1561/2300000063

Mousavian, A., Eppner, C., & Fox, D. (2019, October). 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the ieee/cvf international conference on computer vision (iccv).*

of Robotics, I. F. (2022). *Industrial robots.* Retrieved from `https://ifr.org/industrial-robots`

Open3D. (2018). *Tutorial.* Retrieved from `http://www.open3d.org/docs/0.11.1/tutorial/geometry/pointcloud.html` (Chapter: Geometry/Point Cloud)

Point Cloud Library. (2023). *Getting started / basic structures.* Retrieved from `https://pcl.readthedocs.io/projects/tutorials/en/master/basic_structures.html#basic-structures`

Pérez Ruiz, A., Aroca Trujillo, J., & Rodriguez, R. (2017, 12). Generation and control of basic geometric trajectories for a robot manipulator using compactrio®. *Journal of Robotics*, *2017*, 1-11. doi: 10.1155/2017/7508787

Robotics, W. (2022). *Executive summary world robotics 2022 industrial robots.* Retrieved from `https://ifr.org/img/worldrobotics/Executive_Summary_WR_Industrial_Robots_2022.pdf` (Accessed: May 4, 2023)

Rodriguez Baidez, E. M., & Beltrá Fuerte, J. (2022). *Robot path planning using 2d image processing in a drawing application.*

Rosales, E., Gan, Q., & Gan, J. (2002, 01). Forward and inverse kinematics models for a 5-dof pioneer 2 robot arm.

Schmidt, D. C., & Druffel, L. E. (1976). A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. *Journal of the ACM (JACM)*, *23*, 433 - 445.

Sharma, M., Luthra, S., Joshi, S., Kumar, A., & Jain, A. (2023). Green logistics driven circular practices adoption in industry 4.0 era: A moderating effect of institution pressure and supply chain flexibility. *Journal of Cleaner Production*, *383*, 135284. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0959652622048582` doi: https://doi.org/10.1016/j.jclepro.2022.135284

Sogo, T., Ishiguro, H., & Ishida, T. (2001, 12). Mobile robot navigation by a distributed vision system. *New Generation Computing*, *19*, 121-137. doi: 10.1007/BF03037250

Wilson, M. (2006). Robot modeling and control. *Industrial Robot: An International Journal*, *33*(5), 403. Retrieved from `https://doi.org/10.1108/ir.2006.33.5.403.1` doi: 10.1108/ir.2006.33.5.403.1

Yaddaden, Y., Daniel, S., & Laurendeau, D. (2021). Online point cloud object recognition system using local descriptors for real-time applications. In *Visigrapp (5: Visapp)* (pp. 301–308). Retrieved from `https://www.researchgate.net/publication/350102672_Online_Point_Cloud_Object_Recognition_System_Using_Local_Descriptors_for_Real-Time_Applications`

Zhou, Q.-Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.

Lista de Acrónimos y Abreviaturas

AI	Artificial Intelligence.
API	Application Programming Interface.
DFS	Depth-First Search.
DOF	Degrees Of Freedom.
DSRM	Design Science Research Methodology.
FSA	Finite State Automata.
IFR	International Federation of Robotics.
IoT	Internet of Things.
ISO	International Organization for Standaridization.
IT	Information Technologies.
MI	Mutual Information.
NCC	Normalized Cross Correlation.
OS	Operative System.
RGB-D	Red, Green, Blue and Depth.
ROS	Robot Operating System.
SBMPs	Sampling-Based Motion Planners.
SIFT	Scale-Invariant Feature Transform.
SSD	Sum of Squared Differences.
SURF	Speeded Up Robust Features.
VR	Virtual Reality.