UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

DOTTORATO DI RICERCA IN INGEGNERIA MECCANICA E INDUSTRIALE

ING-IND/13 - APPLIED MECHANICS
XXXIII CICLO

Towards a Conventional and Multi-directional Printing Robotic Cell

Ph.D. Candidate:
Kevin Castelli
Castelli Kevin

Ph.D. Tutor:
Prof. Giovanni Legnani [illegible]

Ph.D. Advisor:
Prof. Ing. Hermes Giberti [illegible]

Ph.D. Coordinator:
Prof.ssa Laura Eleonora Depero

To fail to speak to a man who is capable of benefiting is to let a man go to waste.
To speak to a man who is incapable of benefiting is to let one's words go to waste.
A wise man lets neither men nor words go to waste.
Confucius

Acknowledgments

I'd like to thank Prof. Legnani and Prof.Giberti for the time and support during this research activity.
I'd like also to express my very profound gratitude to my family for the support in these years. This accomplishment would not have been possible without them.
Many thanks to you all.

November 2020

Contents

List of Figures

List of Tables

Abstract Italiano

Additive Manufacturing (AM) rappresenta una tecnologia cruciale nel settore industriale contemporaneo in quanto offre agli imprenditori la possibilità di generare internamente prototipi e pezzi unici su richiesta. Si ritiene infatti che sarà uno degli aspetti pivotali della quarta rivoluzione industriale (Industry 4.0).
Le tecnologie AM sono in grado di processare diversi materiali (metalli, polimeri e ceramiche) sfruttando i principi fisici più consoni a tale scopo. Nonostante le differenze che ne derivano, alcune caratteristiche sono condivise tra queste tecnologie, dove la principale è il metodo di costruzione. Infatti, sono tutte basate su un approccio a layer, il pezzo finale è il risultato della sovrapposizione di sottili strati planari di materiale. Ci sono inoltre elementi che, d'altro canto, limitano ciò che può essere ottenuto dalle tecniche utilizzate nel AM in termini di geometrie, estetica e proprietà meccaniche.
Pertanto, questa tesi propone nuovi approcci all'utilizzo della stampa 3D andando ad agire sugli algoritmi utilizzati dallo slicer (CAM) per definire le traiettorie e utilizzando robot industriali per generare il pezzo. Lo stato dell'arte dell'AM funge da punto di partenza della tesi mettendo in luce gli aspetti relativi ai processi, algoritmi utilizzati dai principali CAM (comprensivi di quelli presenti solamente in letteratura), macchine utilizzate e limitazioni che si possono trovare nel campo della stampa 3D. Le informazioni sono utilizzate per realizzare un setup (per integrazione) adatto a condurre la sperimentazione sui metodi e le strategie identificate; la cella finale è costituita da un utensile (estrusore FDM sviluppato per facilitarne l'integrazione con gli altri dispositivi nella cella) e da un manipolatore industriale a 6 gradi di libertà in grado di definire la posizione del dispositivo d'estremità nello spazio 3D completo (pertanto permettere di utilizzare metodi costruttivi con layer planari o definiti come superfici generiche che possono essere paralleli o con orientamento variabile). Cinque robot commerciali sono stati convertiti in stampanti 3D per definire le strategie migliori per controllarne il movimento al fine di rendere l'utilizzo il più accessibile possibile ad un personale non necessariamente qualificato ad intervenire sull'architettura della cella. Il tema successivo è la generazione del percorso e la relativa assegnazione della legge di moto; il codice sviluppato prende in ingresso il file CAD (STL) del pezzo per generare il percorso che l'utensile dovrà seguire tenendo in considerazione il processo stesso, possibili collisioni all'interno della cella e altri vincoli che si possono incontrare durante la lavorazione. L'algoritmo utilizzato per la valutazione della collisione è sia usato a tale scopo sia per calcolare l'intersezione tra il CAD e la generica superfice di deposizione (purchè am-

metta una rappresentazione tramite mesh triangolare). Test sperimentali sulle possibili strategie di deposizione presenti in letteratura sono condotte per convalidare il setup e il software generato. I risultati, in combinazione con lo stato dell'arte, sono utilizzati per identificare nuove strategie di realizzazione dei pezzi tramite AM. Il mondo naturale è stato utilizzato come ulteriore fonte di ispirazione per la definizione di tali soluzioni (bioinspired). Una prima soluzione, derivante dal modo in cui viene realizzato il nido da parte di alcuni volatili, è trattata mostrandone le principali caratteristiche seguita dalle relative prove finalizzate a validarne i principi base. Ulteriori strategie sono teorizzare sottolineando le ripercussioni sulla definizione della cella.
Infine, i concetti presentati sono trasferiti a delle applicazioni industriali per sottolinearne gli aspetti pratici.

Parole chiave: Robotica, Collisione, FDM, Slicing, stampa 3D multi-direzionale, nuove metodologie di stampa 3D, bioinspired

Abstract

Additive Manufacturing (AM) has a crucial role in contemporary industrial field because allows entrepreneurs the possibility to obtain prototypes and unique items directly in-house and on-demand.
AM technologies are able to process different materials (metals, polymers and ceramics) exploiting distinct physical principles. Despite the differences that arise, common elements are shared among them with the main one being the building method. They are all indeed layer based, meaning that the final object is the results of the superimposition of thin planar (2D) film of material.
There are, moreover, elements that limit what can be achieved by the current AM techniques in terms of mechanical properties, geometries and aesthetics.
Therefore, this thesis proposes novel approaches to the use of AM, acting on the CAM software algorithms used to define the trajectory and by adopting robotic systems to generate the part. More in detail, the current trends in the AM field serve as starting point for the dissertation by shedding a light on the processes, CAM algorithms, devices and limitations found in AM. The information is then used to integrate a suitable setup to carry out experimentation on the subject; the system is constituted by a tool (an FDM extruder integrable with the other devices that can be found in the cell) and a 6 degrees of freedom industrial manipulator able to define the trajectory of the end effector in the complete 3D space (both generic surface and planar layer deposition). Five commercial robots have been converted into 3D printers to define the most suitable control strategy to master the motion of the end effector with the objective to simplify the use of the final machine by the operator. The generation of the path and relative motion law is the following subject; the code is developed to take the CAD of the part and generate the toolpath accounting for the process itself, collision within the cell and additional working constraints. The algorithm adopted for checking the collision is the same adapted to compute the intersection between the model and the generic surface (provided that it admits a triangular mesh representation). Practical test on the current frontiers of 3D printing are then carried out. Their results, in combination with the state of art, are used to identify possible new building strategies. As an additional source of inspiration, the natural has been used (bioinspired). The first solution identified, derived by how some birds form their nest, is described and its preliminary testing is done to validate its principles. Additional building strategies are theorized together with the effect on the AM cell.

Lastly, concepts introduced are applied to actual industrial application to remark their practical function.

Key world: Robotics, Collision, FDM, Slicing, Multi-directional printing, Novel building strategies, bioinspired

Introduction

dditive Manufacturing (AM) has a pivotal role in contemporary industrial revolution (Industry 4.0) that aims at profitability, efficiency, performance and customization by merging the digital and physical world. AM indeed tries to allow entrepreneurs and end-users the possibility to obtain unique or small batch size parts on-demand, without outsourcing and with competitive leads time by converting the digital model of the geometry directly in the corresponding physical object.
Today's market is flooded with CNC machines able to produce components in different material (metals, plastics, ceramics and cements) using different working principles (AM technologies) that best suits the feedstock.
Despite the intrinsic differences that is implied by this, all of these technologies share a common ground that is the way the final part is put together (layer-by-layer approach). The majority of the cases sees the superimposition of thin planar (2D) films, corroborated by the hegemony of CAM software able to generate the toolpath for this specific method. The layer-by-layer approach is the mimicry of the subtractive counterpart (milling) using 3-axis machines.
In recent years, researchers have been investigating to transfer to AM the analogous improvements (surface quality and feasible geometries) obtained by the introduction of 5-axis machines in the subtractive manufacturing field. The resulting methods exploit the possibility to define in space the TCP to shift from flat parallel films to planes with variable relative orientation or using generic surfaces as primary units; this impacts the final surface quality and the mechanical properties.
Although improving the quality of the process, aspects of the process still limit AM as well as restrain the definition of new deploying scenarios. These shortcomings interest the mechanical properties (i.e. anisotropy) and the geometrical tolerances that can be obtained in the printed part.
This thesis inquires novel approaches in the use of AM acting on the CAM software and exploiting industrial robots for the realization of the components. Robotic systems with 6 Degrees of freedom (Dofs) allows to define the pose of the end effector in the complete 3D space (position and orientation of the coordinate system associated with this body) as well as offer larger workspace compared to the traditional CNC machines adopted. This implies the need for software able to completely exploit the possibilities of these systems by generating suitable toolpaths according to the process itself (reference substrate, layer, initial CAD model and parameters) and the specific cell at

hand (avoid collision, unreachable points and limits infringement). Significant is the tendency towards completely automated system (both hardware and software) to make the final machine accessible to a wider group of people, non-necessarily expert in these fields.

Research and Development stages Being the AM, a vast world relying on different physical principles to achieve the final part, a single technology has been singled out to carry out the research of new robot-assisted building strategies. FDM (FFF and other extrusion-based technologies) has been singled out primarily because it operates in a relative safer working environment (no need to contain harmful light emissions, to work with oxygen depraved environments, metallic dust or toxic fumes, although air filters and aspiration system are required for some process materials), hence easier to implement and bring up in a lab.
High priority has to be given to the design, simulation, control and integration of the devices (i.e. extrusion and robotic system) that will be used throughout the research. On a lower level of priority, the path/trajectory planning and collision check/avoidance needs to be added. It has to be decided if opt to work with pieces of software already at disposal (by combining them) or starting from scratch. Available routines might speed up the research activity but create significant gaps in the learning process that could propagate onto later stages, that can make a solution be discredited only upon a lack of understanding of the code used. Therefore, if and where possible, the CAM and simulators are developed using Matlab or Python language. This offers the possibility to give birth to a software able to function for any of the building strategies.
Then, some preliminary experimental tests are going to be performed to check and validate the setup and also to analyze the pros and cons of such solutions. The results will determine which road to pursue.
Here it is reported a list of the activities with a reference to their respective chapter in this document:

- **Additive Manufacturing and Robotics** (chapter 1): review of the current trends on the AM field, highlighting the technologies, processes, CAM software algorithms, kinematic solutions adopted followed by its current limitations.

- **Extrusion system** (chapter 2): it deals with the design and integration of parts to create an extruder able to process different materials, reliable (hardware, but mainly software-wise) and easily interfaced with other machines;

- **Robotic system** (chapter 3): it revolves around the study of industrial robots' control and I/O, in the physical and digital world, to bend them to our goal;

- **Trajectory planner** (chapter 4): it focuses on the development of the required routines to generate the motion of the tools (primarily the nozzle) respecting constraints on the geometry of the 3D model, feasibility of the process and safety;

- **Preliminary experimental test** (chapter 5): during this phase, very simple case studies are manufactured to inquire the feasibility of the main features of each building strategy and avoid the over-development of not promising solutions;
- **Novel building strategies** (chapter 6): it exploits the results from the preliminary test phase in conjunction with the issues identified in the state of the art to redefine how parts are printed (also accounting for the cell and its control), with the consequently testing campaign;
- **Industrial application** (chapter 7): results (partial or definitive) should find space in actual industrial contexts to solve problems that companies have;

Chapter 1

Additive Manufacturing and Robotics

The world of AM and its current trends establish the foundation of the matter. The review starts by addressing the different technologies (subsection 1.1.1), in the broad field of AM as well as the different materials (subsection 1.1.2) that can be processed to build the final part. It is followed by a brief list of known issues of these technologies (subsection 1.1.3). Focus is then put on the steps to obtain the final built starting from the CAD model (subsection 1.1.4. The definition of the machine instructions is a responsibility of a CAM software, that in the AM context it is also known as Slicer, that takes as input a digital model of the object to be created and outputs the machine code.
In the *Slicing* section, how this software works is discussed. The attention focuses on conventional slicing (subsection 1.3.2), that exploits 3-axis machines (such as Cartesian and Gantry robots) and layered manufacturing. For machines with more than 3 Dofs and geometries defined also varying the orientation of the normal to the printing surface, the toolpath generation requires different slicing algorithms that are discussed in (subsection 1.3.3) and (subsection 1.3.4).
In the third batch, the trend of kinematic solutions deployed to the field of AM (and in particular the one of Fused Deposition Modeling, FDM) is analyzed (section 1.4).
The chapter ends with the challenges identified by experts of the field to be solved in order to improve the technology (section 1.5).

1.1 Overview of the Additive Manufacturing

According to American Society for Testing and Materials AM is "*The process of joining materials to make objects from three-dimensional (3D) model data, usually layer upon layer, as opposed to subtractive manufacturing methodologies*".
This definition perfectly embeds the main philosophy behind these technologies. It can be noticed that it is referenced 3D model data as source (usually STL files) and that

the process is carried out as superimposition of layers of material; these two concepts, being the foundation of AM, will be discussed in depth in the following subsections. A third crucial concept in the definition is the reference to subtractive technologies (milling, turning and Electrical Discharge Machining) because, even if true that they are two opposing philosophies, they do share similarities on how the tool is moved (i.e. the machines used).
The first machine was developed in the 1980s by MIT (Massachusetts Institute of Technology) and presented to the world for the first time at the Detroit Autofact show in November 1987; then it has been retailed by 3D Systems [1].
The success and wide spreading of AM machines can be said to be due to the affirmation of rapid prototyping as a viable tool during the design process. This kind of machines offers the possibility to see and touch the developing object as it's being designed. If from one side it offers a reduced lead time compared to other prototyping technologies and small engineering time (variation in the CAD file), on the other not all the configuration or object could be realized (although many geometries cannot be manufactured with standard technologies as well). History of the AM field can be inquired at [2].
Now a days, AM machines are widely used also by hobbyist since the diffusion of cheap and/or open source Cartesian plastic FDM machines.
The technology opens up a new manufacturing field that needs to be charted, since it would be possible to reconstruct damaged goods or directly produce them. For now, it's limited to the production of custom and small-size batches (one single component print is very common).

1.1.1 AM technologies

There are seven building principles that yields as many AM technologies. They are listed in the ISO 17296-2 (last updated in 2015). For the sake of completeness, it's worth mentioning that further technologies by addition exist and that they are usually exploited in the Integrated Circuit (IC) field, such as: Vapor Deposition (Chemical, Low-Pressure Chemical, Plasma-Enhanced Chemical and Atmospheric Pressure Vapor Deposition), Epitaxy, Physical Vapor Deposition, Evaporation, Sputtering and Electrochemical Deposition. If interested the reader can address them at [3].
Here the main AM technologies are listed [4].

Stereolithography: It was the first commercialized AM technique. It exploits a liquid material (photopolymers) that undergoes a chemical transformation upon interaction with UV light.
A platform is placed inside a tank containing the liquid polymer that menages the layer heights. The process begins with the adjusting of the bed in order that a thin film between it and the atmosphere (free surface) is created. Then a light source is used to solidify only the interested partition of the film. Completed a layer, the platforms move

to create a new film upon the one just induced. Since standard laser optics interacts with the polymer only in one point at a time making the process quite lasting, DMD (digital micro mirror device) has proven to realize a single layer simultaneously (mask projection approach).

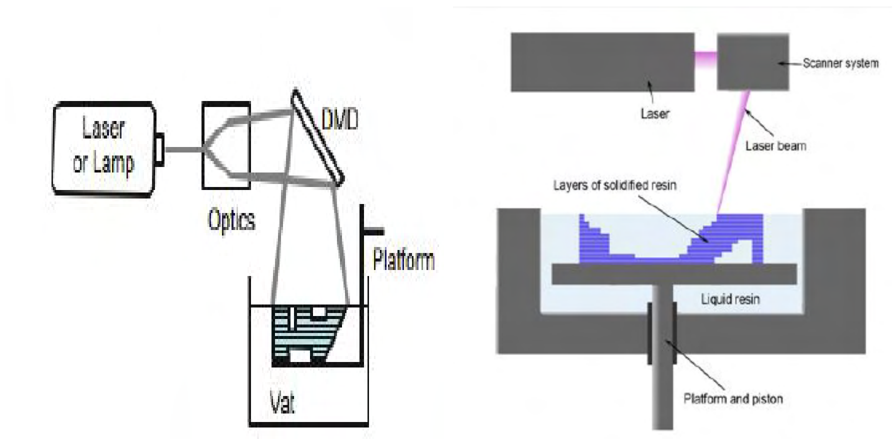


Figure 1.1: Stereolithography with and without DMD [4].

Selective Laser Sintering and S.L. Melting: This technique uses a high-power laser (for instance, $C0_2$ laser) to fuse small particles of plastic, glass or metal. SLM uses even higher power laser resulting in a better final result that requires less post printing operations, but the process in itself is more demanding, errors could lead to residual stresses and strains. The printing procedure is not that much different from stereolithography as can be seen in the Figure 1.2.

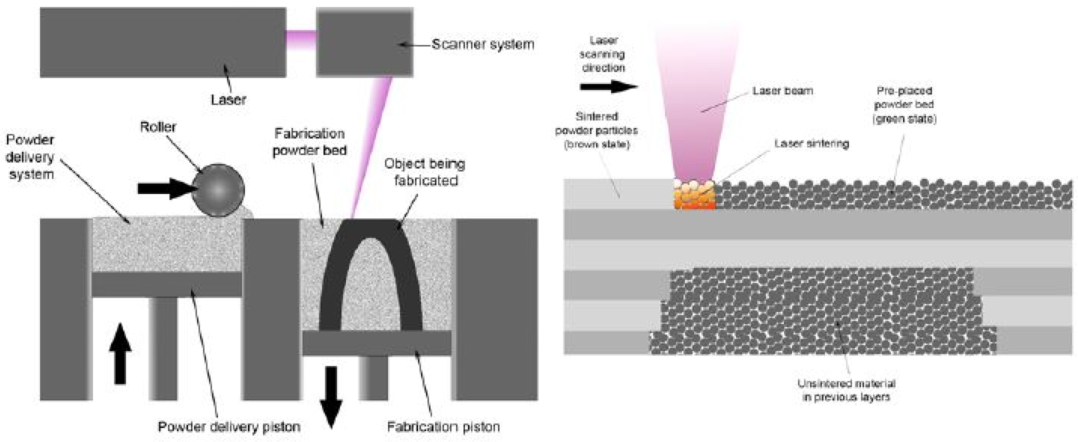


Figure 1.2: Selective Laser Sintering and S.L. Melting [5].

Direct Energy Deposition: A laser creates a melt pool on the build area that is sprayed with powder that melts upon interaction and then solidifies. Inert gas shields the process and, according to the gas composition, it could improve the melting process exploiting exothermic reactions.

Material Jetting: A photopolymer is squirted from hundreds of tiny nozzles in a printhead. As the droplets are deposited on the build platform, they are cured and solidified using UV light. It usually requires support structures that are manufactured using a soluble material that is easily removed during post-processing.

Binder Jetting: A thin layer of ceramic or metal powders is glued together by a binding adhesive agent. When it is completed, a new layer of powder is spread onto the build area. The process repeats until the part is complete. When the manufacturing process is through, the part requires additional post-processing before being ready to use.

Sheet Lamination: Layers of adhesive-coated paper, plastic, or metal laminates are successively glued together by a heated roller and cut into shape by a laser beam.

Fused deposition modeling (FDM): FDM was invented and sold by Stratasys. It's an extrusion-based process in which the material contained in a reservoir is forced out through a nozzle, when pressure is applied, to generate the final object. This is the technology behind the majority of the 3d printers available for private users since they do not require laser system or particular operating environments. The material available are mainly polymeric (ABS and PLA being the most spread). This technique requires the material to reach a temperature a little higher (usually 10°C) than the one of fusion at the nozzle; the deployed filament solidifies once in contact with air.

1.1.2 AM materials

AM works with different types of material and cartridges as it can be deduced from the descriptions of the main additive strategies. Here are reported the main material typologies:

Ceramics: They can be printed by means of a binder (polymer) storing their particles (i.e. feedstock) that, in a following phase, is removed usually by heat (thermal debinding). AM proves to be more effective with these materials compared to standard technologies, due to their brittle and hard nature. Direct printing has been tried, but the procedure is limited by their high melting temperature.

Metals: The main metallic materials used in 3D printers are stainless steels, aluminum, Cr-Co and titanium with particle size around 10-50 μm [6],[7].
They can be directly or indirectly printed. Direct methods are based on melting the particles to obtain the final object by means of laser or EBM (Electron Beam Melting).

The final parts are characterized by high mechanical properties.
Indirect methods relays on the partial melting of feedstock (with metal particles blended in the binder). Green parts require post processing usually to remove the binder itself; then, the debinded parts undergo to sintering (inter-particle bonds are here formed). Shrinkage occurs in this phase so it is something that has to be carefully managed because it could provoke significant distortion and cracks in the part.

Polymers: These materials are made up of molecules that repeats the same structural chain. The main ones used are nylon, photopolymers, ABS, PLA. Nylon is used especially with laser-based techniques for its melting and joining properties. ABS is the most spread material for FDM. Photopolymers reacts if treated with laser with specific wave lengths.

1.1.3 AM quality and known issues

AM is able to manufacture components that with conventional technologies might not be possible, however some drawbacks limit this branch of manufacturing [8]:

- **Anisotropy:** parts have different physical and mechanical properties according to the part orientation. This issue is due to the layer-by-layer material deposition; the bond between layers is weaker compared to the tensile strength normal to the building direction.

- **Staircase effect:** it causes a decrease in the surface quality as huge as the surface curvature increases or the layer height get bigger.

- **Support structures:** overhanging volumes require the generation of supports to prevent them from collapse during printing. Supports are time and material consuming and their elimination in post-processing has to be performed without damaging the build.

- **Printing time:** it is directly related to the number of layers to be printed, the contours length, the number of perimeters, the infill percentage and the support structures to be created.

Several parameters affect the superficial quality, mechanical resistance and geometrical precision [9]. Defects inside the layers are impossible to remove in post-processing, thus the realization process is very important. Geometrical precision depends on how the material is added. In case of FDM, the material is deposited by a round surface whose center it's repositioned following a trajectory, usually engineered considering only straight lines. This can cause the circle to spend more time over a region than another; the print will have therefore overfill or underfill areas (see Figure 1.3). Errors could be due to expansion or shrinkage that the filament might undergo during extrusion and while settling on the plate.

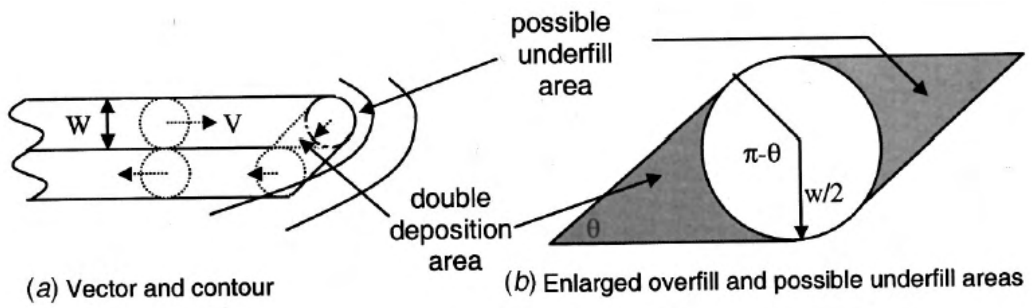


Figure 1.3: Overfill and Underfill example [9]

1.1.4 AM process

An AM process involves several steps (Figure 1.4) that takes the initial digital model to its final physical realization. It can be split in seven phases regardless of the AM technology considered, but goes without saying, that they have to be declined to the actual process and for the material exploited [4]:

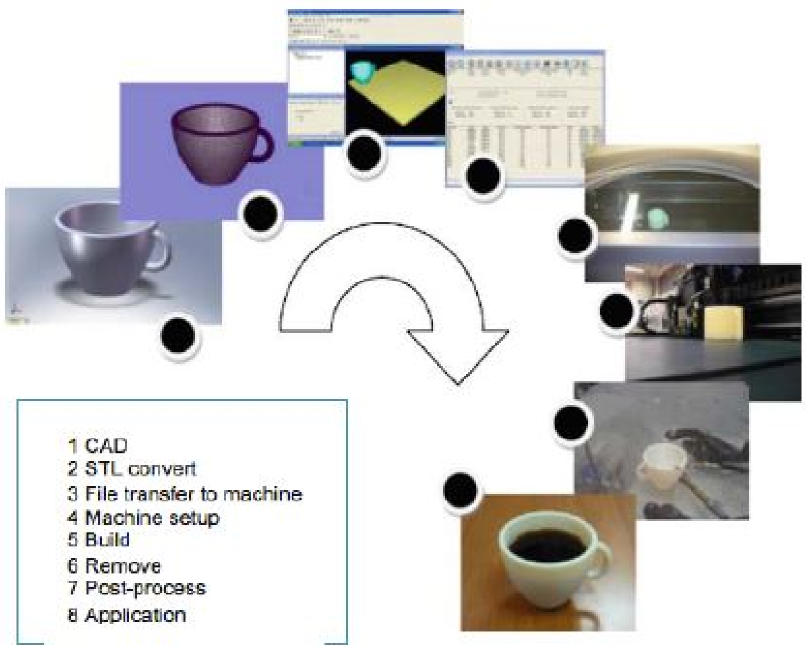


Figure 1.4: AM process [4]

1. **CAD**: The first step is the realization of a software model that fully describes the geometry. Any CAD solid modeling software can be used, but the output must be a 3D solid or a surface representation. Parts should be modeled following principle dictated by designing for AM to avoid the forwarding of unfeasible parts to the next phases. Reverse engineering equipment (e.g., laser and optical scanning) can also be used to create such representation. See subsection 1.2.

2. **Conversion to STL**: Every slicing software accepts STL files, which has become a de facto standard, and nowadays nearly every CAD system can output such file format. It describes the external closed surfaces of the original CAD model and forms the basis for calculation of the slices. See section 1.2.1.

3. **Transfer to AM Machine and STL File Manipulation**: The STL file must be processed by a slicing software before being transferred to the AM machine. Here, there may be some general manipulation of the file so that it is the correct size, position, and orientation for building.

4. **Machine Setup**: The AM machine must be properly set up prior to the build process. Slicing software requires all these boundaries in order to generate the correct *G-code* that the AM machine will use during build.

5. **Build**: Building the part is mainly an automated process and the machine can largely carry on without supervision. Only superficial monitoring of the machine needs to take place at this time to ensure no errors have taken place like running out of material, power or software glitches, etc.

6. **Removal**: Once the AM machine has completed the build, the parts must be removed. This may require interaction with the machine, which may have safety interlocks to ensure for example that the operating temperatures are sufficiently low or that there are no actively moving parts.

7. **Post-processing**: Once removed from the machine, parts may require an amount of additional cleaning up before they are ready for use. Parts may be weak or present supporting features that must be removed. This often requires time and careful, experienced manual intervention.

From here on in, the discussion will vert on these phases to get the reader acquainted with such aspects.

1.2 AM features

The possible geometries that can be created using the principles of *design for AM* are classified in two categories, *2.5D* and *Free-form features*, [6], [10]:

1. **2.5D feature**: They have their reference plane on a planar surface, which makes them very suited to planar slicing. They can be split into the following subcategories:

    - *Stretch*: it is defined as a planar profile lying on the reference plane (the z=0 plane in the local coordinate system) and stretching along the z-axis as shown in Fig.1.5*a*. The sketch profile parameter describes the profile to be stretched and the stretch height parameter determines the length of the

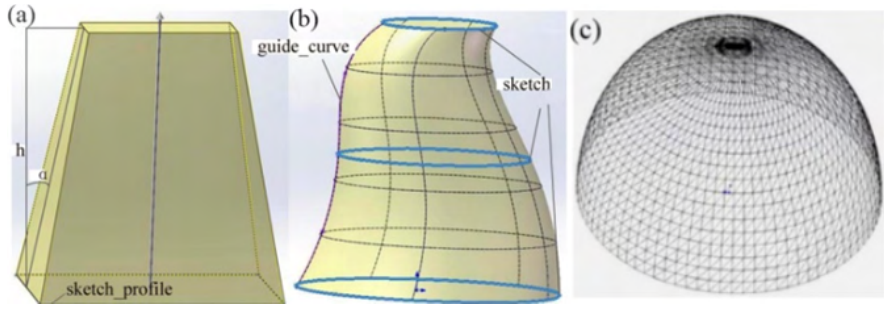


Figure 1.5: Types of 2.5D features [6].

yielded solid. If the sides of the feature are not orthogonal to the reference plane, a non-null value for the draft angle is specified; if it is positive, the solid shrinks inward towards the z direction, otherwise, it expands outward;

- *Loft*: it is essentially a solid that connects a series of profiles lying on parallel planes along a guide curve as shown in Fig.1.5*b*. The section profiles are contained in the sketch profiles list, and the guide curve is used to control the trend of the solid shape;
- *Mesh*: The mesh feature is used when the geometry is described by meshes which does not require high precision and usually refers to the widely used triangular mesh depicted in Fig.1.5*c*.

2. **Free-form feature**: They have their reference plane on a curved surface (just as the fins depicted in Fig.1.6), which makes them the classical application case for the non-planar layer slicing.

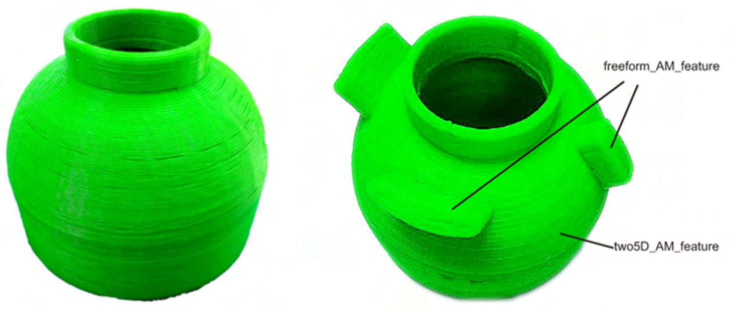


Figure 1.6: The fins added to the jar are an example of free-form AM features [6].

1.2.1 CAD File Formats in AM

The CAD file formats on which AM rely can be divided in the following two macro-categories:

- Native File Formats. They are proprietary of a specific CAD software maker and can be edited only with the proprietary environment;

- Neutral or Standards. They are specifically created to encourage interoperability, which helps files exchange between different software programs. IGES, STEP and STL are the most popular.

Hereafter the analysis focuses on the standardized file formats due to their higher versatility.

IGES [10]
The IGES format was introduced in 1979. The CAD geometry is represented through an analytical model. For this reason, this type of format is not prone to data degradation and the consequent inaccuracy.

STEP [11]
The STEP (Standard for the Exchange of Product model data) format was introduced as ISO standard in 1994 (ISO 10303). The reason why the STEP format is becoming increasingly popular is that it preserves the original geometrical information (the features tree is still available after the conversion).
The solid features are represented through the Boundary Representation (BRep) which is a way of characterizing a volume in 3D space through a collection of closed surfaces. The features recognition is accomplished by a volume decomposition algorithm based on the concave detection of loops. According to this model, the edge directions in the object model is defined such as, walking along an edge, the corresponding face is always on the left-hand side. When an edge is in the external loop of a face, its direction is in a counterclockwise direction relative to the surrounding face. On the other hand, when an edge is in the internal loop of a face, its direction is clockwise as shown in Fig.1.7.
Once determined the loops direction, the further step for the accomplishment of the

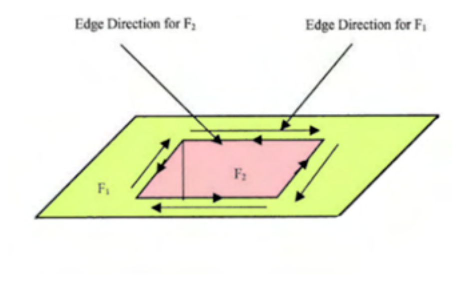


Figure 1.7: Determination of the loop direction.

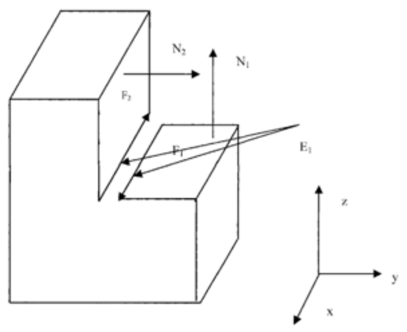


Figure 1.8: Concavity test for volume decomposition.

volume decomposition is the so-called concavity test (Fig.1.8). Considering two contiguous faces F_i and F_j sharing a corner edge E_k, the concavity test is performed as follows:

- The direction of the edge E_k with respect to the face F_i is determined. The normal vectorN_i of the face F_i must be the first component in the cross-vector product of the second step;

- The cross-vector product V of the faces directional vectors is computed as:

$$V = N_i \times N_j$$

- If the direction vector of the edge E_k is in the same direction of the cross vector product V, then the edge E_k is a convex edge, which concludes that F_i and F_j are convex faces; on the contrary, it is a concave edge and F_i and F_j are concave faces. Eventually, if the cross-vector product vector V is a zero vector, then the edge belongs to a tangent category.

STL [6],[12],[13],[14],[15]
The STL format (Standard Triangulation Language) was invented by the Albert Consulting Group for 3D Systems in 1987.
A surface is described by a triangular mesh (using many triangles together with their outward pointing normal). This representation is an easy way to characterize any kind of surface, but has some shortcomings:

- it approximates the original surfaces. The user needs to input the acceptable chordal tolerance which is the distance between the triangle facet and the surface it is approximating (Fig.1.9 a). The data degradation is as huge as the tolerance is broad (Fig.1.9 b): increasing the number of triangles smooths the surface, but also leads to much larger data files. The user has to balance the accuracy issue with the file size issue, especially for highly non-linear surfaces;

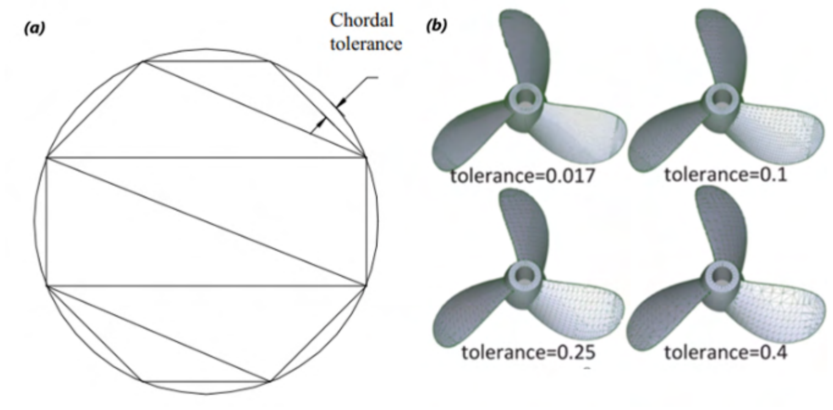
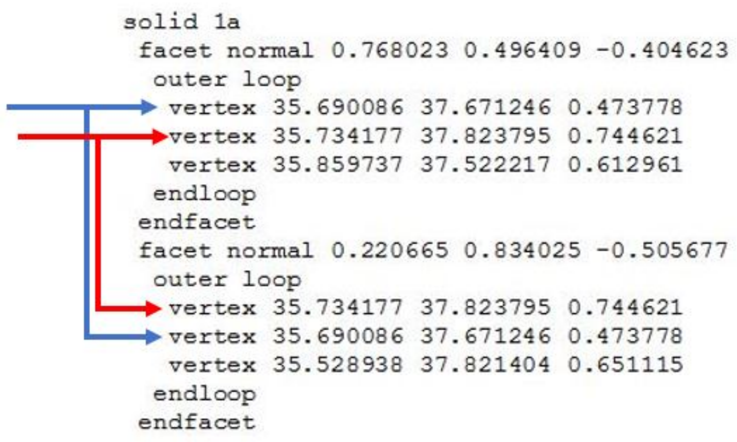


Figure 1.9: (a) The chordal tolerance sets the STL data degradation degree [14]. (b) The final surface is as smooth as the chordal tolerance is small [6].

- since every triangle shares all its vertices with the neighbor ones, the shared ordinates are duplicated within a file (Fig.1.10);



```
solid 1a
 facet normal 0.768023 0.496409 -0.404623
  outer loop
   vertex 35.690086 37.671246 0.473778
   vertex 35.734177 37.823795 0.744621
   vertex 35.859737 37.522217 0.612961
  endloop
 endfacet
 facet normal 0.220665 0.834025 -0.505677
  outer loop
   vertex 35.734177 37.823795 0.744621
   vertex 35.690086 37.671246 0.473778
   vertex 35.528938 37.821404 0.651115
  endloop
 endfacet
```

Figure 1.10: example of repeated coordinates

- there's no topological information, which often induces errors such as gaps, mixed normal, overlaps, which requires the use of "verify and repair" software.

1.3 Slicing

In the previous section, the stages of a generic AM process have been outlined and the focus has also been put on typical geometries and digital model file formats. These models need to be processed to yield the path that tool has to trace to reconstruct the object. This operation is called *Slicing*. As the name suggest, it consists in cutting the model using planes (or generally surfaces) to get where to add the material.
Because the slicing procedure strongly depends on the layer geometry, a review on them is due; it is then followed by the current trends in toolpath generation for AM machines. They are going to be addressed according to the building strategies that are: planar-layer manufacturing (conventional), non-planar and multi-directional.

1.3.1 Building surfaces

The cutting plane (or planes in some case) that the path planner has to exploit shapes the building strategy. This impact the complexity and generalization of the software to be developed or run.
Let's introduce first the types of layers for each building strategy. Namely they are: *Conventional*, *Planar multi-directional*, *3D multi-directional* and *Cylindrical*.

Conventional printing (exemplified in Fig.1.11, *top-left*) is straightforward to work with. Once identified the surface (i.e. a plane) defining the substrate, it is possible to define the equally or feature-based spaced layers that will slice the object.
The starting plane could be horizontal, the common case, or oriented in space. This

variation does not affect the fact that the printing will be carried out without change in the orientation of the tool. The algorithm can either face these cases by tilting the batch of planes so that they are oriented as the building direction or by rotating the digital model so that is aligned as in the common case.

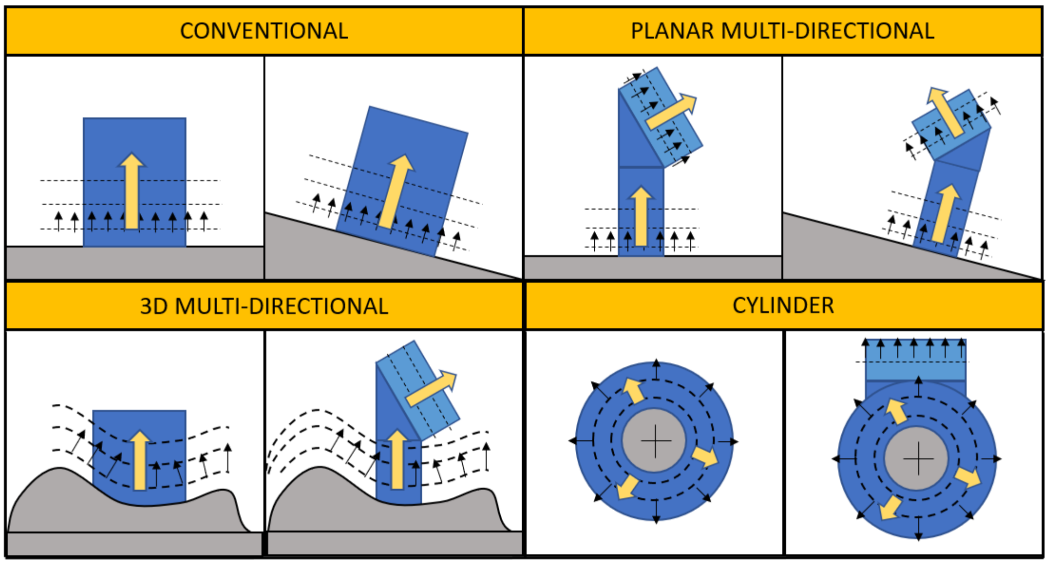


Figure 1.11: Layer (or cutting planes) change according to basic building philosophy (*conventional*, *planar multi-directional*, *3D multi-directional* and *Cylinder*)

Planar multi-directional printing (exemplified in Fig.1.11, *top-right*) is a direct derivation of the previous one. This method is used to avoid the generation of support and subsequently reduce time and waste material. It requires the decomposition of the model in subparts, characterized by the same orientation, either by automatic or manual intervention.
Layers can be generated for each subpart as stated for the conventional case. Batches of parallel planes can be oriented as the identified printing directions; the most delicate regions are the ones where one direction gives way to the new one. The object could be physically divided in subparts generating n independent models sliced separately (hence avoiding planes leaking from one part to the other); an alternative is to transform a subpart to a standard case and then, after the slicing, map back the toolpath in its original position.

3D multi-directional printing (exemplified in Fig.1.11, *bottom-left*) differs from the previous two strategies because, instead of planes, the layer mimics the substrate that is a generic surface in the 3D space. The substrate could be known in advance or acquired from the physical part (contact or contactless measuring system).
It should be checked that the surface is reachable by the extruder; convex geometries do not present any obstacle, while concave might have regions that are unreachable by the TCP (collision of the extrusion body with the substrate).
The "multi-directional" classification refers to the fact that the layer requires the tool

orientation to be updated at each position; it's possible, moreover, to have the surface adapting from one layer to the following one or after a batch to generate a new substrate geometry as for the previous cases, hence reinforcing the name.

Cylinder printing (exemplified in Fig.1.11, *bottom-right*), that will also be indicated with *lathe-like* printing, is a spacial case of the 3D multi-directional in which the substrate rotates like in a lathe machine. This method is suitable when shaft, rods and such have to be printed.
The superimposition of passes around the rotation axis and the ones along it mitigates the issues related to notch formation and crack propagation of the counterparts realized with conventional printing (they form an intertwined structure).
Examples of this method in literature are [16],[17],[18].
For non-tubular shapes, this building strategy can only characterize a portion of the overall print (for instance, the phase depicted in Fig.6.20-B uses this method after conventional printing). After completing a transition phase where a new building direction is set up, conventional (or the generic 3D multi-directional printing) method can take over (see Fig.1.11-*right*).

1.3.2 Conventional Slicing

This slicing procedure strongly depends on the format of the input file [6],[13].

- If a STEP file is provided: the contours of each layer can be evaluated with an analytical formulation and only in some cases it is necessary to resort to the plane intersection method. The following list describes the analytical slicing procedure according to the feature type:
  - *Stretch*: The stretch direction is orthogonal to the reference plane and the contour of every layer is similar to the base profile. Therefore, the contour of a layer is determined only by the base profile, the layer height and the draft angle:

    $$contour(\Delta h) = f(sketchprofile, \Delta h \tan(\alpha))$$

    Where: Δh is the height of the layer; α is the draft angle value; f is the offset algorithm for plane offset curves.
    For the inner profile of the shell solid the computation is carried out subtracting the wall thickness to the above equation;
  - *Loft*: The contour of a certain layer is either one of the section profiles or the interpolation of two adjacent profiles which can be directly calculated with the feature's definition parameters;

  - *Mesh 2.5D and freeform*: In this case, the layer contour can't be obtained in a closed form and there is the need of computing the intersection. Once again note that the mesh model approximates the original model, consequently there is an error between the contour computed by the intersection algorithm and the theoretical contour.

  The slicing procedure can be performed directly in the CAD software environment without any file format conversion (direct slicing). In this way all the surfaces keep their mathematical formulation and the intersection between the feature and the plane can be carried out analytically. The resulting layer contours come with no approximation, but at the price of high computational cost. Moreover, the slicing procedure is run entirely in the proprietary software environment without the possibility of using other slicers more tailored on the user needs.

- If an STL file is provided: the most widespread approach is intersecting the model with planes parallel to the lower layer. In the simplest case of evenly spaced layer, each of them intersects some of the STL triangles thus producing some intersection lines on their facets. Then, these segments are rearranged to form a contour per layer. This algorithm is easy to implement but might introduce a certain degree of approximation due to the data degradation built-in in the STL format.

STL based conventional slicing

Being the STL file the go to format for all the slicing software, the procedure is here detailed.
The algorithm that yields the toolpath from a STL file can be summarized in these steps [19]:

1. Import STL
2. Slice STL
3. Offset contours
4. Infill
5. Optimization
6. G-code generation

1. Import STL In this first step, the n triangles (defined by their vertices) stored in the .STL file are read and stored. These triangles are not necessarily saved in a logical or deterministic way in the imported model, fact that will impact the following phases. Vertices are stored in a matrix whose dimensions are $3n \times 3$ due to repetition of shared points. The connectivity list is a $n \times 3$ matrix. Data could be manipulated to get a lean representation (i.e. reduction of points required).

2. Slice STL The next phase consists in the determination of the contour imprinted on each slicing plane by the model [20], [21], [22], [23].
These planes can be categorized in two families:

- *Constant spacing*: the widespread solution for AM manufacturing. The planes are parallel to each other and equally distanced by a quantity h. The planes $\Gamma(h)$ are the loci identified by:

$$\Gamma(h) : n \cdot (P - P_0 - nh) = 0$$

with P_0 the reference point of the plane (for example the origin of the print bed) and n the unitary vector normal to the plane.

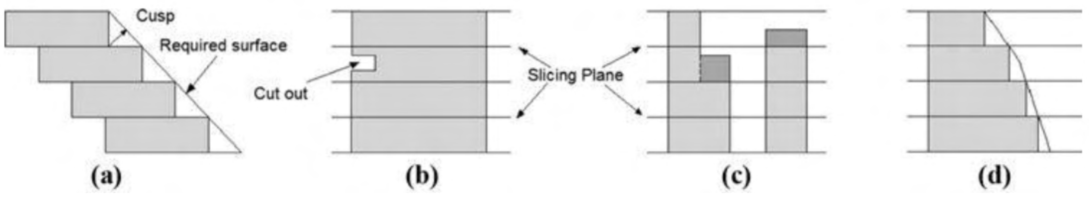


Figure 1.12: Surface quality issues induced by the uniform thickness slicing [23].

- *Variable spacing*: the uniform spacing can be detrimental for the surface quality because it neglects local surface curvature variation, Fig.1.12. A possible solution could be to select the layer height as a function of the desired cusp height (h_c) and the local curvature (β):

$$h(h_c, \beta) = \frac{h_c}{\cos(\beta)}$$

This value needs to be checked against the process constraints (for example in FDM the nozzle diameter bounds the range in which the thickness can be selected, see subsection 2.1.1).

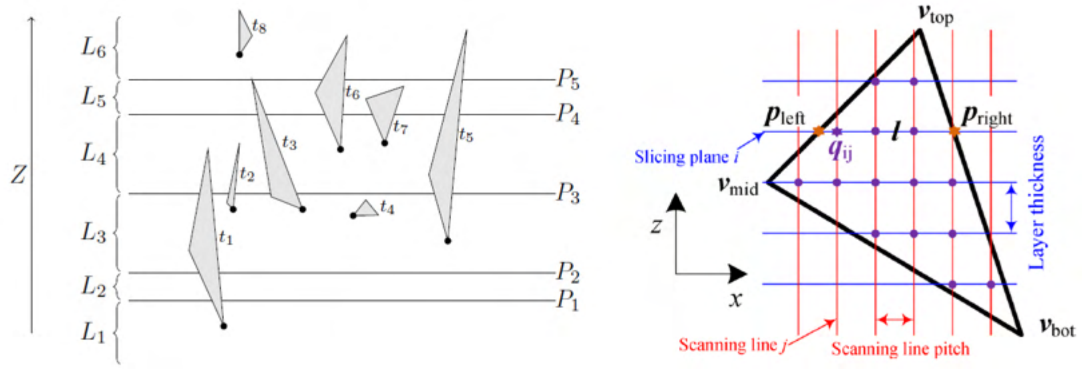


Figure 1.13: Triangle slicing [20],[22].

The intersection between a triangle and a plane is straightforward; the geometrical representation is available in Fig.1.13. Given a segment between $\nu_1 = (x_1, y_1, z_1)$ and

$\nu_2 = (x_2, y_2, z_2)$ and a plane parallel to the XY plane at Z equal to z_i, then:

$$\begin{cases} x_i = x_1 + \frac{(z_i - z_1)(x_2 - x_1)}{z_2 - z_1} \\ y_i = y_1 + \frac{(z_i - z_1)(y_2 - y_1)}{z_2 - z_1} \end{cases}$$

After considering all three segments, the intersection points are obtained. The possible combinations are here listed, considering just the unique values:

- *0 point*: no intersection has occurred
- *1 point*: the plane touched only one vertex
- *2 points*: the plane sliced the triangle
- ∞ *points*: triangle and plane are coplanar

The sparse segments generated need to be sorted to get a sensible contour. There are several means in literature, the simplest uses the L^2 norm to determine, once selected a first element, the one to put before and/or after. Due to the fact that at least two triangles share the same edge, the distance is close to 0. This way is possible to distinguish multiple contours, for example tell holes from outer boundaries.

3. Offset contours The final build object could have been designed to have multiple perimeters. Therefore, algorithm able to generates inward or outward copies of the contours have been adopted to this end [24],[25],[26].
The offset curve can be indicated, knowing the original curve $C_{original}(\lambda)$ and its normal vector space $n(\lambda)$, (still in the 2D space) as:

$$C_{offset}(\lambda) = C_{original}(\lambda) + n(\lambda)d$$

This kind of approach hides some issues that can be seen in Fig.1.14; they are, referring to the figure:

1. at corner points, moving inward, there are discontinuities or curvatures that generates ambiguities in the determination of the profile; *Case a-d*
2. at corner points, moving outward, additional arcs need to be added to close the profile to guarantee constant distance from the corner; *Case b*
3. the offset curve can self-intersect giving birth two at least two children curves that will be use for the formation of extra perimeters; *Case c-e-f*

Several algorithms exist in literature tailored to solve or cope with these drawbacks because it is a consequential aspect in machining and path planning (for example, toolpath to mill the desired profile of a cam disk).

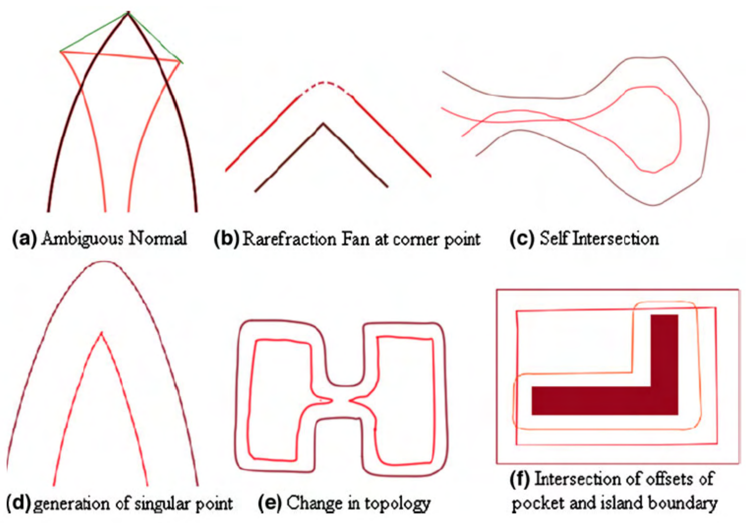

(a) Ambiguous Normal
(b) Rarefraction Fan at corner point
(c) Self Intersection
(d) generation of singular point
(e) Change in topology
(f) Intersection of offsets of pocket and island boundary

Figure 1.14: Problems arising in contour offsetting [25].

4. Infill Once identified all the contours for a given cutting plane, how the remaining region is going to be filled needs to be assessed. Here some examples taken from literature are reported [27], [28], [29], [30].
Toolpath generation strategies, deriving from milling, can be split into two categories (Fig.1.15). The first one uses the desired outer perimeter to define the inner strokes (i.e. continuously generating the offset perimeter inward). The second one uses a given pattern to cover all the region.

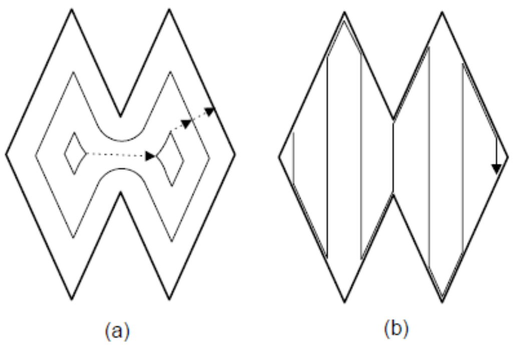


Figure 1.15: Contour-Parallel (a) vs Direction-Parallel (b) [27].

Let's consider only the latter (the philosophy and issues behind the first approach have been presented in the previous paragraph about additional perimeters) using just parallel lines instead of specific patterns and geometries (Fig.1.16 shows an example). First, the contours are intersected with a series of parallel lines. The distance between neighboring segments yields the infill percentage that is how much dense the build will be. The points obtained need then to be connected (i.e. final toolpath). This operation has to be carried out striving for:

- Minimize the number of tool retractions
- Minimize the number of tool-path elements
- Maximize the average length of tool-path elements
- Technological requirements

Fig.1.16-*step2* depicts an example of segment connection. Starting from one point, randomly selected or depending on the last point of the previous layer, it first moves along the line, arrived at the edge, it jumps over to the closest one and repeat the operation. Some points will be left out in this way. The algorithm uses the next available point as

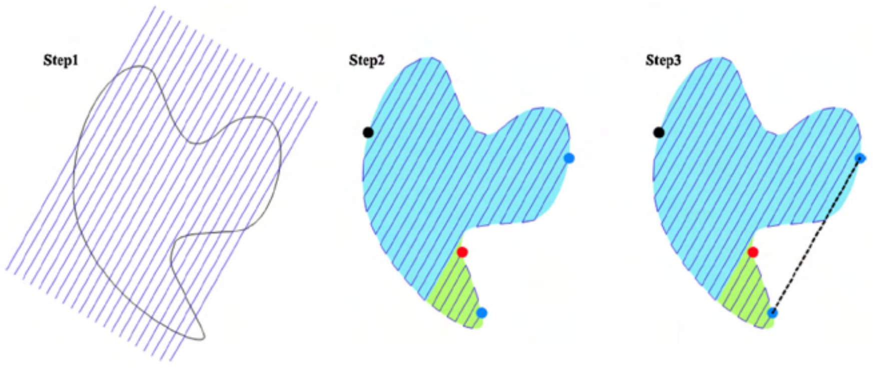


Figure 1.16: Direction-Parallel example [31]

the starting one for an analogous run. Crucial is how the system transition from one subpart to the other. Subtractive manufacturing performs this by completing a "U" shape exiting the work piece to avoid any undesired machining. AM uses analogous mechanism to stop extrusion (for example FDM retracts the filament, SLA-SLS-SLM switches the "light" source off).
Tool retraction might not be as effective as expected especially in case of extrusion-based technologies (for example FDM-FFF or concrete printing). These types of extruder might leak due to some technological drawbacks or to the state of the working material. Thus, other types of infill have been developed to minimize the number of free movements. Fig.1.17 is an example of such approach that decompose the region in simple subpart and generates an infill that can walked back and forth. Then, the parts are connected making a unique loop.

Discontinuities of the tangential field (i.e. sharp turns and corners) cause the tool to expose a wider region with the source light or extrude more material (already anticipated in the AM known issues, Fig.1.3-a). The tool spot can be wider (order of the $10^{-1}mm$) causing an undesired accumulation of material in one region (overfill). Due to the shape of the nozzle some region might never be exposed (underfill).

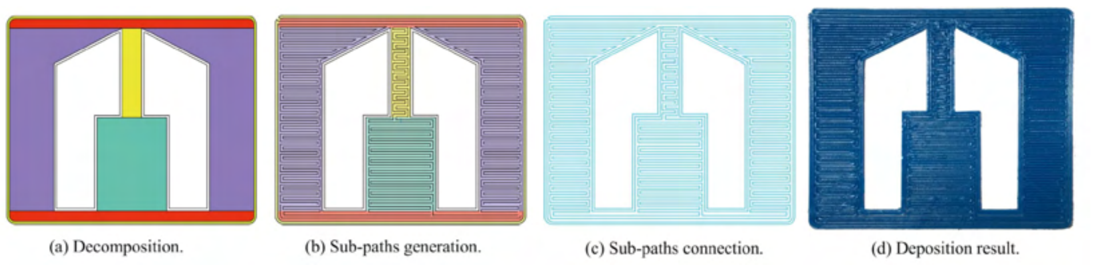


Figure 1.17: Retraction-free toolpath generation example [30].

As concerns other patterns, the infill commonly used by commercial slicers are namely: grid, triangles, stars, cubic, concentric, honeycomb, 3D honeycomb, gyroid, Hilbert curve, Archimedean chords and octagram spiral (Fig.1.27 shows a collection of these patterns).

5. Optimization The toolpath identified until now might not be sufficient to get a good final print. Additional tweaks are due, of course they do strongly depend on the technology adopted. Only the ones related to extrusion will be briefly discussed hereafter. For a detailed account on the matter, please refer on resources available in literature.
The additional feature can be split into two categories: a series of expedients used to improve the adherence of the first layer to the plate (skirts, brims, rafts; Fig.1.18) and a second to bear overhanging layers (supports; Fig.1.19).

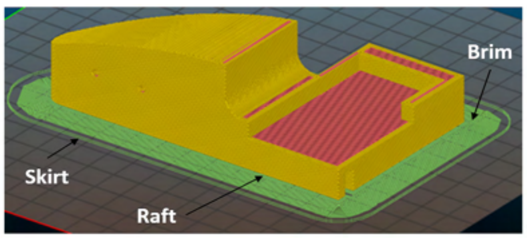


Figure 1.18: Additional feature at start

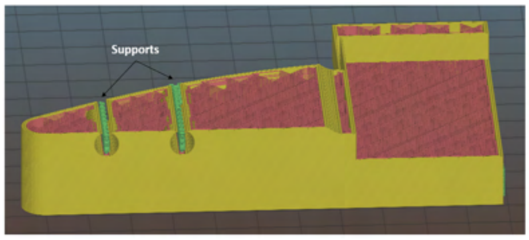


Figure 1.19: Addition during build

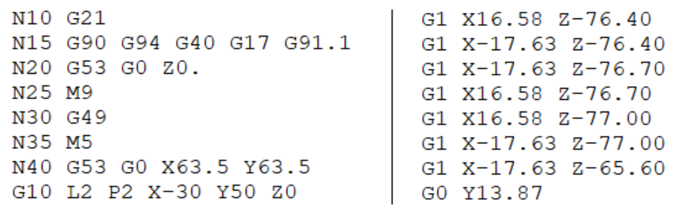
Here follows a list of the tools that fall in these categories:

- **Skirt**: It's used to stabilize the material flow, by extruding some sacrificial material first (thus filling the extrusion chamber). Usually consist in one or more lines mimicking the contour of the first layer at a given distance from the rest of the print to not compromise the actual start.

- **Brims**: To improve the adhesion, the first layer is swollen to cover a larger area. It acts as sacrificial material that undergoes warping instead of the actual component. It does not constitute part of the build (the first layer has to be duplicated).

- **Rafts**: One or more layers at the base for support and adhesion. Usually is larger and with lower infill to simplify its removal. Usually separates the layer in contact with the bed from the actual bottom layer. It's not part of the final piece.
- **Supports**: Sacrificial material added to support layers that otherwise could not stand on previous one (overhangs and bridges). If the distance between two elements (i.e. bridging) to be covered is over 5 mm, then bridging can be used alongside supporting to improve the surface finish [32].

6. G-code generation The final toolpath, accounting also for the machining operation required, is translated in G-code language understood by CNC machines [33], [34], [35]. Therefore, a brief overview of this type of code is mandatory.
It is one of the most spread Numerical Control (NC) languages. G-codes can be classified into two types: modal and non-modal code. A modal type code is effective throughout the following blocks until it is canceled; a non-modal-type holds only within the commanded block and it's automatically canceled by the next block.
A command block is a single line of code (Fig.1.20) referencing to a series of functional groups pre-built in the CN controller (listed in Table1.1) called by means of alphabetic characters.



```
N10 G21
N15 G90 G94 G40 G17 G91.1
N20 G53 G0 Z0.
N25 M9
N30 G49
N35 M5
N40 G53 G0 X63.5 Y63.5
G10 L2 P2 X-30 Y50 Z0
```

```
G1 X16.58 Z-76.40
G1 X-17.63 Z-76.40
G1 X-17.63 Z-76.70
G1 X16.58 Z-76.70
G1 X16.58 Z-77.00
G1 X-17.63 Z-77.00
G1 X-17.63 Z-65.60
G0 Y13.87
```

Figure 1.20: G-code example

Table 1.1: Addresses group of CNC system

variable	**description**	**example**
G	preparatory function	G0: rapid linear movement
		G21: unit of measure [mm]
		G10 L2: Set Coordinate System
M	auxiliary functions	M107: Fan off
		M112: emergency stop
F	feed function	F1800: set feed to 1800 [mm/min]
S	spindle function	S2000: set spindle speed to 2000 [rpm]
T	Tool functions	T01: tool "1" is selected

The main ones are: *G* functions related to axis control and preparation of the CNC system (i.e. movement, set local coordinate system, interpolation); *M* functions dealing with the commands for the control of the tool (i.e. coolant/tool activation, adjustment of the parameters used by the tool, change of tool, workpiece and table); F function updates the relative speed between tool and workpiece; S function sets the spindle speed (RPM); T function imposes a new tool to be used by the interpreter for the following blocks.

Let's consider a single line in the file, for example: $G1\,X6\,Y6\,F600\,T01\,S8500$ indicates that we want the TCP to reach the point (6,6) in XY plane (*X6 Y6*) using a linear interpolation (*G1*), if possible, with constant feed (*F600* [mm/min]) using a defined tool (*T01*) spinning at a given speed (*S8500* [rpm]).

1.3.3 Non-planar slicing

In the previous section, the stages of a conventional slicer have been discussed. Planar slicing might excessively constrain and limit the possibility to manufacture the part as it has been intended. Hence non-planar toolpath generator is being developed. These methods will be discussed pointing out only the differences from the standard.

Four Vectors Algorithm [36]

The four vectors algorithm has been specifically thought to work with the STL files. It considers four different vectors at each surface point and exploits the neighboring points to build the cross-product vectors which serves to get a representation of the surface variations. Though effective in capturing the true surface trend, the method involves too many calculations and it is also sensitive to the relative position of the adjacent points. Besides, the absence of surrounding points on the edges requires the creation of some external pseudo nodes, thus adding further calculations.

These issues are overcome by slightly modifying the four vectors method and consid-

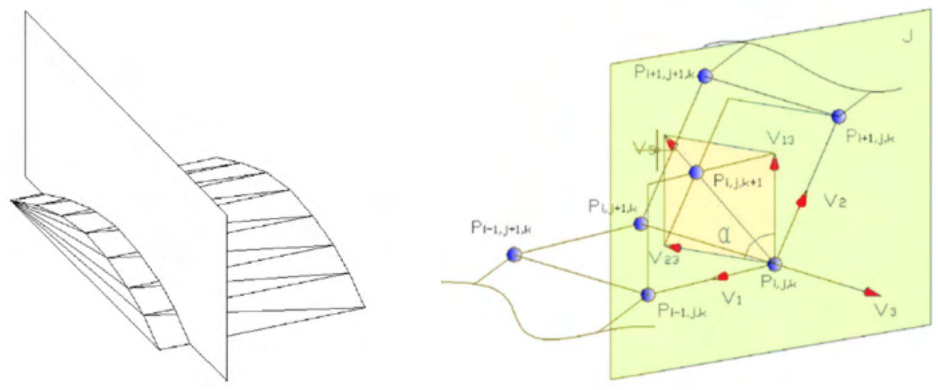


Figure 1.21: Vector definition for the modified for vectors algorithm [36].

ering a vertical plane J passing through the three consecutive surface points $P_{(i-1,j,k)}$, $P_{(i,j,k)}$ and $P_{(i+1,j,k)}$. The algorithm creates two vectors $\bar{V_1}$ and $\bar{V_2}$ between points $P_{(i,j,k)}$ and $P_{(i-1,j,k)}$, $P_{(i,j,k)}$ and $P_{(i+1,j,k)}$ respectively, while the auxiliary vector $\bar{V_3}$ is orthogonal to the vertical plane J from $P_{(i,j,k)}$. Then, the two vectors on the surface

belonging to the plane J ($\bar{V_1}$ and $\bar{V_2}$) and the auxiliary vector $\bar{V_3}$ are combined to calculate the cross-product vectors ($\bar{V_{13}}$ and $\bar{V_{23}}$). Eventually, the displacement vector from $P_{(i,j,k)}$ to $P_{(i,j,k+1)}$ is computed from the cross-product vectors. The entire procedure is depicted in Fig.1.21.
The repetition of this procedure on all the points of a given surface generates the points cloud data for the next curved slice. This approach works when the toolpath is generated by intersecting a series of parallel vertical planes with the model surface.

Cylinder surface slicing [37]

It start with the identification of cylinder surfaces on which the part is going to be printed. The model is then flattened to use conventional slicing algorithms before reverting perimeters and infills back to the initial domain. Fig.1.22 shows an example of a flat layer (blue color) and its counterpart in the original space (red one) after a proper interpolation to retain the information. It's possible to infer that only one dimension stays unchanged by the mapping function, that is the one parallel to the cylinder axis. The transformation from the conventional layer space (P') to the original (P), having indicated $\alpha = y'/R$, is done using:

$$\begin{cases} x = x' \\ y = \tan\alpha \cdot |\cos\alpha| \cdot (R - z') \\ z = R - |\cos\alpha| \cdot (R - z') \end{cases}$$

Additional comment on this method can be found in appendix A.

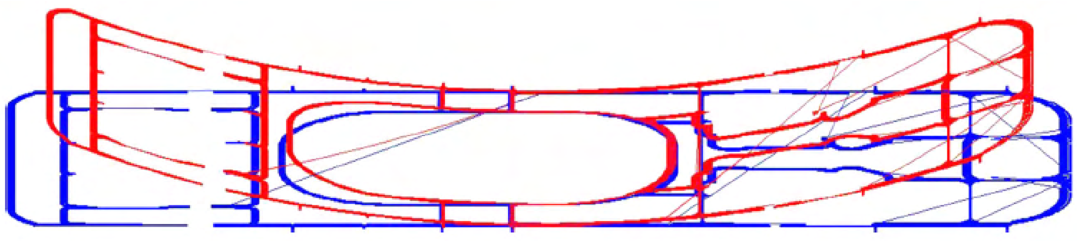


Figure 1.22: Planar toolpaths of one layer and corresponding curved toolpaths [37].

Curved Layer Fused Deposition (CLFD) [38]

The Curved Layer Fused Deposition (CLFD) continuously reorient the nozzle so as it can deposit the filament exactly following the surface curvature. The ideal strategy is to ensure that the extruder axis is perpendicular to the surface all the time, so that the extruded filament can be placed accurately and joined to the former layers applying a vertical force from the extruder tip. This building paradigm for FDM results in better material structure and part strength due to the fiber continuity along the extrusion path.

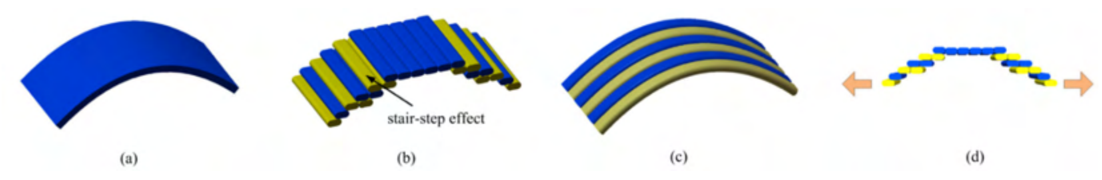


Figure 1.23: Comparison between the surface quality (a and b) and the mechanical strength (c and d) of CLFD (left) and flat FDM (right) [38].

Spherically Curved Layer (SCL) METHOD [39]

The Spherically Curved Layer (SCL) method counts one translational (z) Dof and two rotational ones (ϕ, Ω). This change is due to the adoption of a spherical reference system. The coordinates of a point in the printing volume are now expressed as follows:

$$\begin{cases} x = r\cos\phi\sin\Omega \\ y = r\sin\phi\sin\Omega \\ z = r\cos\Omega \end{cases}$$

This technique overcomes the tool-piece interference issue, avoids the creation of many support structures and is also robust with respect to not differentiable surface points, since the computation of the derivative is not required.
The SCL is sliced with a group of concentric spheres whose radius is defined by the printing tool location. With reference to Fig.1.24, the SCL method can be summed up in three steps:

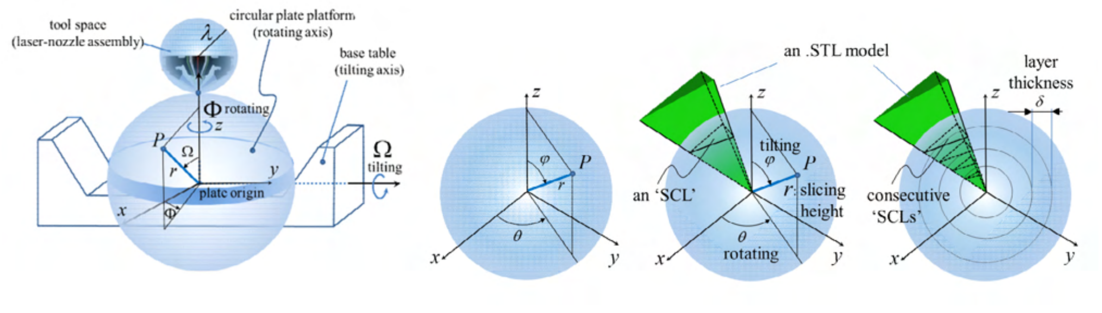


Figure 1.24: The slicing procedure for the SCL method [39].

- The workspace is depicted as a spherical space with radius r from the plate origin. ϕ and Ω are respectively the rotation and tilting angles of the building plate;
- The SCL is defined by a set of 3D vector points connected as a closed loop curve on the surface of a concentric sphere. It is created intersecting triangular facets of a STL model with the aforementioned sphere;
- The SCL model is build accumulating a group of consecutively generated SCLs, each located on the surface of the individual concentric spheres. These are progressively accumulated from the zero ground to a certain height and are defined

according to the number of layers needed for the build of the model. The radius difference between any pair of adjacent concentric spheres is simply the layer thickness of the build process.

As being based on the STL standard, the SCL model is simpler, quicker and even more reliable than other curved layer models developed using the parametric surface. However, an algorithm generating the SCL still requires a sophisticated investigation of all the different cases of the intersection between a triangular facet and a spherical surface.

Thirteen intersections between a triangular face and a sphere have been quoted and

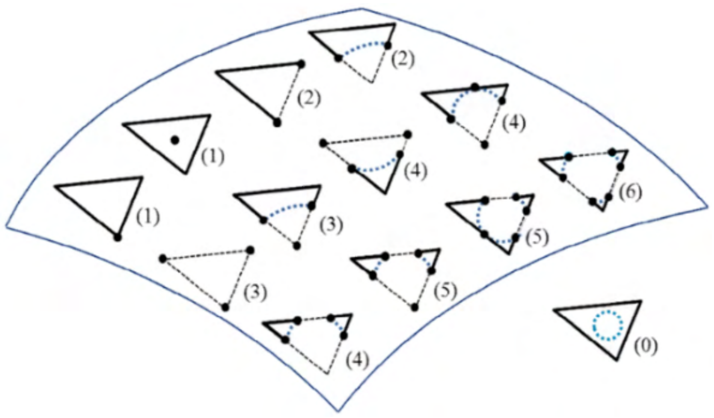


Figure 1.25: Possible intersections of the triangles with a sphere [39].

depicted in Fig.1.25 by the authors of [39]; the number marked on each triangular facet is the number of intersection points between the facet and the slicing surface. Still relative to the same figure, the dotted line indicates a portion of the facet edge submerged inside the slicing surface, while each thick dotted line, arc, or circle is an intersecting boundary between the triangular facet plane and the slicing surface. All the twelve cases drawn inside the surface boundary have intersection points between the triangular facet and the slicing surface. Only one case, drawn outside the surface boundary, has an intersection with no overlapping with either vertexes or edges of the facet (dotted circle).

Reference Surface Offset Algorithm [6],[10]

It is applicable to both planar and curved continuous parametric surfaces. A STEP file results more suitable to accomplish this task due to the capability of detecting the freeform AM features and the built-in analytical formulation of its geometries. If a STL file is provided, the algorithm is still applicable as long as a preliminary step is performed: the discrete STL mesh points must be fit with a B-spline surface. In this way the discrete surface turns into a continuous parametric one, the *reference slicing surface.*

There are two approaches to achieve offset surfaces based on the *reference slicing sur-*

face: the first method is offsetting the surface along the normal vectors directly by a distance equal to the layer thickness; the second generates the offset surface by offsetting the triangles that are used for the fitting process.
Stepping into the mathematics of the first procedure, given a parametric surface $r = r(u,v)$, the offset surface can be computed with the following equation:

$$S_{off}(u,v) = r(u,v) + \Delta h \cdot N(u,v) \quad ; \quad N = \frac{r_u \times r_v}{\|r_u \times r_v\|}$$

Where Δh is the height of the freeform layer; N is the unit normal vector to the surface r=r(u,v); r_u and r_v are the partial derivatives of r=r(u,v) with respect to u and v.
Given that two partial derivatives are required, the singularities onset should be carefully approached especially when there is a concave region on the reference surface, otherwise the offset algorithm may become invalid. Consider the stretched surface of Fig.1.26*a* whose sketch curve C_0 is shown in Fig.1.26*b*. After offsetting the original curve C_0 twice, there is a sharp point at the valley location of curve C_3, which degenerated from the valley of curve C_0.

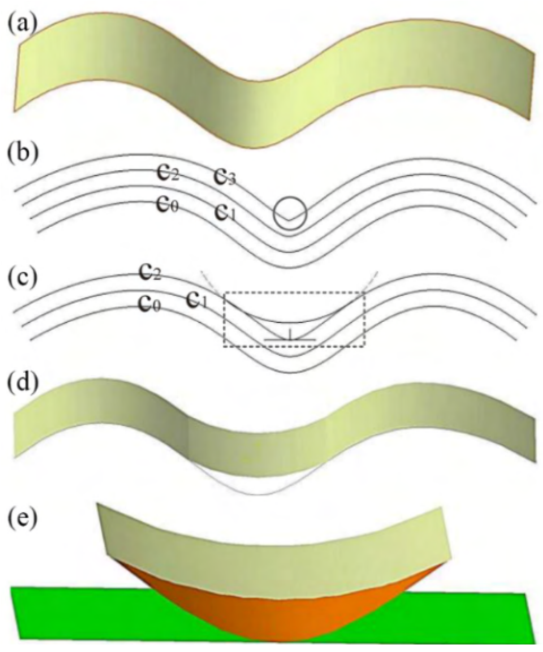


Figure 1.26: Singularity onset in the offset surface algorithm [6].

Under this circumstance, the continuity of the curve declines to C^0, which causes the offset algorithm to collapse in this step. To solve this issue, the surface needs to be modified before it becomes deteriorated: an arc tangential to both sides of the valley curve C_2 is designed to replace the segment of the curve under the arc, as shown in Fig.1.26*c*. The radius of the arc should be large enough:

$$R \geq h_{max} - \Delta h$$

Where h_{max} is the maximum height of all the feature layers; Δh is the height of the current layer.

The result of the modified surface is shown in Fig.1.26*d*. The domain between the arc curve and the original profile is detached into a new feature (the brown one in Fig.1.26*e*). This new feature should be completed before a new layer.
Next, the offset freeform surfaces and the feature to be built are intersected:

$$contour = \{p|p \in (S_{off} \cap f_i\} , i = 1, ..., n$$

having indicated with f_i is the B-Rep surface.

Path planning [6], [38], [40], [41]

When moving from the planar to non-planar layers, the generation of the passes for the infill has to be adapted as well. The possible 2D infill deposition patterns are depicted in Fig.1.27 together with the 3D version. The distance among the points should be

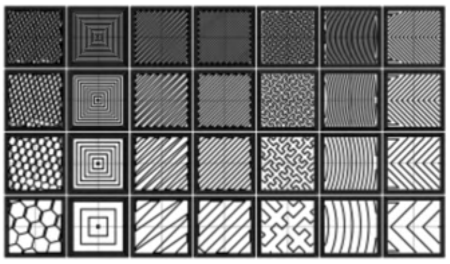
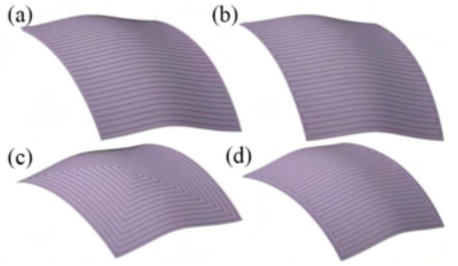



Figure 1.27: Infill patterns for 2D (*left* [42]) and 3D (*right* [6]) layers.

the geodesic distance, namely the geodesic offset on the curved surface. Since the determination of the geodesic distance is not something straight-forward (for the non-planar surface), it is worth to spend some words to understand how to compute it. Given two distinct points p and q in $\mathbb{R}^2$, the so-called midpoint method is an effective and easy way to do; it is based on the triangular inequality that iteratively tries to shorten the path between the two endpoints. It works like this:

- It starts with a first trial approximation $\gamma_1 = p_1...p_n$ made up of a finite set of points in $\mathbb{R}^2$ such that p_i and $p_(i+1)$ are separated by a small distance and the endpoints are $p_1 = p$ and $p_n = q$;
- It computes the midpoints of each segment (p_i, p_{i+1}) and connect them to get a new path. Place p_1 at the beginning of this path and p_n at the end of it: the resulting path is γ_2;
- Repeatedly perform step II for each new γ, until getting a path close enough to the geodesic, which is obviously a straight line in $\mathbb{R}^2$.

While the midpoint algorithm works well in $\mathbb{R}^2$, it does not work on a curved surface S because the midpoint m of any two consecutive points on the path is not guaranteed to be on S. In order to fix this issue, it is necessary to calculate the point *m'* on S closest to m and use that point in the path approximation instead of m. It is important to note that a geodesic is not necessarily the global shortest path between p and q: the algorithm finds the shortest path in a neighborhood of the original path, but the structure of the surface could prevent it from carrying out the global shortest path.

Once chosen the infill pattern, the exact profiles to be followed by the printing head for each layer is computed. The first check to be done regards the presence of cusps and self-intersections in the toolpaths generated by the offsetting method: whenever the offset distance is larger than the minimum radius of curvature of the surface, a self-intersection in the path is likely to occur. In such cases, it must be removed, and the new surface offset is obtained applying continuity conditions and smoothening.
The only way to translate the aforementioned roughness and strength requirements into a mathematical formulation, is the realization of a uniform and constant contact cord (CC) between the adjacent printed filaments. The model discussed hereafter is carried out assuming that the extruded filament is of circular cross-section with a constant diameter throughout the process. This is not completely true because it is not shaped only by the nozzle tip, but also by the spreading behavior during the extrusion and the deposition processes. Therefore, the final shape of the deposited filament can be described as oval rather than simply circular.
According to the model, the filament cross-section locations are determined by two sequential steps:

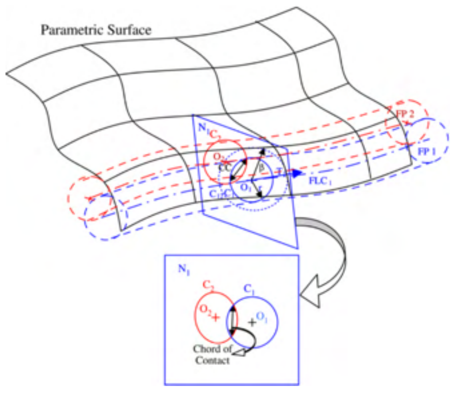


Figure 1.28: Initial guess [40]

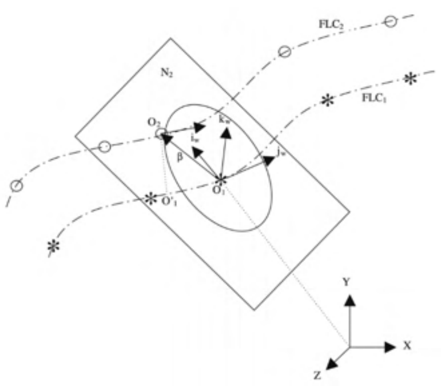


Figure 1.29: Point adjustment [40]

- *Initial Guess Point Determination*: A circle C_1 with center O_1 and radius r is drawn to represent the circular cross-section of filament path FP_1 at O_1. T_1 and N_1 are respectively the tangent and normal plane to filament location curve FLC_1 at point O_1. Now a circle C_{12} with center O_1 and radius $\beta = \sqrt{4r^2 - c^2}$ (c is the desired chord of contact) is drawn on N_1 to cut the Offset Surface (OS) trace on N_1 at O_2. The circle C_2 with center O_2 and radius r on N_1 has the required superposition with C_1 (Fig.1.28). However, this procedure constrains the adjacent filament cross-section C2 to be co-planar with C1. Although it is not the right solution, it can be adopted as an initial guess point for an iterative process seeking the correct adjacent filament cross-section location.

- *Point Adjustment*: The circular cross-section of FLC_2 with center at O2 lies on a plane N_2 which is normal to FLC_2 at O_2. The intersection of FLC_1 with N_2 yields O_1' . If $O_2O_1' = \sqrt{4r^2 - c^2}$, then O_2 is accepted as a correct FLP_2, otherwise several iterations of the procedure are carried out with the radius of circle $C_{12} = \sqrt{4r^2 - c^2} \pm \Delta r$ where Δr is a small fraction of $\sqrt{4r^2 - c^2}$ (Fig.1.29).

1.3.4 Multi-directional slicing

The last building strategy, and therefore slicing paradigm, treated is the multi-directional one. It consists on the realization of an object by identifying sub-regions that can be approximated by 2.5 object.
The effective exploitation of a multi-directional deposition system goes through many steps which have to be performed accurately. The entire process can be recapped in the following key tasks:

1. Part volume decomposition;
2. Build directions determination;
3. Sub-volumes deposition sequencing;
4. Slicing and Path-Planning.

Only the first three will be deepened, being the fourth subjected to the same principles discussed up to now.

1. Part Volume Decomposition [43],[44]

The strategy used in multi-direction slicing is to subdivide the part P into volumes V_i that can be built keeping a certain direction b_i constant, belonging to a set B of build directions. The first build direction is assumed to be already defined by the print bed; the other direction needs to be evaluated according to the geometry. Volume decomposition deals with the identification of those features which cannot be built along a selected direction.
Given a build direction $\overline{B}$ and a point p on the boundary ∂P of P, if the angle between the surface normal N(u,v) at p and $\overline{B}$ is greater than a process specified angle, then P is unbuildable along $\overline{B}$ such as all the points on ∂P in the ϵ-neighborhood of p. The set of all these points defines the unbuildable surface features (Fig.1.30).

Aiming at identifying the boundary between the buildable and unbuildable surface features, the isocline of ∂P, that is the locus of points on a surface S(u,v)$\in C^1$ for which the surface normals have a constant slope with a given vector V, are exploited. For a generic surface S(u,v), isoclines either form closed regions on the parametric surface or start and terminate at the boundary of this parametric domain. As a result, if the given vector V is parallel to the build direction $\overline{B}$, the isoclines for a given maximum angle variation delineate the surface into regions R_i.
Once identified the surface features which cannot be built, the second task in part

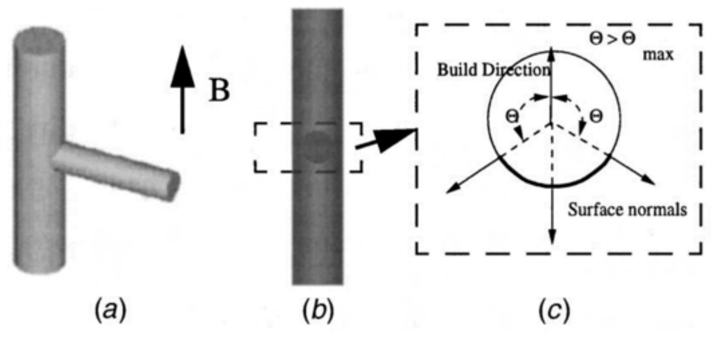


Figure 1.30: Definition of the buildable and unbuildable surface features [43].

volume decomposition is to determine the unbuildable sub-volumes of the part. Possible algorithms are available in literature. In Fig.1.31, it is presented the silhouette edge-based decomposition; it checks the angle between the building direction and the surface normals so as to determine the points where there is a variation. To do so, a plane orthogonal to the build direction and passing through the lowest (minimum height) of the selected points slices the part creating an intersection area used to obtain a tool body open on both ends. The tool body is then subtracted from the solid model to separate the overhanging features from the solid model.

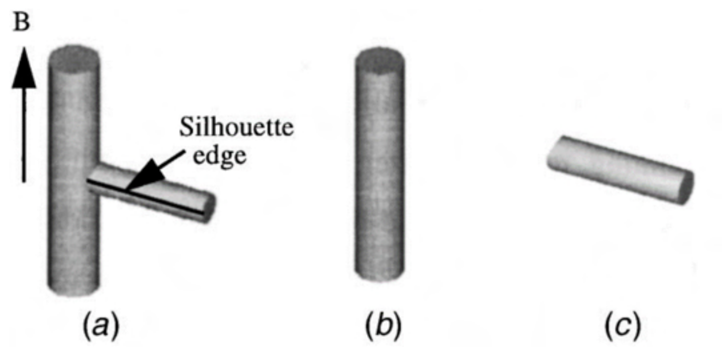


Figure 1.31: Silhouette edge-based decomposition [43].

2. Build Direction Determination [43],[45]

Once the entire volume P has been split in sub-volumes, a build direction has to be selected for each of them. The determination of the direction exploits the notions of Gauss and build maps.
The Gauss map *G(S)* is a function N from an orientable surface S in $\mathbb{R}^3$ to the unit sphere S^2 in $\mathbb{R}^3$:

$$N : S \to S^2$$

The function samples the surface boundary at discrete intervals and associates each normal vector *N(p)* orthogonal to the p-th point of the surface S to the unitary normal vector *N(p)* departing from the center O of the unit sphere S^2.

If the ϵ-neighborhood of the point p can be approximated by the tangent plane of the surface, the given definition of the build map can be extended to define the set of feasible build directions for this plane. The build map of a surface S is then obtained by intersecting the build maps evaluated at each point $p \in G(S)$. The build map is non-empty only in the neighborhood of the base surface shared with the adjacent sub-volumes as shown in Fig.**??**.

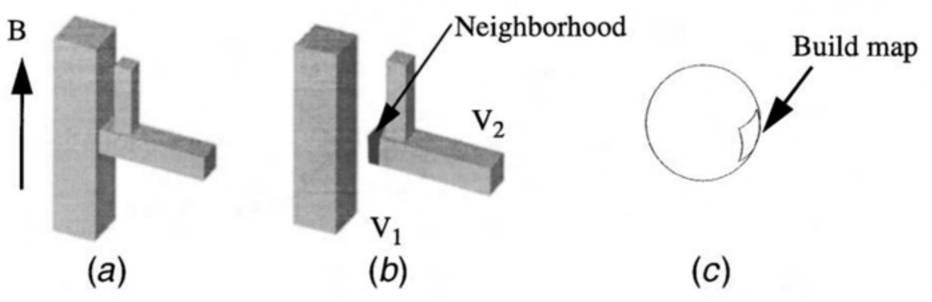


Figure 1.32: The build map is non-empty only at the interface between the sub-volumes V_1 and V_2 [43].

3. Sub-Volumes Deposition Sequencing

The deposition sequence is constrained by the base surface existence and the need of collision avoidance.
The key-tool to determine the correct deposition sequence is the decomposition tree carried out by the part volume decomposition. If the existence of the base surface is the only constraint, the processed volumes are sequenced in the ascending order of their depths in the decomposition tree so that the volumes at depth i lie on one or more processed volumes at depth i-1. Eventually, to account for collision avoidance, two or more sub-volumes are allowed to be deposited simultaneously.

Tool - Piece interference evaluation [46] The problem of tool accessibility considers the detection of global gouging by looking for possible collisions between the printing head and other surfaces. It can be formalized as follows:

Given a surface S(u,v), a vector field O(u,v) prescribing the orientation of the tool at each point p(u,v) on the surface and a check surface K(s,t), then find all the regions in S(u,v) that are O(u,v) accessible with respect to K(s,t).

The proposed approach provides a global solution instead of a point by point local validation which can't guarantee gouging lack in between the verified points. The algorithm is based on the definition of a regular surface in a parametric form: $S : D \subseteq$

$\mathbb{R}^2 \to \mathbb{R}^3 | S(u,v) = \Psi(f(u,v), g(u,v), h(u,v))$ that is regular if the following conditions are met:

- the components $f, g, h \in C^1(D)$;
- the function Ψ is injective;
- for every point *p(u,v)* exists a tangent plane to the surface *S(u,v)*.

Once defined $O_1^n(u,v)$ and $O_2^n(u,v)$ (two vector fields spanning the plane orthogonal to O(u,v), the tangent plane of S(u,v), in $\mathbb{R}^3$), it is possible to state the necessary and sufficient condition to find the boundary of the accessible regions of S(u,v):

"A(u,v,s,t) point is on the boundary between O(u,v)-accessible and O(u,v)-inaccessible regions of S(u,v) if and only if the $F_i(u,v,s,t) = 0$ for i = 1,2,3 "

Where:

$$F_1(u,v,s,t) = \langle O_1^n(u,v), S(u,v) - K(s,t) \rangle$$
$$F_2(u,v,s,t) = \langle O_2^n(u,v), S(u,v) - K(s,t) \rangle$$
$$F_3(u,v,s,t) = \left\langle n^k(s,t), S(u,v) - K(s,t) \right\rangle$$

These are necessary and sufficient conditions completely constraining the tool's orientation. Since the system presents three equations in four unknowns, the solution is expected to have one degree of freedom: it delineates accessible region in S(u,v) from the inaccessible regions.

1.4 Devices in AM

In the last section how the 3D model is processed to infer the path, according to the different building strategies, has been the subject. It remains to address the systems that actually use this information to physically construct the part.
Before going in detail of the kinematics solutions used in AM, the definition of *robot* is due. According to Robot Institute of America (1979) a robot is:

"*A reprogrammable, multi-functional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of task.* "

The ISO 8373:2012 also states:

"*[a robot is an] actuated mechanism programmable in two or more axes with a degree of autonomy, moving within its environment, to perform intended tasks. A robot includes the control system and interface of the control system. The classification of* *robot into* industrial robot *or* service robot *is done according to its intended application.*"

The standard then goes on defining *Industrial robots* as "*automatically controlled, reprogrammable, multipurpose manipulator, programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications.*" and *Service robots* as "*robot that performs useful tasks for humans or equipment excluding industrial automation applications (Industrial automation applications include, but not limited to, manufacturing, inspection, packaging, and assembly).*".

The device required to trace with the EE (End Effector) the path yielded by the slicer can be therefore classified as a robotic system.

1.4.1 3-Dofs machines

Most of the commercially available AM machines are three-axis Cartesian robots. They are characterized by a series of mutually orthogonal prismatic joints that can be also placed above the workspace (in this case they are called gantry robots) [47]. It's considered the simplest kinematic chain based on the fact that the robot does not alter the metric of the operational space with respect to the one of the joints. In other words, the Cartesian robot is the realization of a linear function between the two spaces. Linear axes are obtained using belt system or ball screws.

The second most widespread solution (specially in FDM system) uses a robot belonging to Parallel Kinematic Machine (PKM) family that is the Linear Delta. It is constituted by three $\underline{P}$RPaR (actuated Prismatic - passive Revolute - Parallelogram - Revolute joints) serial chains, also known as legs, connected to a fixed and mobile base; the Prismatic joints are usually vertical and parallel so that the vertical motion is the result of the same displacement of the three linear axes [48]; the connection of the last revolute joint to the mobile base constrains it to only translation movement. Although it has a lower ratio between the operation space and structure space than the Cartesian (the same hold for all PKMs compared to Serial machines), it has higher precision at the TCP.

These systems are mastered by suitable controller. The hardware has to control the 3 motors for the XYZ positioning plus the one (or more) of the tool. For stepper motor, it is translated in the generation of the stepping sequence at a frequency that yield the desired number of steps per cycle. For servomotors, angular position is measured to close the feedback loop. The motherboard has to feature also all the components necessary to carry out the process as engineered (managing sensor, tools, power supply and the like). Example of such motherboard is the RepRap Arduino Mega Polulu Shield (RAMPS) that integrates the Arduino Mega board (CPU plus memory) with

all the electrical component to control an FDM printer.
The software needs to be able to parse G-code lines and translate each instruction block into feasible trajectories that the actuated axes will follow (stepper or servomotors) while maintaining the process parameter. The microprocessor has to have discrete CPU frequency (starting from 16MHz, that is the one of an Arduino Uno board, ATmega328, used for low-cost projects) to be able to manage the different phases as close as possible to parallel task (if not feasible by the IC), that are:

1. Read incoming data
2. Interpret a single line
3. Plan the motion (Look Ahead)
4. Execute next motion block from the buffer
5. Read sensors to close relative control loops or halt

For those interested to thoroughly study and understand these aspects refer to the documentation (online) about *LinuxCNC*, *Repetier-Firmware* and *Marlin* firmware that are open-source software widely used in low-cost and industrial applications.

1.4.2 Higher-Dofs machines

The integration of higher-DOF robot systems with AM technologies offers the possibility to fabricate complex geometries with a significant reduction of support structures and with patterns that could improve its mechanical properties. Currently, leading researchers worldwide are mainly focused on incorporating "Extrusion based processes", "Photo-polymerisation processes" and "Direct-Energy-Deposition processes" into multi-DOF kinematic mechanisms to gain more flexibility in terms of the orientation of TCP for deposition.
In literature several examples of such systems can be found:

- [49] A group of researchers, in collaboration with ABB research center, developed a cell with A 6-DOF robot arm coupled with extrusion printing head to build complex geometrical objects combining ABS and Carbon fibers. For the process fabrication simulation, optimization and visualization, the ABB native software platform, Robotstudio, is employed.

- [50] A 6-DOF UR3 robotic arm is employed to produce a polymeric part using PLA with almost zero-support structures. Support-free structures are developed by decomposing the primary model into multiple sub-models with specific building directions and printed in a collision-free sequence (Fig.1.33).

- [51] reports a new AM method using a robotic arm in which 3D objects on any given working surface by using rapidly hardening thermo-set resins in combination with innovative extrusion technology: double-curved lines of varying diameter without the need for support structures can be 3D-printed, Fig.1.34.

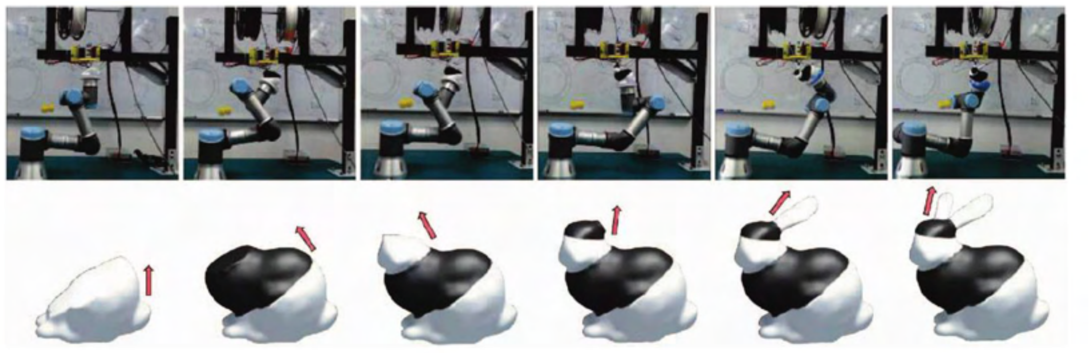


Figure 1.33: The progress of using our 6-DOF UR3 robotic arm to fabricate a free form model-bunny. Different parts of the model are fabricated along with different directions (see the red arrows in the illustration), and filaments in different colors are used for making different parts [50].

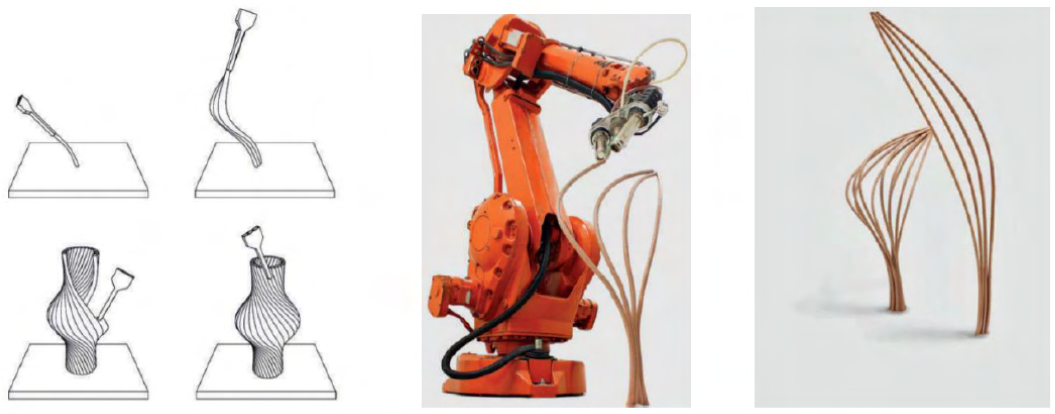


Figure 1.34: (left) The technical setting for Anti-Gravity AM. (Center-Right) Surfaces and objects are formed by combining 3D curves instead of successive 2D layers [51].

- [52] 6-DOF UR5 arm with a worktable fixed on the end effector and is coupled with a fixed extrusion system to take advantage of gravity. The main motivation for adopting this kind of machine setup is to minimize collisions with the part or robot during operation. A similar example is discussed in [53].

- [54] In the field of architectural design and construction, several robotic-assisted extrusions AM systems have been extensively explored due to the size and shape of the building elements. [55] deal with the production freeform thermoplastics shells.

- [56] reports of industrial robots to print concrete (Fig.1.35), while in [57] small mobile robots and cable-driven robots are being tested for the same goal.

- [58],[59],[60] report the use of robots, such as AGV (Automated Guided Vehicles), integrated with industrial robots to increase the workspace. The process could

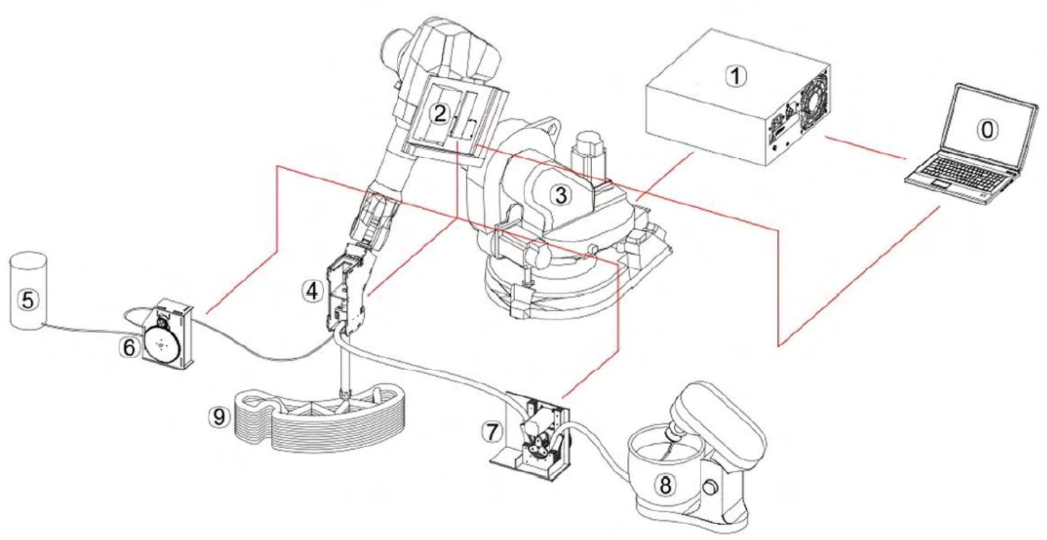


Figure 1.35: Schematic of the 3D printing setup for building concrete blocks: 1. Robot controller; 2. Printing controller; 3. Robotic arm; 4. Printhead; 6. Peristaltic pump for accelerating agent; 9. 3D printed object [56].

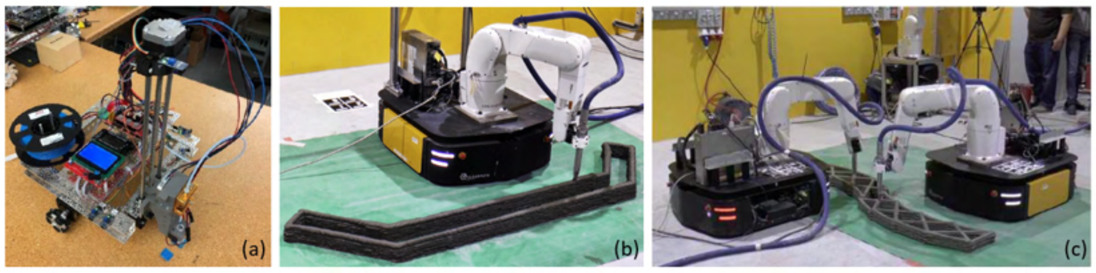


Figure 1.36: (a) example of a 3d printer cart for FDM [58]; AGV and manipulator for concrete AM production with a single unit(b) or multiple (c) [59],[60].

be undertaken by one single unit or by a swarm of mobile robot arm as can be seen in Fig.1.36.

1.4.3 Considerations on the use of machines with more than 3 Dofs

Due to the introduction of higher Dofs kinematic configurations, the construction of this system becomes more complex. For instance, there are specific positions in which workpiece can be operated with good performance considering velocity ellipsoids in the work-volume. Moreover, it has already been discussed the modifications required in the process planning and in generating and adapting the slicing technique.
These capabilities of AM with multi-DOF robotic configurations that can be expanded in terms of functionalities can be achieved by considering these following characteristics [61]:

- *Composite material components*: multi-DOF kinematics and multi-material extrusion heads can be used to build objects made of multiple materials and complex shapes. Efficient task planning, scheduling and motion planning must be established to have multiple extrusion heads cooperate at the same time.

- *Ultra-scale parts with small scale feature*: large components, such as turbine components for Aeronautical applications, are nearly impossible to build by employing conventional AM process. Moreover, the cost of the process increases exponentially in terms of time with part scaling. In this aspect, robots with more than 3 Dofs are employed to manufacture these class of parts. Besides, multiple robots can be adopted for building the same components decomposing the primary component into multiple secondary parts.
- *High accuracy*: Tolerance limit is of high importance in the manufacturing process, generally AM processes doesn't yield good tolerance. Consequently, conventional NC machining is employed for achieving good accuracy of the building geometry. Alternatively, closed-loop control of the robot with laser tracker can be adopted for achieving good accuracy of the build, in this way post-processing can be eliminated. Very importantly, the optimization of trajectories must be employed for a defined motion law by imposing kinematic, technological constraints.
- *High build speeds*: As most of the conventional AM techniques are much slower if the production is scaled up. However, utilizing conformal layering can reduce significantly the overall build time.
- *In-situ printing*: Robots are relatively easier for portability (this holds for civil AM). In an industrial context, the corresponding robot can be utilized for performing other mechanical processes such as milling, grinding etc.
- *Hybrid manufacturing*: Conventional AM techniques yields poorly finished product in terms of surface finishing for instance. Thus, post-processing has to be carried out by using a conventional NC machining process. By employing both AM extruder and machining heads, lead time required for transferring the part between these two systems can be eliminated. However, realizing a hybrid cell requires addressing complex process planning and vibration resistance is a significant parameter to be considered for robot milling.
- *Repair*: The possibility to repair a damaged component is nearly impossible to perform by employing conventional AM systems due to its immobility and inflexibility. These barriers can be overcome by utilizing kinematic systems with more than 3 Dofs.

1.5 Challenges of AM

Research in the field of Additive Manufacturing is increasing at a rapid pace; the intent is to utilize this technology to realize complex functional geometrical objects to overcome shortcomings of Subtractive Manufacturing, SM. AM, as promising as it is, has several known issues (some of which already introduced in the section about AM) that also represent the main challenges to overcome in this field; they are here briefly explained [13], [62], [63], [64], [65].

Multi-material printing It is not a viable option in conventional AM systems. AM system can produce a variety of materials for building homogeneous components, but not for heterogeneous materials. There are two main barriers in printing multiple materials, complexities of modeling and realization of the designed model.

AM methodologies There are mechanical and physical limitations induced by layered manufacturing such as anisotropy (Fig.1.37). The final part will be able to bear higher loads applied in the layer plane than along the printing direction. Another issue is an inaccurate approximation of a model for layers, in general, they are assumed to be rectangular but in reality, layer cross-section has parabolic curvature at the vertical edges. This cause the hull to be characterized by notches that will promote two layers to separate, thus enforcing the weakness along the printing direction.

Pre-processing It is comprehensive of the methods to decompose and prepare the CAD model for the next phase, such as optimal orientation, in order to yield sensible slices of the model (and subsequently trajectories) and limit the generation of supports. Before initializing the actual printing process, these objectives must be defined methodologically.

Part orientation It is a significantly important process parameter which has a direct correlation with build time, surface finishing and mechanical properties of the object

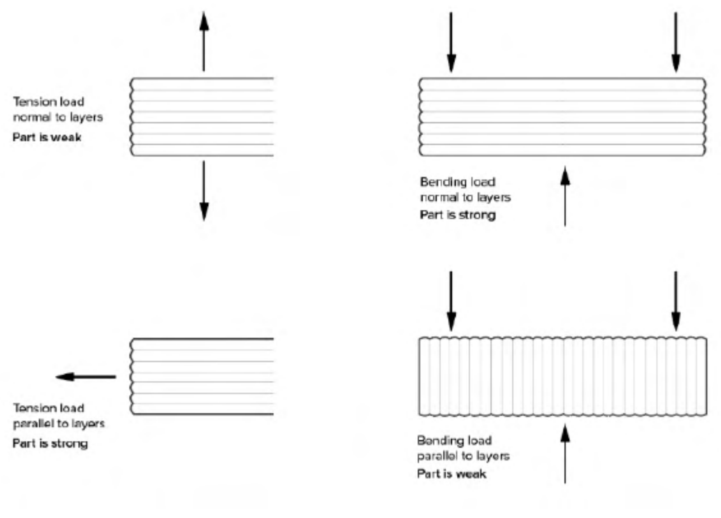


Figure 1.37: A beam object built-in horizontal and vertical build direction [66].

(Fig.1.38). Moreover, with flexibility in adopting multiple orientations by the AM system, support structures required for building the object can be minimized to an absolute minimum (i.e. no need for supports).

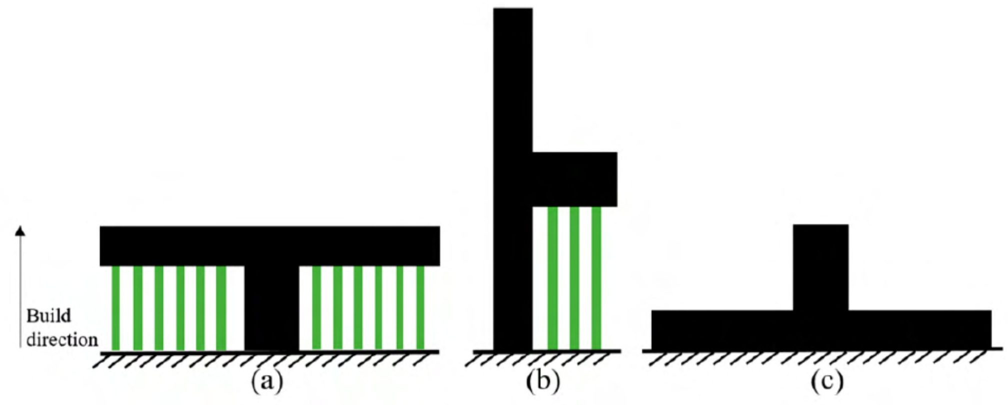


Figure 1.38: Three different orientations and its corresponding support structure required for a successful deposition [67].

Speed With the increase in complexity of CAD/freeform model, the complexity of the process in terms of deposition and building time increases.

Slicing It is one of the important aspects of AM process planning. By default, uniform slicing is used in the conventional AM systems. Due to this constraint, staircase effect which is stepped edges caused by 2.5D contours and containment problem (which is when slice either falls inside or outside the original model). Moreover, the slicing process is directly related to the build time, accuracy and surface finishing. Another approach of slicing used is to slice the CAD model directly by using numeric geometrical functions such as NURBS, but it's computationally expensive, size of the file increases as well with the increase in complexity of geometry.

Shape optimization It deals on how the material is allocated in the design space (area within the boundary of the model that can be modified). Optimization of this area has a direct correlation with the reduction of materials, production time, energy consumption and environmental costs. To acquire these benefits, the way in which this area is filled has to be optimized accounting for strength, mass and volume parameters. There are two ways in which the allocation of material can be estimated, they are: predetermined geometries (such as, honeycombs, lattices and other repeatable shape elements) or topological optimization tool. Topological optimization has notable advantages over predetermined cellular structures because it requires know-how of material and part usage, but this is also is major issue.

Process planning errors They are the errors caused before and during printing. There are quite a few significant occurrences of errors due to tessellation, slicing and orientation (staircase effect and support for instance). During printing, the speed of movement of TCP must be synchronized with the material extrusion, error in the actual speed profile can cause reconstruction errors that, even if small at first, can be propagated through the layers causing bigger and bigger defects. Generally, most of the conventional AM systems doesn't have a functionality to regulate appropriate inputs to nullify process errors (by means of process monitoring) such as over or under extrusion.

Post-processing To achieve a required accuracy of the printed object, additional operations have to be performed on the built. In FDM AM process, post-processing operations involve sanding, bead blasting, traditional machining and acetone finishing.

Hardware and maintenance issues Generally, the AM system must be set up with proper process parameters for effective functionality. Additionally, energy consumption, material consumption and other specific constraints must be postulated based on the available know-how of the entire system. Based on the material that is being used for building an object, appropriate adjustment to the extrusion system must be set up. Moreover, as in any manufacturing system, periodic maintenance must be carried out (for example, in FDM system, it avoids blockage of nozzles).

Additive Manufacturing in Industry 4.0 [68], [69] The fourth industrial revolution aims at customization, profitability, efficiency and performance by combining the digital and physical world. 3D printing, Internet of Things and cloud computing are few of the tools that Industry 4.0 exploits to perpetrate its goal.
AM is appealing to such philosophy because it can, on one side, meet customer requirement at an optimized cost and in a fast and effortlessly manner and, on the other side, be used to develop innovative product. Moreover, it can combine different materials with specific properties (for instance smart materials) and directly add electric and hydraulic circuit to make ready to use parts.

1.6 Conclusions

In this chapter the 3D printing technology has been dissected according to the building principle as well as the building material. Focus is given to the phases of the process shared by all the AM families. The main aspects, that need to be comprehended, are: common types of geometries (both in the physical world and their digital representation), slicing (generation of the toolpath starting from the intersection between the digital model and the building surfaces) and printing machines (i.e. robotic systems). The relationship among these three elements determine the overall quality of the yielded component (mechanical and aesthetic) and of the process (scraps and lead time). Id est that, for a specific geometry, exists an optimal combination between slicing method

(as a consequence of the right building strategy) and suitable end-effector placer able to overcome the possible defects (anisotropy, staircase effect and such).
3-axis machines can only be adopted for parallel planar building (also referenced as cutting) planes suitable for 2.5D geometries; the further the shape departs from it, the greater the chances are that the part will require the optimization of its placement on the substrate, the generation of support structures and that staircase effect will be present in the final built.
5-axis machines can be used for all the types of cutting surfaces presented and, therefore, for both 2.5D and freeform geometries at the expense of an increase in cost of the machine per workspace volume; supports and staircase can be eliminated or partially mitigated.

Hence, it has been decided to investigate the possibilities and limits of a printer based on a 6 Dofs industrial robot. This requires the realization of the testing cell in combination of a CAM software able to generate suitable trajectories. Among the seven technologies, FDM is considered due to its relative simplicity and safety. Therefore, the resulting main elements of the test rig are an FDM extruder, an industrial manipulator and a trajectory planner that are going to be addressed in the following three chapters. Focus is put on the individual system (software and hardware, if present) as well as their interconnection, testified by manufacturing tests on the setup.

Chapter 2

Extrusion system

Among the AM technologies, the simplest one to use in order to test new building strategies and production systems is FDM (Fused Deposition Modeling). FDM governing principles are inquired at subsection 2.1.1.
Extrusion is the process of melting material and pushing it through a nozzle. The raw material is either in the form of pellets, feedstock or filament. In the industrial field, it is exploited primarily by Injection Molding and 3D FDM printing. In the former, the molten material is used to fill a machined die, while in the latter to transfer passes of material.
Several extrusion devices are purchasable on the market with the possibility to customize the end product to user specifics (e.g. material properties and size, operating power supply and maximum temperature). Controllers, on the other end, are usually close systems developed to also deal with the G-code interpretation and motion planning of the entire 3D printer.

In this chapter the development of a "Direct" filament extruder is discussed (section 2.1). Selection and integration of the hardware components (such as feeder, heat sink, sensors, control boards), able to reflect the identified process parameters, is the initial step (subsection 2.1.3). In order to have at disposal a versatile tool to comply with different systems (easily interchange components, code and machines during the bring-up and experimental phases), it ought to receive instruction over an Ethernet port, so that could be easily connected to any robotic system (corroborated by section 3.1 and 3.2) or a LAN. Moreover, a firmware is coded to master and direct the hardware, such as motor and temperature control and device communication (subsection 2.1.4).
In section 2.2 a review on additional extrusion systems (syringe and screw) used for FDM system can be found. In section 2.3 it is briefly presented, as an example, a 3d printer used also during this research period that exploits feedstock (plunger-based) instead of a filament to point out its differences.

2.1 Filament Extrusion System

This section deals with the design of a direct FDM extruder able to process material cartridges in the form of filament spools (”Direct” means that the control of the filament speed is done close to the hotend). For clearance’s sake, the discussion is split in three parts: the first discuss more in detail the FDM technology, while in the remaining two the hardware and software aspects of the device. These topics are trailed by the design of the flange to fix it on the robot wrist.

2.1.1 FDM governing principles

The FDM (Fused Deposition Modeling) system is made up of an extrusion system (the most common being a filament-based extruder) and a machine that controls the relative movement between the print area and the nozzle. This means that the extruder could be either moved with respect to the substrate, vice versa or split between the two. The

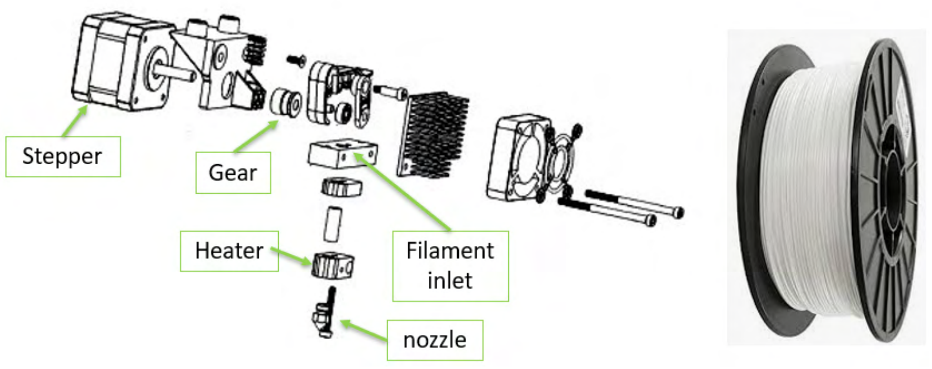


Figure 2.1: Example of a filament extruder for FDM

choice depends on many factors with dynamics playing an important role.
The deposition starts with the filament (wrapped on a spool) being pushed by a gear (directly connected to a stepper motor or with a gearbox in between) inside a heat sink towards an heater; here both the hot source and temperature sensor are found [70], [71].
The melted wire exits through a nozzle that defines the material flow that the print area sees (Fig.2.1).
The extruded material, in air, maintains the section of the nozzle (although some vena contracta followed by expansion is to be expected). If the material hits a plate, then the resulting geometry resembles the one in the Fig.2.2-*left* (where the section depends on the nozzle diameter d_n, the gap h, and the volumetric flow Q_n) providing that the width w is less than the outer diameter of the nozzle D_n. If this is not the case then the actual width might differ; the lateral edges will tend to flow along the nozzle sides reaching an height above the one of expected for the layer instead of spreading on the substrate (the layer will have a greater height and it will be characterized by furrows

left by the nozzle outer shape).
The FDM process can be therefore summed up into the synchronization of the flow rates at the nozzle outlet with the one expected on the print area. In mathematics language, it is translated into:

$$A_{wire}V_{robot} = \frac{\pi d_n^2}{4}V_{extrusion} = \frac{\pi d_f^2}{4}V_{feed} \tag{2.1}$$

Equation (2.1) represents the relationship among the flow rates at the filament, nozzle and print bed under the assumption that the material density, ρ, is not significantly affected by the extrusion process [1]. The cross section of the deposited material is computed as:

$$A_{wire} = wh - \left(1 - \frac{\pi}{4}\right)h^2$$

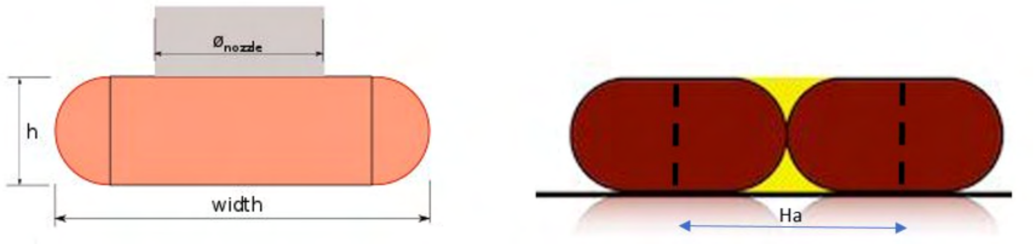


Figure 2.2: Extruded wire cross-section [72].

When computing how the layer will be realized, once defined the outer perimeters, it is crucial to decide how two neighbor passes will be handled. Two tangent wires will resemble the drawing in Fig.2.2-*right*, leaving the yellow area unfilled. CAM software compensates this by adjusting the distance (hatch distance, Ha) so that also this region will be filled (although is quite difficult for the material to flow completely below the previous one). As concern the hatch distance, it can be written:

$$Ha = w - (1 - \alpha)h$$

where α can span from $1 - \frac{w}{h}$ (100% overlap) to infinite. The significant cases are:

- $\alpha = \frac{\pi}{4}$: $Ha = w-(1-\frac{\pi}{4})h$, the yellow region in Fig.2.2 is filled, that has a surface of $(1-\frac{\pi}{4})h^2$;
- $\alpha = 1$: $Ha = w$ that represents two wire juxtaposed;

An empirical minimum for α is $\frac{pi}{4}$ since it will tend to close completely the gap between two neighboring wires. The greater α, the lower the build density.

In order to achieve acceptable results, the material needs to be extruded at a specific temperature and onto a surface with a desirable temperature. This last concept can

[1]If $\rho \neq const.$, it would not be possible to simplify the continuity equation to (2.1) as it has been done.

prevent the part already printed to shrink and distort while cooling or upon completion. This phenomenon is kept at bay by keeping to a minimum the temperatures between the part and the surrounding environment and by also providing a controlled and gradual cooling process. Table 2.1 reports some printing parameters used by the most widespread materials [4].

Table 2.1: Process temperature of the main FDM materials

Material	**Extrusion Temperature °C**	**Heat Bed Temperature °C**
PLA	200-230	50-60
PETG (PET, PETT)	220-250	50-75
ABS	240-260	80-110
NYLON	235-260	~100

Bonding between the layers and the material on the heat bed (first layer) is a crucial aspect. Concerning inter-layer bonding, sufficient residual heat energy is required to activate the surfaces promoting the bonding. If there is a lack of energy, the regions may adhere, but a distinct boundary between the two layers will be formed that can be easily separated. An excess of energy may melt the substrate, resulting in a poorly defined part.

Adherence on the heat bed relies on other principles [73], [74]. Adsorption theory states that adhesion is the result of molecular contact between surfaces and the resulting attractive forces (van der Waals). To obtain this kind of forces, an intimate and continuous contact with the substrate surface is required ("wetting"). Good wetting is obtained when all the valleys and crevices on the substrate are filled by the adhesive; on the other hand, poor wetting when air pockets are left (Fig.2.3). Therefore, good wetting is obtained adjusting the initial layer height and flow rate to yield a suitable pressure that assures the match between the roughness of the printing surface and the added material.

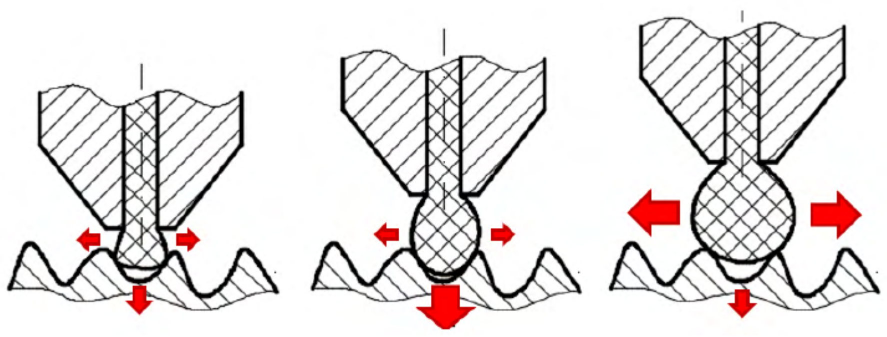


Figure 2.3: Wetting as a function of layer height [73].

2.1.2 Filament-Based Extruder types

Filament extruders can mainly be divided into two types: *Direct* and *Bowden*. The fundamental difference between these two is that the *Direct Drive* has the motor-pinion unit connected directly to the hotend, while in the *Bowden Drive*, it is placed detached from the hot end (in a non-moving space on the printer) and the filament moves inside a PTFE tube (resembling a Bowden cable) [75].
These extruders are small in size and quite precise, but they have some drawbacks. Great attention must be paid to the thickness of the filament, which must not be too thick or too thin; the thread can get tangled at the spool; and crucial is the temperature control (heating and cooling system) of the filament throughout its passage inside the device [71].

Pros of Direct Drive:

- The material has less distance to travel because the Direct Drive is mounted above the hot end. It is more responsive.
- Less force is required from the stepper motor compared to the Bowden extruder.
- Wide range of materials: certain materials with the Bowden Drive extruders have a worse result.

Cons of Direct Drive:
The direct drive's disadvantage is the weight, because when the extruder is mounted on a moving axis of the printer causes in some cases oscillation of the frame.

Pros of Bowden Drive:

- Weight is reduced since that the extruder is removed from the printhead.
- Prints are more accurate and faster.
- Bowden extruders take up less space and are more compact than Direct.

Cons of Bowden Drive:

- The feedback distance is greater.
- The motors are more powerful to compensate friction in the forwarding of the material.
- Friction leads to slower response times and a lack of responsiveness.
- Increased resistance.
- Flexible and abrasive filaments are not recommended as the consume the PTFE tube.

2.1.3 Hardware

The tool is constituted by three functional blocks: material supply, heating system (see Fig.2.1) and control board.
The hotend (made by a nozzle, a heating block and a heatsink) has been selected according to the maximum extrusion temperature among the possible materials that are going to be used during this research that is 285-295°C belonging to a glue (see the industrial case study reported in 7.1).
The material flow rate is controlled by a stepper motor (NEMA 17, 400 Steps Per Revolution, 0.34 Nm Stall Torque) coupled with a 3:1 speed reducer that directly drives the filament (3mm in diameter) to the hotend.
The control of the extrusion system is achieved by an Arduino Mega based board with RJ45 connector and Ethernet chip, combined with a shield (RAMPS 1.6) featuring a stepper driver and a MOSFET connected to a heater cartridge and a temperature sensor. It's possible to see the interconnection between the parts in 2.4.

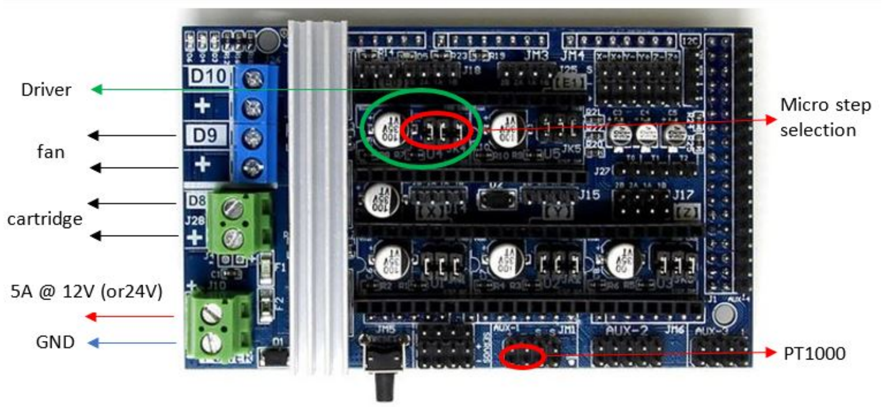


Figure 2.4: extruder wiring using a RAMPS 1.6 shield

Let's now address the single components to understand their function and how to control them (it will be detailed in the next subsection).

Hotend and mechanical parts The gearbox and material feeding system is purchased from catalog ("Titan Extruder" by *E3D*). The presence of the 3:1 speed reducer allows to yield higher thrust force on the filament (or same, but with a lighter motor) and a discrete resolution without using large microstepping (that cause a reduction of the torque available).
The hotend has to bear 300°C, so an all-metal design is mandatory (a "V6" hotend is bought also from *E3D*). The original heat block is replaced with a copper plated one that is able to withstand up to 500°. Also, the nozzles are selected to easily operate at this temperature. The heat source (a heater cartridge) of 40W running at 12V is used. A fan is put on the heatsink to guarantee that the filaments heats up only in the last part of the hotend, just as it's about to be extruded.

In Fig. 2.5, a cross section at the gear pitch plane can be seen as well as the final solution.

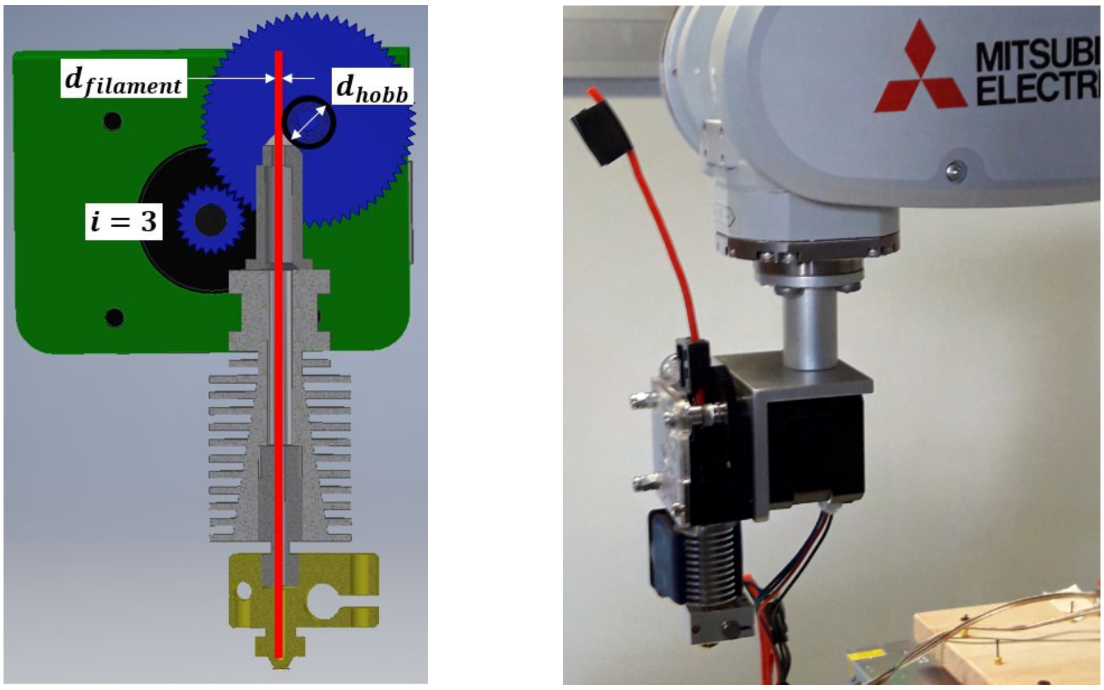


Figure 2.5: extruder feeding system cross section

Arduino Mega featuring Ethernet communication and RAMPS 1.6 It is a single board combining an ATmega2560 microprocessor (CPU), a 10/100 Ethernet controller chip (W5100) and an ATmega16u2 USB-serial converter. The board can be powered up to 28V, thus integrable with robot controllers or PLC racks running at 12V (or either 24V). Of the 54 digital pins and 16 analog inputs, only 4 pins are used by the W5100 (i.e. D10, D50, D51, D52), the others can be used for the application.
RAMPS shield (already introduced in 1.4.1) features all the electric circuitry to read the temperature sensors and actuate the motor (stepper driver) and the heater cartridge (MOSFET), thus has been chosen to obtain a compact solution.

Temperature Sensor The temperature sensor has also to be selected according to the maximum temperature. Such systems usually mount a Negative Temperature Coefficient (NTC) thermistor (for example, having $100K\Omega$ @ 25°C) that has to be coupled with its relative sensing circuit. As the temperature leaps over 300°C, a thermometer (or thermocouple) is advised. Therefore, a PT1000 (Platinum Resistance Thermometer) is installed on the extruder in combination with an amplifier that outputs an analog signal between 0-5V. In Fig 2.6, the volt-temperature curve can be seen for both sensors.

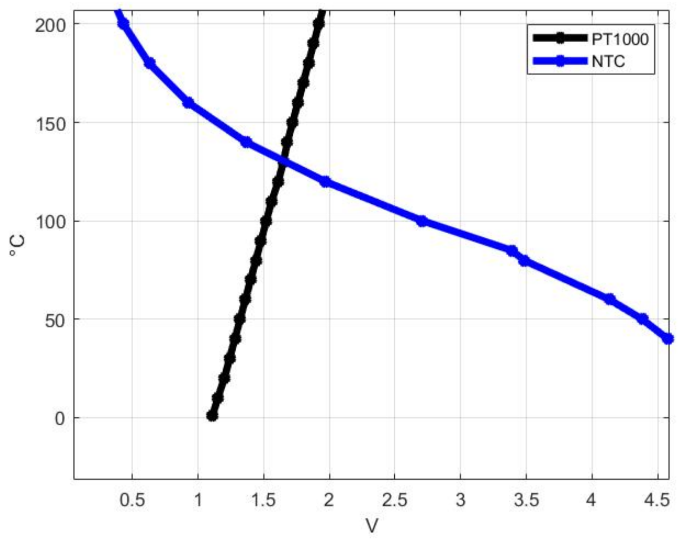


Figure 2.6: Volt-Temperature curve of a PT1000 vs NTC sensor

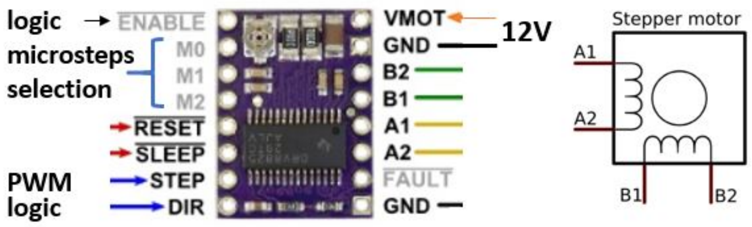


Figure 2.7: Stepper driver installation schematic

Stepper Driver A DRV8825 driver (by Texas Instrument) has been chosen for its operating voltage and current (2.5A maximum against a 1.68A required by the motor) and the possibility to reach up to 1/32 microstepping. In Fig 2.7, the component with its pin-out is reported, highlighting the main input signal types (i.e. 5V logic signal for the enable and direction, CW and CCW, and PWM for the stepping frequency).
The DRV8825's current limit needs to be set at or below the current rating of the stepper motor. To set it, the voltage on the "ref" pin needs to be measured in order to calculate the resulting current limit. The current limit relates to the reference voltage as follows: $CurrentLimit = V_{REF} \times 2$.
The selected stepper motor requires 1.68A, thus if the V_{REF} is limited to 0.75V the driver will be able to output 1.5A maximum.

MOSFET A Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) has an insulated gate that according to the voltage applied determines the conductivity of the device. The MOSFETs are used as electronic switching elements, even at high frequency, especially because they need low signal current to control the load current. They are used to regulate the current flow to the heater cartridge (12V and 40W, thus absorbing 3.3A) with a 20mA @5V signal from a board like the Arduino Mega. They can also be used to convert a 3.3V logic to a 5V one to interface board like the Arduino Due or Teensy 4.x to this kind of devices.

2.1.4 Software

Once outlined all the electrical components required, a mean to control them is mandatory. The routine of the code can be sum up in three categories: communication, temperature and motor speed control.
The firmware coded, hence, reads the output of the temperature sensor to close a PID loop that defines the duty cycle of the PWM signal going to the MOSFET gate, in concomitance with the generation of a second PWM signal used by the stepper driver to energize the coils at the desired frequency. The motherboard, meanwhile, awaits instructions over TCP/IP with a dedicated syntax to identify a given command (it is possible to adjust the temperature reference value, the stepper speed and request the current temperature).

TCP/IP client-server The extruder can be set to act as server or client in the LAN. For both cases, the first step is to set the static IP (Internet Protocol) address, the DNS (Domain Name System), gateway and subnet mask. As a server, the port, where it's running, has to be indicated when bringing it ip. As a client, during the setup, the board tries to connect to the server indicated using its IP and port.
While the connection is sustained, the firmware awaits incoming messages to operate the extruder. The possible strings are listed in Table 2.2.

Table 2.2: extruder command strings

Function	String	Comment
Step speed	G—n	the value send is half of the PWM period [micros]
Temperature set	M—n	
Temperature read	T0n	
Fan speed	F—n	PWM duty cycle in the range 0-255
SW Reset	R0n	

The incoming data package (with a maximum length is 16 bytes) is read only when available to avoid blocking lines of code. The string is parsed: the "n" is used as an internal terminator (character used for its analogy with the line feed "\n"), the first char is used to select a subroutines (for example, "G" sets the stepper speed and "M" the temperature. Both are inspired from the G-code language) while the bytes in between contain the reference values (in case of "T" and "R" the value is just a dummy and could be removed). Only the "G" starting line can contain negative values

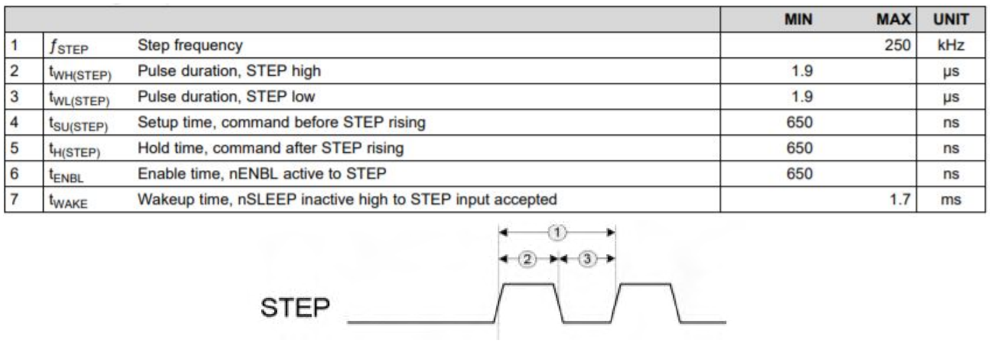
associated with the retraction of the filament.
An equivalent solution is to define a c++ data structure with two entries, a char and integer (the exact data format, and therefore number of bytes, depends on the microprocessor and CPUs used). This constraint the number of bytes to read from the buffer every time. When opting for this solution it's mandatory to make sure to align the machines by indicating the byte order (little or big-endian), the size of the data types (an *int* on the Arduino Uno, 8 bit CPU, has 2 bytes that is equal to a short on 64 bit computer) and the data alignment rules (padding is the mechanism used by the compiler to add empty bytes to have always a fixed block of bytes to read; different architectures might have different rules). This solution is advised when multiple instructions have to be done simultaneously (relative to the updating frequency) or data retrieved from the board.

Feed speed - steps per second ratio Before addressing the routine to control the speed of a stepper motor, how to relate the linear speed to the one of the motors has to be investigated.
Given $\dot{x}$, the feed speed of the filament in [mm/s], and *f*, the stepping change rate in [steps/s], then:

$$\frac{\dot{x}}{f} = J = \frac{\pi d_{hobb}}{\tau i} \quad \frac{[mm]}{[steps]}$$

J is the mm per steps ratio that allows to linearly shift between the operational and joint space. τ is the product of the number of steps per revolution and the number of microsteps per single step. i is the gear ratio. d_{hobb} is the operating pitch diameter of the last gear in the gearbox that actuate the filament (see Fig. 2.5).

Stepper speed control The stepper driver requires a first logical signal to be enabled (i.e. energize the coils of the stepper motor). This can be done at startup or before the first moving command by setting the signal to "Low". The other two main signals coming from the controlling board are the direction logical value ("Low" for CW and "High" for CCW) and the PWM signal for the stepping frequency (see Fig.2.8 for the characteristics of the signal). The generation of a step is obtained upon the PWM signal



			MIN	MAX	UNIT
1	f_{STEP}	Step frequency		250	kHz
2	$t_{WH(STEP)}$	Pulse duration, STEP high	1.9		μs
3	$t_{WL(STEP)}$	Pulse duration, STEP low	1.9		μs
4	$t_{SU(STEP)}$	Setup time, command before STEP rising	650		ns
5	$t_{H(STEP)}$	Hold time, command after STEP rising	650		ns
6	t_{ENBL}	Enable time, nENBL active to STEP	650		ns
7	t_{WAKE}	Wakeup time, nSLEEP inactive high to STEP input accepted		1.7	ms

Figure 2.8: Timing Requirements of the DRV8825

going from "High" to "Low". The Pulse-Width Modulation period correspond to the time to complete a single step; usually the duty cycle is fixed at 50%, but different drivers could accept other values.
To guarantee the generation of the correct signal from the motherboard while dealing with the other task, that could prevent the timing, it has been decided to use an interrupt every half period, T_{half_period}, to swap the logic level.

Arduino timers and interrupts Let's briefly review microprocessor timing and interrupting functions.
The Arduino Mega has a clock speed of 16MHz (i.e. 62.5ns per count) and 5 timers, three of which are 16-bit (can count up to $2^{16} = 65536$ before their restart).
Various registers and their definition make a timer work in a specific manner. The main ones are:

- OCRnA/B (Output Compare Register): its values are used to compare the counter value for a given actions.
- TCCRnA/B (Timer/Counter Control Registers): it allows to define a prescaler and operation mode. The prescaler is a subdivision (accepted values: 1, 8, 64, 256, 1024) of the primary clock frequency that leads to the actual counting frequency. For example, a prescaler of 8 slows down the 16MHz clock to 2MHz (a count every 62.5ns is changed to one every 500ns).

The Operation mode used is the CTC (Clear Timer on Compare match). It compares the value defined in the OCRnA/B register to the counter of the prescaled clock; if they are the same, the interrupt takes place and the counter is restarted.
In order to use one of the microprocessor timers, the prescaler needs to be selected.
To do so, the printing process is simulated. Knowing the diameter of the filament, J and the cross section of the strand on the part (3 [mm], 1674.18 [steps/mm] and 0.1514 [mm^2] respectively), it's possible to compute the variation of the OCRnA (value that has to be greater than 1) by letting vary the robot speed and the microstepping value:

$$OCRnA = round\left(\frac{16 \times T_{half\_period}}{prescaler} - 1\right)$$

When choosing the prescaler, microstepping should be selected from those values that guarantee a smooth operation and fine positioning resolution, therefore 1/8, 1/16 or 1/32 (for the motor selected: full step is 0.9[deg], half-step is 0.45[deg] and so forth). As the number of microsteps per step and/or speed increases, the OCRnA decreases, as can be seen also in Fig.2.9 and 2.10). From these curves is possible, moreover, to obtain the sensitivity of the extruder that is how much the printing speed needs to change to induce a variation of the motor speed. It can be seen how the OCRnA curves degenerate in a stair-step profile (prescaler of 64 and 256) as the robot speed increases. Therefore, for printing speed approaching and crossing 200 mm/s (i.e. a higher frequency interrupt), a prescaler equal to 8 guarantees a viable stepping resolution.

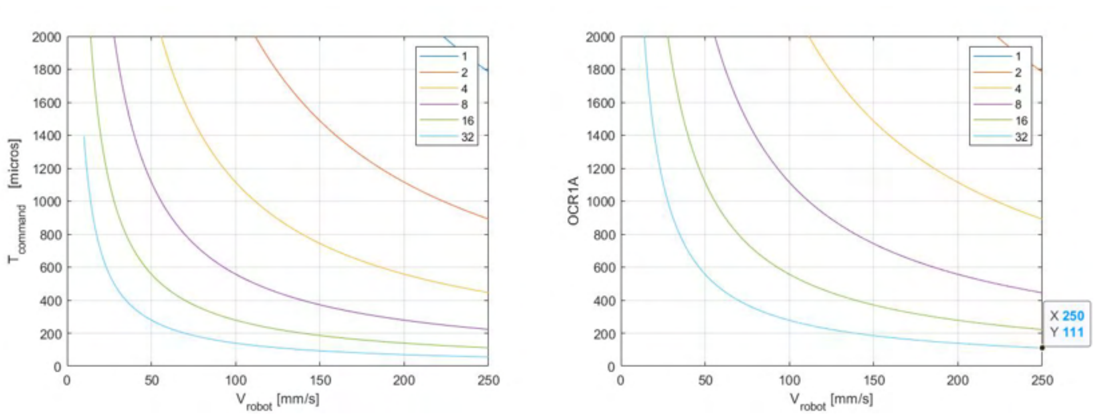


Figure 2.9: T_{half_period} and OCRnA as a function of the microsteps selection and robot feed for prescaler equal to 8

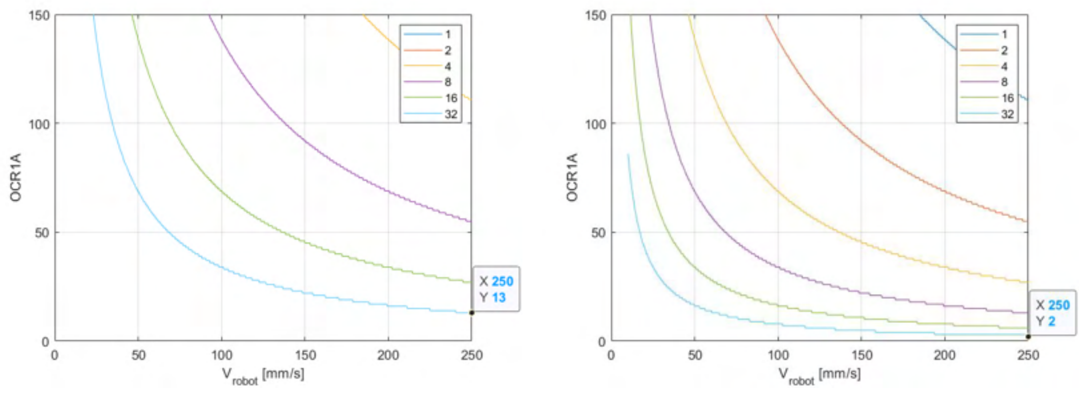


Figure 2.10: OCRnA as a function of the microsteps selection and robot feed for prescaler equal to 64 (left) and 256 (right)

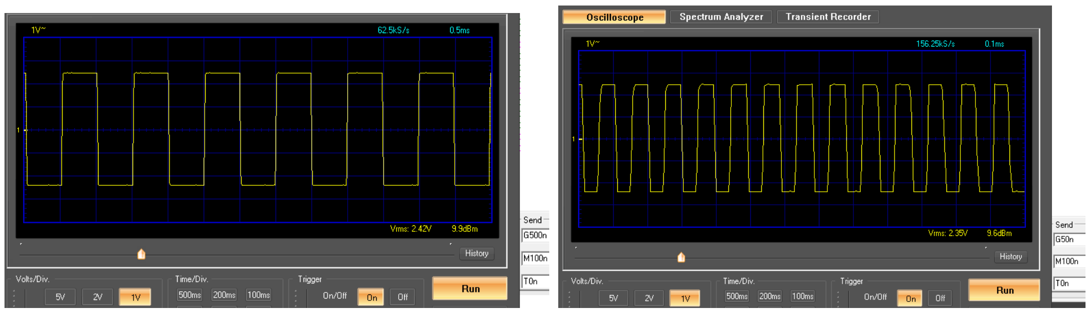


Figure 2.11: PWM signal generation measurement for validation for half periods equal to 500 and 50 [μs]

The resulting implemented code has been tested by connecting an oscilloscope[2] to the Arduino pin designated for the PWM signal. The reference speed is set using a

[2]Velleman Mini Pc Scope PCSU01

TCP/IP client terminal (*Hercules setup utility*[3]). Results can be seen in Fig2.11 for “G500n” and “G50n”.

Speed control test To validate feed-forward control of the stepper speed, an incremental quadrature encoder (8192 counts per round) is coupled to the motor shaft by means of an elastic coupler (Fig.2.12).

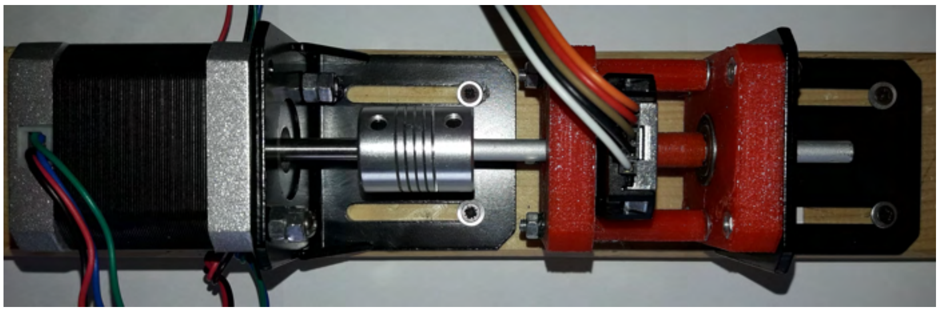


Figure 2.12: speed profile test rig

The measuring unit is created using a conventional 3D printer for the bearing housings and spacers and a lathe-like 3D printer (it is discussed in 5.3) for the shaft shoulder and sensor interface.
A first sets of tests are carried out at constant speed before attempting a trapezoidal speed profile curve. In the latter case, it is adopted a trajectory that has to complete a single turn (360 [deg]) in 5 [s] with A/D of 10% of the total time. The controller (and consequently the sampling frequency of the motion law) is run at 100Hz. The motor speed is converted in the time interval to interrupt the Arduino board using:

$$T = \frac{1}{2} \times \frac{10^6}{v \times J_d} \quad [\mu s]$$

Where J_d is variation of the microstep per degree.

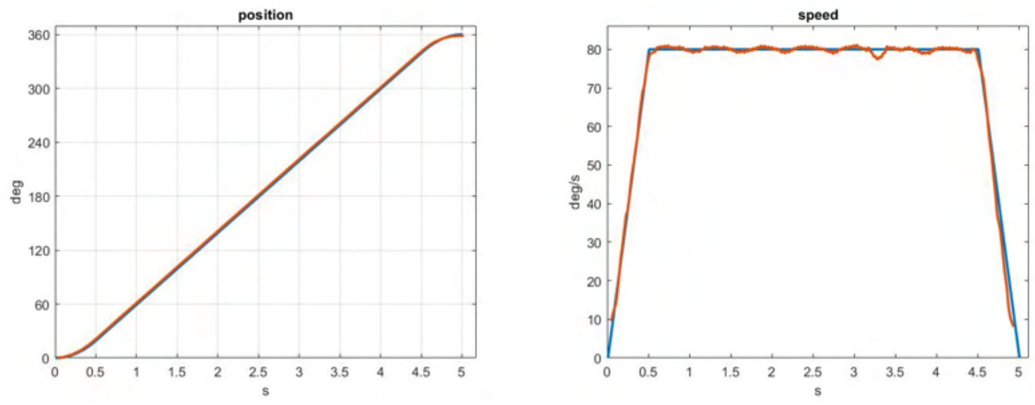


Figure 2.13: position and speed result for $\theta = 360\,\text{deg}$, $T_{total} = 5s$, $A/D = 0.5s$

[3]*https://www.hw-group.com/software/hercules-setup-utility*

The results, both in position and speed (as numerical derivative), are depicted in Fig.2.13; the sample of the angular position is filtered by a low-pass filter at 150Hz while the speed by a Savitzky-Golay filter (using Matlab's *smoothdata* function and a window of 32 samples).
It has also to be validated the responsiveness of the firmware to an incoming message. The controller has been tested on a trajectory similar to the previous one, but with A equal to 15% and D to 30%; the motion law is then mirrored as can be seen in Fig.2.14 to bring back the motor to its initial position. The controller frequencies tested are 250, 142.86, 100, 66.67, 50, 20Hz. The firmware is able to react to inputs send every 4ms.

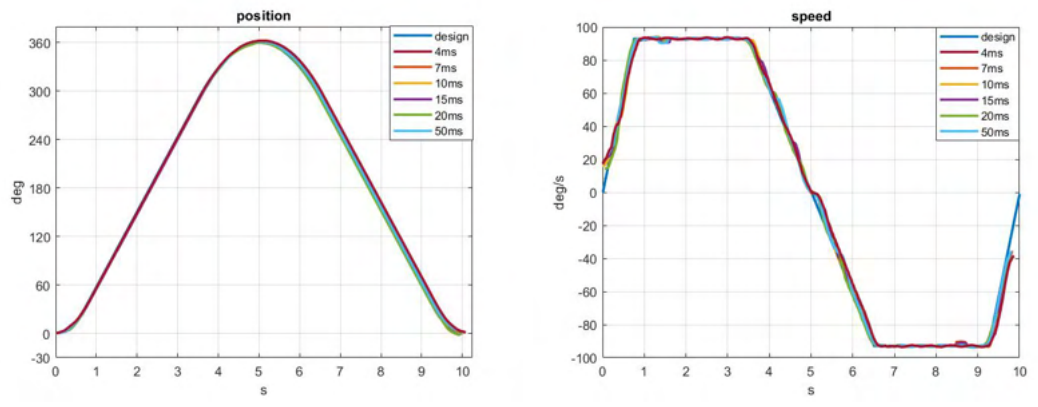


Figure 2.14: position and speed result for controller frequency 20Hz-250Hz

Temperature control The temperature at the nozzle is done by means of a heater cartridge connected to a MOSFET that regulates the current flow to a heat cartridge. The temperature is read by a sensor and used to compute the control action that is the duty cycle of a PWM signal running at 4 Hz going to the gate of the electrical component. In other words, the controller determines for how long the heater has to be on for a period of time equal to 250[ms].
The PID implemented is taken from literature [76] and is depicted in Fig2.15; the derivative action depends only on the temperature measured; the integral term saturates as anti-wind-up measure and its contribution is accounted only within a range of the set temperature (10°C). The overall control action is bounded between 0 and 255 (maximum value writable on an analog output of the Arduino board).

The tuning of the PID parameters is done experimentally. The values identified are 20, 0.5 and 5 for the proportional, integral and derivative gain respectively. The behavior of the system can be seen in Fig2.16 as a function of time for "M150n", "M180n" and "M200n". The curves are obtained using a Python executable that asks at a fixed frequency the current temperature using the "T0n" until the printing process could be started. The time-temperature points are then saved for subsequent processing.
The integral contribution starts 10°C before the reference value, as the PID designed, and settles after around 70s to the reference value.

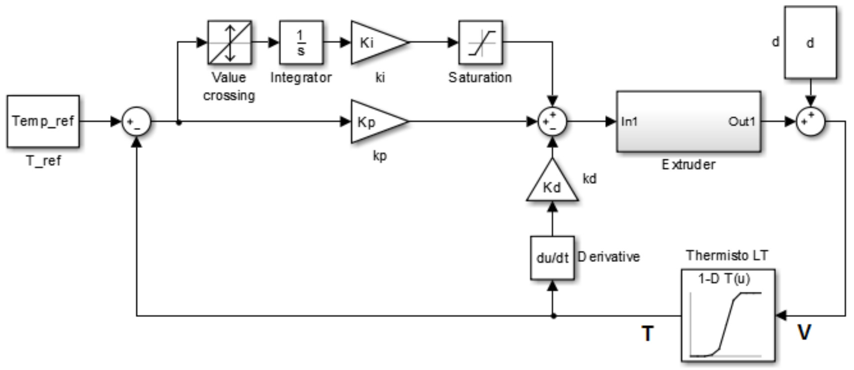


Figure 2.15: PID scheme of the temperature control

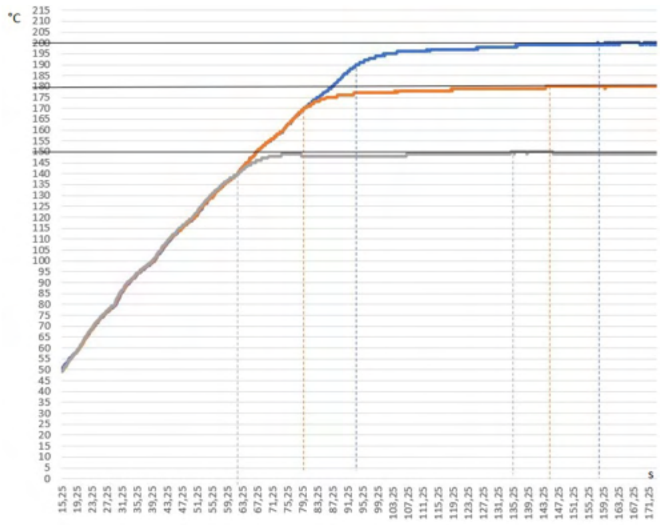


Figure 2.16: experimental time-temperature curve for 150°, 180°ans 200°

2.1.5 Robot flange

Having completed the firmware and its debugging phase, how to mount it on the wrist of a robot has to be addressed (in case of *mobile tool* cell). The design of a suitable flange can be stated as the problem to find the position of the center of gravity (COG) of the tool on the robot payload curve (Fig.2.17) that guarantees an even flow of the filament.
The dimensions of the tool are taken from the respective CADs; the nozzle (our TCP) can be expressed as a vector, whose coordinates are [57, -10.5, 97], from a reference frame positioned in the rear-middle-top point of the stepper. The weight of the extruder is around 0.55 Kg (motor: 0.35 Kg; feeder with hotend: 0.2Kg) that is increased for safety reason to 0.7Kg ($\eta = 1.25$). From this information, some preliminary conclusions can be drawn: the displacement of the filament inlet is already enough to assure a continuous provision; placing the TCP far from the wrist will reduce the printing volume without any significant advantage. The flange, therefore, has to create a gap along the

z-axis to allow the tightening of the screws for rapid tooling. The robot-tool interface (that can be seen in Fig.3.24) is compliant with the ISO 9409-1.

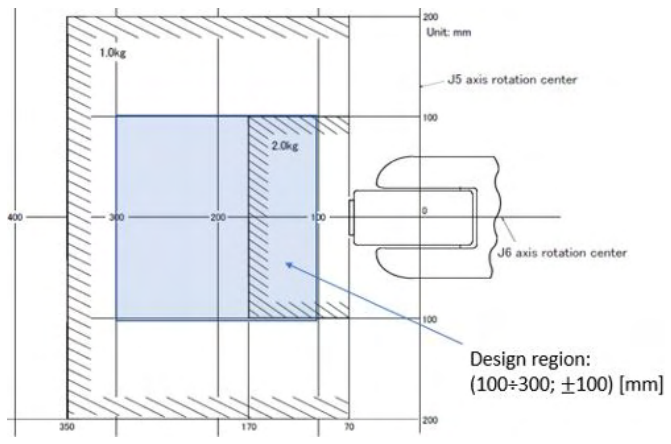


Figure 2.17: Position of center of gravity of loads for the RV-2F-Q

2.2 Additional Extrusion system

The extruder detailed until now, is one of the possible solutions that can be found in literature and in the industrial field. Different devices could provide features that enhance the performances of the printer; therefore they have been investigated.
There are three possible solutions in which the flow can be generated: filament based, syringe and screw. Generally speaking, the resulting process can be continuous, that is producing a not discontinuous flow of material, or semi-continuous (or discontinuous). Filament extruders, both types, are capable to process one cartridge (spool of 1Kg or even less) at a time, making it a discontinuous process for large prints. Bigger spools are available (10-25 Kg) on the market that could be exploited in these cases. Such cartridges cannot be stored on the robot itself (as could have been done for the mobile extruder case with a standard reel), therefore only a fixed extruder could process them. The other two types of extrusion system are discussed in the following section highlighting working principles and relative advantages and disadvantages.

2.2.1 Syringe / Plunger Extrusion

In the syringe extruder, the material (liquids or viscous paste) is inserted inside a chamber and its flow is controlled by a plunger. These extruders can cope with material that can be heated or not.
Biomaterials can be used and it is thanks to them that the range of possibilities increases; in this case through the syringe will no longer come out a thread, but drops of polymers [77]. The great potential of this devices is the extreme precision on which you can rely; one problem with this method, however, is that the process is discontinuous.

2.2.2 Screw Extrusion

Screw extruders use a rotating element around which material is extruded constantly and continuously; usually this part is a screw, in other cases it is a disc.
The screw designs adopted for a continuous extrusion can be of two types: single [78] or dual screw design [79].

Dual screw extruder: A dual screw extruder has a complex structure, but allows to operate at very high precision, it is mostly used by advanced industries. The fundamental ability of this type of extruder is that it allows pellets to be mixed with remarkable quality, thanks to a detailed design of the parts. The price is higher than single screw for the complexity of the machine and for the control requirements that the process requires compared to a single screw, which has an easy control process. There are two types of twin-screw extruders on the market: co-rotating and counter-rotating. The difference is in the relative direction that the screws have with respect to each other. In counter-rotating extruders the pressure is higher, but so is the degree of mixing.
The screw geometry is modular, as several elements allow a quick reconfiguration for a change in the process or for the replacement of specific components due, for example, to wear.
The main features and advantages (with respect to the single screw) of this type of extruder are:

- Stable flow, quality and yield.
- Reliability in the production process.
- A large amount of heat of the extruder is generated by the mechanical transformation, while in the single-screw machine a preheating of the parts is often required.
- Lower energy consumption (it is about 30% lower than the single screw).
- Simple material feeding.
- Excellent discharge capacity. Higher productivity and plasticizing capacity.
- Provides a high level of process flexibility and the ability to control multiple processes such as mixing, cooling, etc.

Single screw extruder: It is the most widely used, especially for costs and parameters to be managed. The screw allows for a greater ratio between motor speed and polymer feed causing less irregularities in the output flow. The single-screw type has only one screw placed inside the device and is mostly used with pellets; it's simple and effective. The screw is responsible for the transport of the raw material, its fusion and dosage.
The disadvantages that can occur are:

- The feeding performance is reduced, since the material is transported mostly through friction.
- It has a minimal effect on the surface of the exhaust gases.
- It is not suitable for special processes such as the coloring of polymers or the manufacture of products through thermosetting powders.

The screw is divided into three sections: feeding, melting and dosing section [80][81][82]. They can be seen in Fig.2.18.

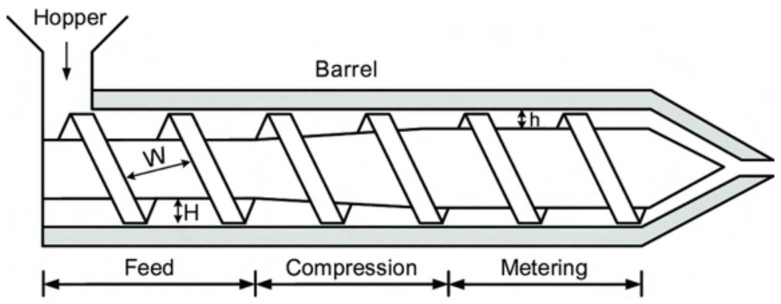


Figure 2.18: Screw sections: feeding, melting and dosing [83].

- Feeding Zone or Solids Transport Zone: in this part the material, in the form of pellets, arrives from the hopper, often after having been dried and mixed. The depth of the channel is usually the same in the whole area.
- Melting Zone or Transition or Compression Zone: this is the part where the material is brought forward and where most of the polymer is melted. Independent PIDs can be placed to heat the barrel in a controlled manner, dividing it into various zones, so as to progressively increase the temperature towards the front of the extruder, where there's the output to reduce the possibility of overheating. It is also true, however, that if the process is carried out quickly enough, the heaters can be turned off and the temperature can be maintained by friction and pressure.
  The depth of the channel becomes smaller and smaller. Very important is the difference in the depth of the barrel that allows to increase the pressure towards the dosing area, where they must arrive already melted so as not to create problems during the process.
- Dosing or Transport Zone of the melt: here the depth is constant, as in the first part; the melted particles form a uniform composition. It is in this part that the material is pressurized and transported toward the nozzle. Filtering devices and static mixers can also be found here. An irregular flow at this stage can cause a product with undesirable imperfections.

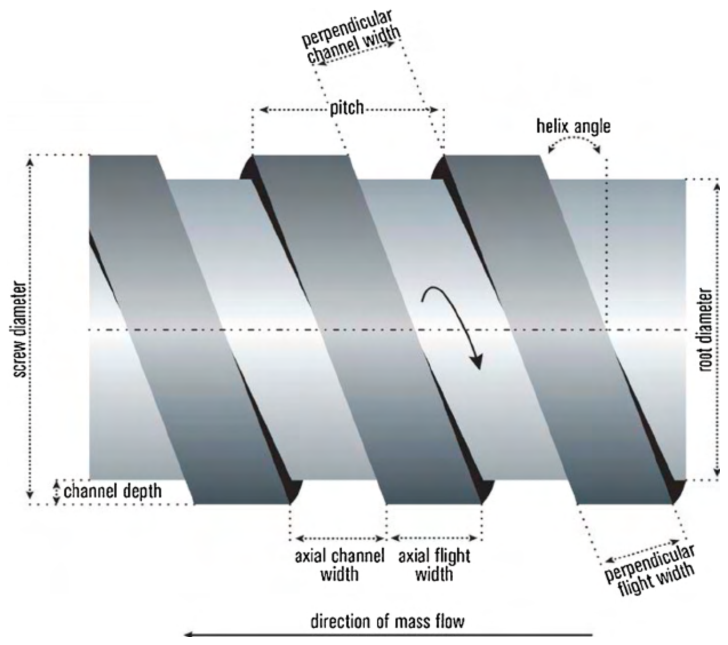


Figure 2.19: Screw sections: feeding, melting and dosing [84].

2.2.3 Screw Parameters

The most important parameters (depicted in Fig.2.19), characterizing a screw used to extrude pellets, are [71]:

- Length: the most commonly used length / diameter ratio is 24:1; in general, it can go from 10:1 to 50:1. The length and melting time are directly proportional and are also related to power.

- Pitch: horizontal distance of the points corresponding to two successive lands.

- Flight thickness: is about 0.1 x diameter. If it were thicker it would mean more heat and less material transport per revolution, while thinner would cause less pumping, but more mixing.

- Helix angle: angle between the thread and the transverse plane of the screw; if it is too small it increases degradation and wastes too much channel volume. The formulas to be followed change according to the material, the flow rate of the channels and the type of process used.

- Depth of the channel: perpendicular distance from the highest point of the thread to the base of the surface. In the dosing areas, more superficiality means excellent mixing but less power, instead if deeper you will have the opposite and a higher pressure and sensitivity.

2.3 Pellet Extrusion System

Directly working with feedstock (pellet form) will reduce production cost because the plant will be directly working with raw material and a continuous flow of material provided that a suitable pumping system is used to connect the extruder with the reservoir.
An equivalent in-house solution to the one for a filament reel is being developed based on a single screw. The final design will enhance the manufacturing cell with virtually endless prints and the possibility to recycle material (if possible).
An example of discontinuous feedstock extruder, used during this research activity, is Ephaestus, a linear delta robot combined with a controlled injection molding unit [85]. In this case, the system is provided with a loading cylinder through which the pellet (from a hopper) is preheated, plasticized and discharged in the feeding chamber. Here, thermocouples are used to guarantee the desired temperature of extrusion in the chamber itself and at the nozzle. The material feed speed is obtained by controlling the forwarding of a piston (hence the source of discontinuity, same goes for the plasticize unit that is actuated by the same principle). Control and monitoring of the material flow rate are crucial to yield good layers and therefore good parts. In Fig.2.20, the machine can be seen as well as for a printed sample.

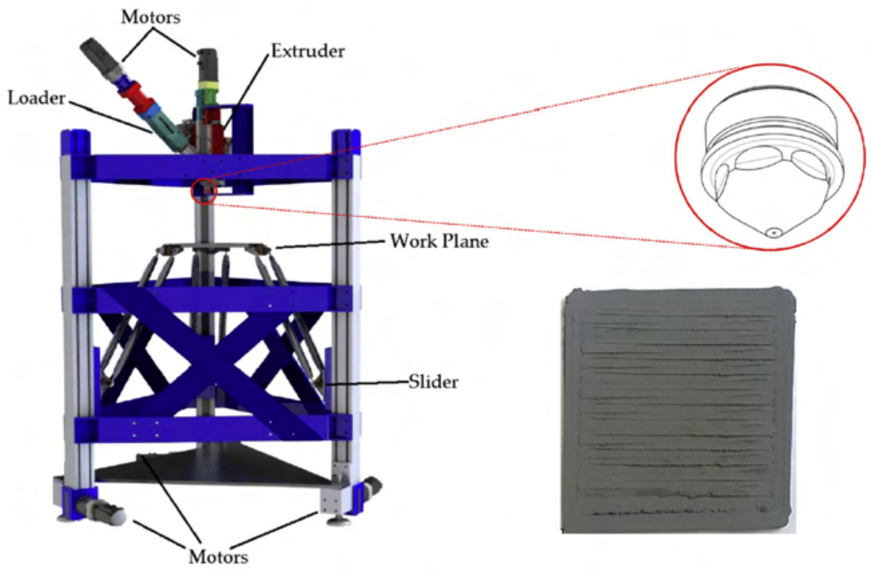


Figure 2.20: Example of 3D printer with a discontinuous extrusion unit: Ephaestus

Chapter 3

Robotic system

A tool able to extrude material as instructed has to be supported by a manipulator able to generate the printing curve. It could either be manufactured or purchased; it is opted for a system ready to use because its development from scratch lies outside this research activity and would have not introduced no significant advantage at this stage. The selection of a suitable robot and control system assures that the printing process is carried out as planned. Therefore, it's required to investigate the robotic system in its kinematics, joint limits and I/O capabilities to prevent constraints to the final solution. In this chapter, five robots, that have been turned into AM machines, are briefly presented outlining they characteristics (section 3.1). In-depth analysis of robotics can be found in [47], [48] and [86].
The matter is then shifted to the identified control architectures (namely: *Data Link*, *Real Time*, *Stand-Alone*) that can be used to master all the devices in the cell. In section 3.2, it is also illustrated how they can be implemented using the previously discussed robot as examples.
Testing programs on the physical machine can be dangerous due to the presence of bugs in the programs. Offline simulations are therefore a viable and sensible tool to check the performance of the implemented control without damaging the machine or causing harm (section 3.3). Basic curves are used to validate the trajectory followed by the TCP in the case of the three architectures.

3.1 Industrial robot considered

Five industrial robotic system have been considered, namely an Epson SCARA (*T3*), anthropomorphic robots by Mitsubishi (*RV-2F-Q*) and ABB (*IRB 2400*) and cobots manufactured by Universal Robots (*UR3*) and Techman Robot (TM5-700).
Before addressing how the control can be set to be transformed into a 3D printer, an overview of these specific machines is due. They all belong to the serial kinematics family with 6 Dofs, except for the SCARA robot (with 4). Only a part of the functions will be addressed in this chapter, the manuals will be referenced for those interested in a thorough analysis and dissertation.

3.1.1 Epson SCARA T3 [87][88][89]

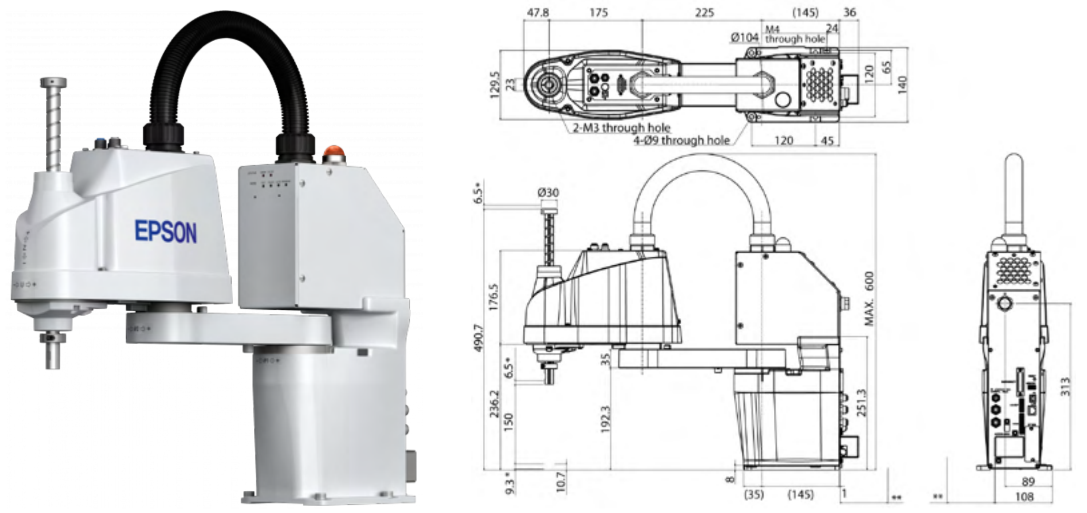


Figure 3.1: Epson SCARA T3

This is a SCARA with 4 actuated axes (by AC servo motors). The kinematic chain is a standard RRPR. The reach is of 400 mm (link1 is 225, while link2 is 175 mm); the maximum vertical displacement is of 150 mm and the J4 can rotate $\pm 360 deg$. Repeatability is in the order of 0.02 mm-deg. The maximum payload is 3 kg (rated 1Kg).
The maximum operating speed is, for a PTP (Point To Point) movement, 3.7 m/s on J1 and J2, 1 m/s on J3 and 2.6 kdeg/s on J4. Speed and acceleration for unconstrained path movements can be chosen between 1 and 100% (120% for the acceleration). For liner movements the maximum speed and acceleration are 2 m/s and 10 m/s^2.
The controller offers PTP and CP (Continuous Path) position control programmable by means of *SPEL+* (multi-tasking robot language). It also features vision system, force control and conveyor tracking.
It can communicate with the outer world using I/O (18/12 pins), teach pendant, USB-B (programming), USB-A (memory) and Ethernet (8 ports - 10/100 Mbps) ports.

3.1.2 Mitsubishi Anthropomorphic RV-2F-Q [90][91][92]

This robot arm has 6 Dofs (all revolute joint, AC servo motor with absolute encoders) with J4-J5-J6 forming a spherical wrist. The reachable radius is 504 mm (upper arm is 230mm long, while the forearm is 270mm). J6 can spin $\pm 360 deg$ while the base (J1) between $\pm 240 deg$. Repeatability is 0.02 mm. The rated payload is 2 Kg; the maximum one is 3 Kg.
The maximum resultant speed is 4.95 m/s; the maximum speed of motion for each axis is respectively 300, 150, 300, 450, 450 and 720 deg/s. The minimum acceleration/deceleration time is 0.2 s. It can be manually set or let the controller estimate the A/D percentages to comply with the tool and payload.

It's programmable using *MELFA-BASIC V* language that offers standard position control, vision system, force control, conveyor tracking and monitoring. It has 8/8 I/O (on the EE), teaching pendant, USB-Bmini and Ethernet (10/100 Mbps; 9 ports + 3 dedicated to external Real Time, RT). The controller uses multi-threads and accept instruction (with RT) every 7.11 ms (although some controllers runs at higher frequency).

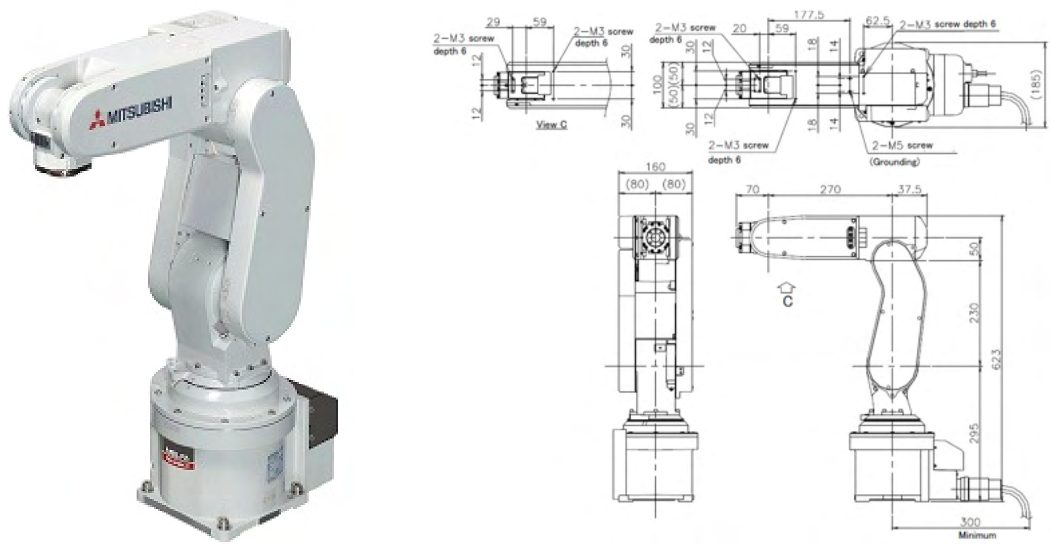


Figure 3.2: Mitsubishi Anthropomorphic RV-2F-Q

3.1.3 ABB Anthropomorphic IRB 2400 [93][94]

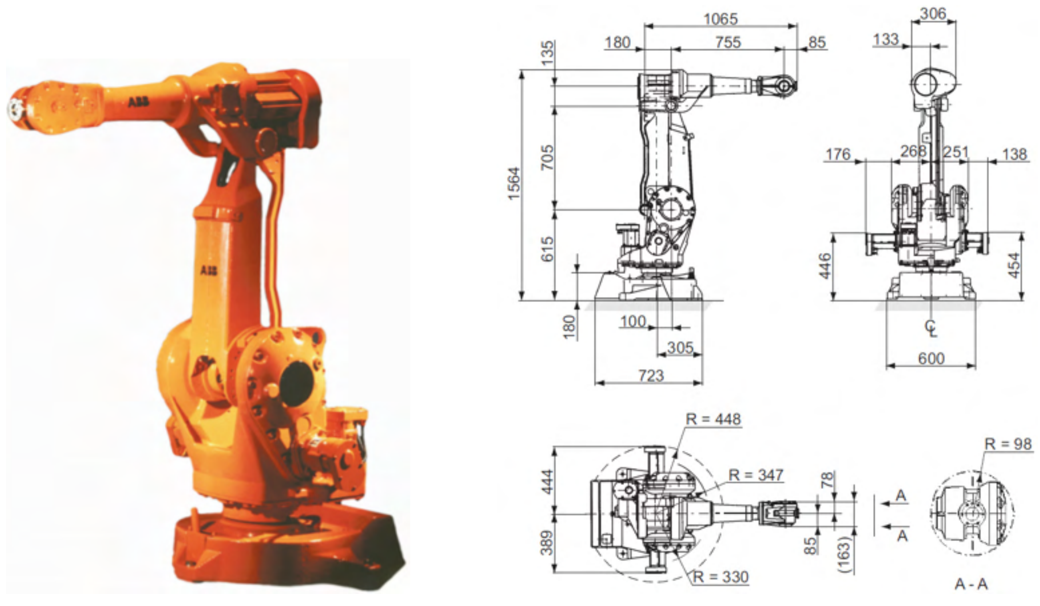


Figure 3.3: ABB Anthropomorphic IRB 2400

This robot has 6 Dofs (all revolute joint, AC servo motor with resolvers), three of which form a spherical wrist. J3 motor is placed near J2 and it is connected to its axis by means of a 4-bar linkage, making the IRB 2400, technically, a hybrid robot (serial plus parallel kinematics). The reachable radius is 1.55 m. J6 can spin $\pm 400 deg$ while the base (J1) between $\pm 180 deg$. Pose repeatability is 0.03 mm, while the path one is 0.15mm. The rated payload is 16 Kg.
The maximum TCP linear speed is 7 m/s while the reorient speed is 500 deg/s; the maximum speed of motion for each axis is respectively 150, 150, 150, 360, 360 and 450 deg/s.
RAPID language offers the possibility to master the robot either from a PC (via Ethernet) or directly from the teaching pendant (FlexPendant). The possibility to integrate the controller with PLC modules allows to expand it with I/O, Serial and Ethernet ports. USB-A connectors can be either used to upload-download files or to save (or restore) the controller creating a backup.

3.1.4 Universal Robots UR3 [95]

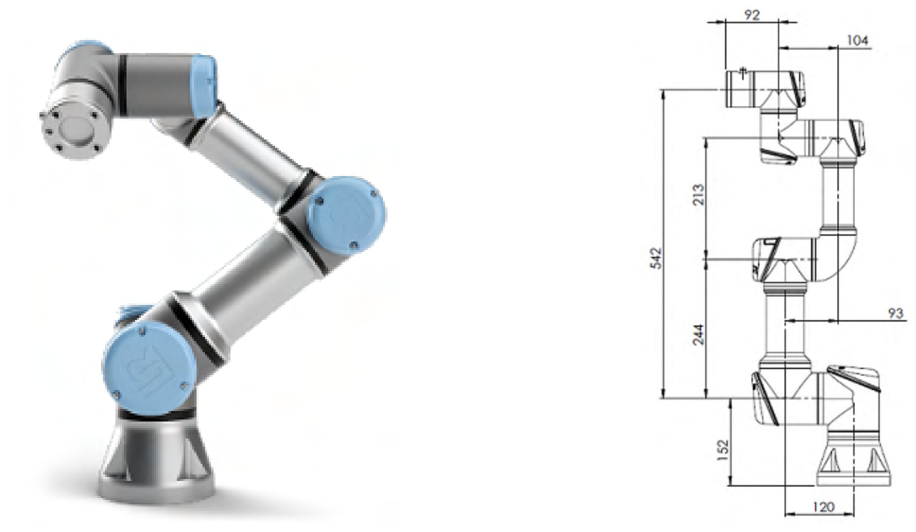


Figure 3.4: Universal Robots UR3

This is a 6 Dofs collaborative robot that differentiate, just by addressing the kinematic, from the two previous ones for the absence of a spherical wrist (the J4-J5-J6 axis do not intersect in a single point). The reach is 500mm, the payload is 3Kg and the repeatability is $\pm 0.03mm$. All the joints can span between ± 360 deg while J6 can actually spin infinitely. On the EE a 6-axis force/torque sensor (3.5N accuracy) is used to monitor the force exchanged with the outer world. The maximum force measured is 30N (resolution 1N), while the torque is 10Nm (resolution 0.02Nm).
As concerns the speed, the first three joints can reach up to ± 180 deg/s, while the others ± 360 deg/s. The TCP speed is 1 m/s (250mm/s if in manual movement according to the standards).

The robot is programmable directly from the teaching pendant (offline simulator is available for debug).The user can either generate the program by using the GUI interface (or PolyScope) or by script using the *URScript* language. The controller features 18/18 I/O (2 per each are analog), RS 485 for tool communication and a 1000BASE-T Ethernet port.

3.1.5 Techman Robot TM5-700 [96]

This is a 6 Dofs collaborative robot that has the same kinematic chain of the one manufactured by Universal Robots. The reach is 700mm and the payload 6Kg. J1 and J6 can rotate $\pm 270\,\text{deg}$. The repeatability is $\pm 0.05mm$.
The first three joints can reach up to ±180 deg/s, while the others ±225 deg/s.
The robot is programmed connecting a monitor directly to the controller or using a PC to access *TMflow*. This graphical HMI allows users to manage and set the parameters of the robot and to implement graphical flow charts to plan the robot movement and process logic. The controller features 16/16 I/O, Ethernet port, HMI-VGA and 6 USB ports. The arm is characterized by the presence of a camera on the EE.

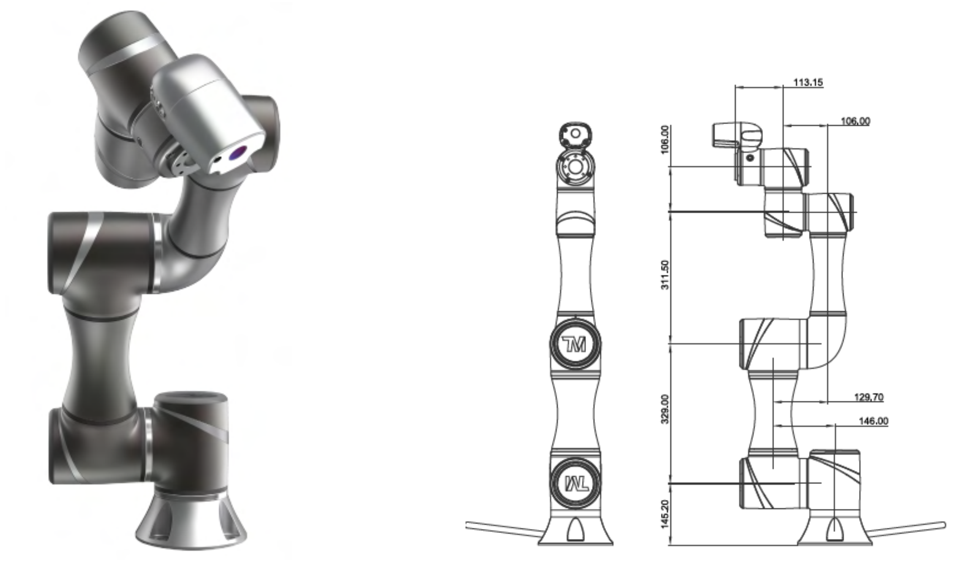


Figure 3.5: Techman Robot TM5-700

3.2 Control architectures

Industrial robots, as has been outlined in the previous section, offer several means to communicate with the external world: I/O modules, serial and Ethernet ports.
It has been decided that any subsystem should talk via Ethernet (reflected by the extrusion system in section 2.1) to simplify future addition of devices in the network. Thus, a LAN (Local Access Network) is set with the robot as server and the extruder as client (with the possibility to be converted into the mirroring case).

Having identified the medium over which the communication will occur, the transport protocol has to be selected. It's opted for TCP (Transmission Control Protocol, TCP/IP) and UDP (User Data Protocol) since it's available in all machines and programming languages (computer side) and avoid using hard real-time kernels (corroborated by the fact that at this stage of the research, the in-house control of the joint axes is not interested, to allow a fast conversion of any robot into a 3D printer).
TCP leads to a first mean to instruct the machine that is *Data Link* (*DL*, Fig.3.6); it sees the robotic system loosely resembling any CNC machine, to which a series of target positions is feed leaving the motion law assignment to the built-in functions.
UDP generate a second mean, named *Real Time* (*RT*, Fig.3.7). It relies on soft real-time protocols to master the actual motion profile; therefore, it should make sure that the synchronization of the extruder to the TCP speed is obtained as designed.

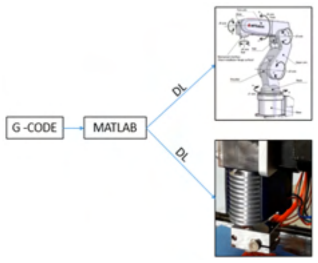


Figure 3.6: 1st scenario

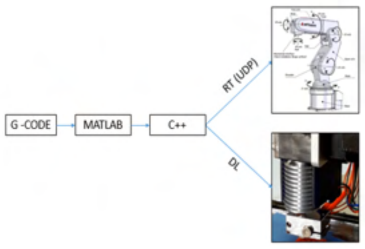


Figure 3.7: 2nd scenario

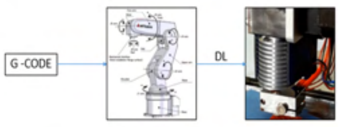


Figure 3.8: 3rd scenario

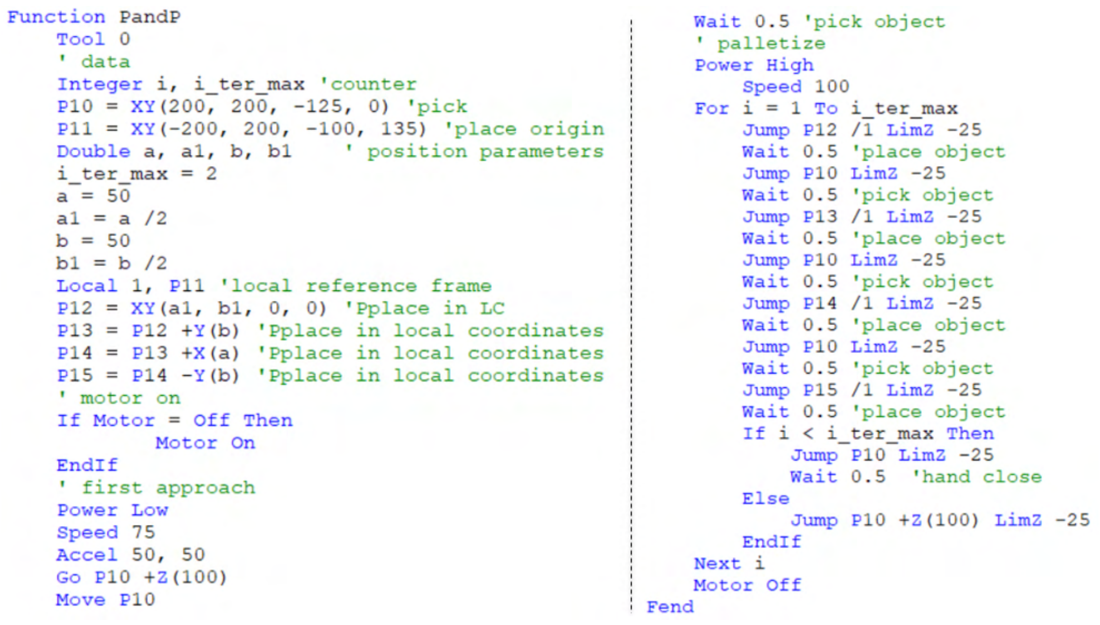
Both solutions require an extra device able to guarantee the time interval between two instructions (down to milliseconds) and to process the trajectory. Therefore, a PC is set up to belong to the same LAN. On the computer side all the planning is performed ahead (using *MATLAB* or *Python*) and, once checked and post-processed, the data is sent to the robot controller where the control loops are closed.
The *DL* case might be adjusted, if the controller is suited to such end, to read a G-code file from the hard disk and letting the robot masters the other actors in the network. This third solution (Fig.3.8), avoids the additional device in the LAN, therefore it has been indicated with the name *Stand-alone*.

3.2.1 *Stand-alone*

With this architecture, the robot controller either reads a G-code file uploaded by an interpreter function or receive a previously parsed file converted in machine instruction (less time consuming for the robot). In both cases, a single *G* command (for example, "G1 X10 Y10 Z10 E10 F600") is split in at least two commands, depending on the robot programming language, one to control the extruder and one for its own movement.
In the following two subsections, the two cases are presented.

Machine instructions

This is probably the simplest solution to implement. It consists in using a piece of code after the trajectory planning (if accessible in the SW used) or after the G-code routine to generate the instructions (program file) for the machine. This function is highly customized around the specific brand of robot used. The only significant drawback is the maximum program file size that can be uploaded on the controller.
Let's consider a specific case, a *Epson* SCARA robot that has to work as a standard FDM 3D printer.
Firstly, the language used by the controller has to be investigated. Fig.3.9 reports

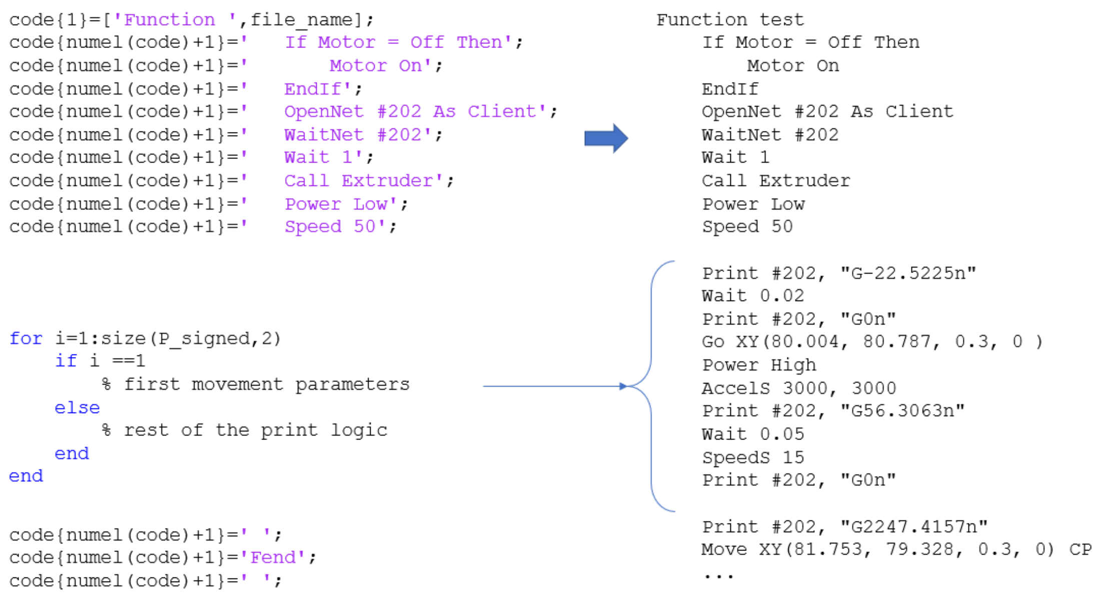


```
Function PandP
    Tool 0
    ' data
    Integer i, i_ter_max 'counter
    P10 = XY(200, 200, -125, 0) 'pick
    P11 = XY(-200, 200, -100, 135) 'place origin
    Double a, a1, b, b1     ' position parameters
    i_ter_max = 2
    a = 50
    a1 = a /2
    b = 50
    b1 = b /2
    Local 1, P11 'local reference frame
    P12 = XY(a1, b1, 0, 0) 'Pplace in LC
    P13 = P12 +Y(b) 'Pplace in local coordinates
    P14 = P13 +X(a) 'Pplace in local coordinates
    P15 = P14 -Y(b) 'Pplace in local coordinates
    ' motor on
    If Motor = Off Then
            Motor On
    EndIf
    ' first approach
    Power Low
    Speed 75
    Accel 50, 50
    Go P10 +Z(100)
    Move P10
    Wait 0.5 'pick object
    ' palletize
    Power High
        Speed 100
    For i = 1 To i_ter_max
        Jump P12 /1 LimZ -25
        Wait 0.5 'place object
        Jump P10 LimZ -25
        Wait 0.5 'pick object
        Jump P13 /1 LimZ -25
        Wait 0.5 'place object
        Jump P10 LimZ -25
        Wait 0.5 'pick object
        Jump P14 /1 LimZ -25
        Wait 0.5 'place object
        Jump P10 LimZ -25
        Wait 0.5 'pick object
        Jump P15 /1 LimZ -25
        Wait 0.5 'place object
        If i < i_ter_max Then
            Jump P10 LimZ -25
            Wait 0.5  'hand close
        Else
            Jump P10 +Z(100) LimZ -25
        EndIf
    Next i
    Motor Off
Fend
```

Figure 3.9: Example of *Epson* program for pick and place operation

a basic routine for a pick and place task to serve as reference for the programming language. As for any other coding language, it's possible to declare and initialize all the variable at the beginning such as the specific tool used (*Tool #*), local reference frames (*Local #*, P_{origin}), points ($XY(X_p, Y_p, Z_p, U_p)$) and operation variable (as for C/C++). Motors have to be powered (*Motor On*) to move the machine. For *Point to Point* movements, speed (*Speed Val*) and acceleration/deceleration (*Accel valA, valD*) have to be called before the function (*Go Point*). In case of interpolated movement (i.e. *Move Point* for linear motion), the profile parameters can be set by *SpeedS and AccelS* likewise. The performances of the robot can be switched between low and high power (*Power Low/High*) during the routine. Loop-cycles (*for/while*) can be used to operate the machine. To start a TCP/IP handshake with the robot as client, the "*OpenNet # As Client*" needs to be instated once set in the controller parameters the IP and Port.
Secondly, a function to generate the robot program automatically from the G-code or trajectory is required. The first lines of the code switch the motors on and open the Ethernet communication; if successful, a subroutine is executed that manages the

bring up of the tool and pauses the main function til the objective condition is met. Power is limited for the first movement (approach to the print bed). The instructions to the extruder are send before the continuous move function. To retract and recover the filament, a pause is used to put on hold the robot while the extruder is controlled. Fig.3.9 places side by side a writing function with the final result, highlighting the direct correspondences.



```
code{1}=['Function ',file_name];
code{numel(code)+1}='   If Motor = Off Then';
code{numel(code)+1}='       Motor On';
code{numel(code)+1}='   EndIf';
code{numel(code)+1}='   OpenNet #202 As Client';
code{numel(code)+1}='   WaitNet #202';
code{numel(code)+1}='   Wait 1';
code{numel(code)+1}='   Call Extruder';
code{numel(code)+1}='   Power Low';
code{numel(code)+1}='   Speed 50';

for i=1:size(P_signed,2)
    if i ==1
        % first movement parameters
    else
        % rest of the print logic
    end
end

code{numel(code)+1}=' ';
code{numel(code)+1}='Fend';
code{numel(code)+1}=' ';
```

```
Function test
    If Motor = Off Then
        Motor On
    EndIf
    OpenNet #202 As Client
    WaitNet #202
    Wait 1
    Call Extruder
    Power Low
    Speed 50

    Print #202, "G-22.5225n"
    Wait 0.02
    Print #202, "G0n"
    Go XY(80.004, 80.787, 0.3, 0 )
    Power High
    AccelS 3000, 3000
    Print #202, "G56.3063n"
    Wait 0.05
    SpeedS 15
    Print #202, "G0n"

    Print #202, "G2247.4157n"
    Move XY(81.753, 79.328, 0.3, 0) CP
    ...
```

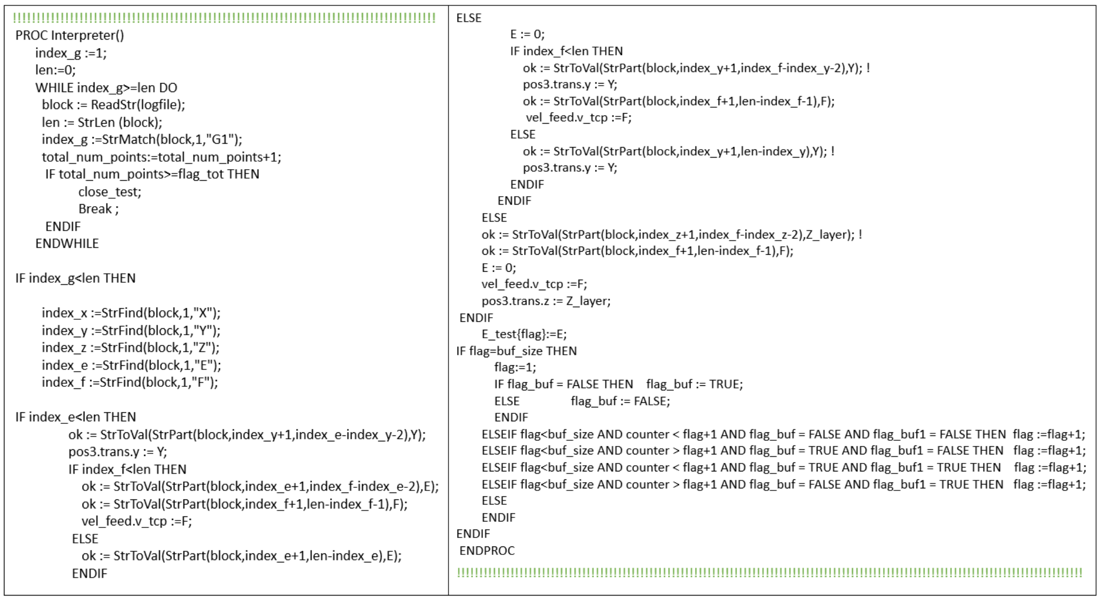
Figure 3.10: Example of *Epson* program generator in Matlab

In-machine interpreter

In this variation, the G-code file is uploaded on the robot controller (provided enough memory size). The less time-consuming approach, operation wise, requires fixing the G-code flavor and therefore the interpreter. Goes without saying that it has to be implementable in the controller (it is necessary to have a string library). If possible, a buffer of motion commands should pace the consecutive points to yield a smooth path without delays caused by the interpreter. Robot-tool synchronization can be obtained using callbacks linked to an interrupt associated to a percentage of the current total rise covered. Let's consider a specific case, an *ABB* anthropomorphic robot that has to work as a 3D printer (for clarity's sake, the project for which it has been developed strives to create a concrete printing cell).
Initially it was verified that a string library was available on the controller (basic functions are always accessible to manage serial messages) with a set of functions to parse a single command block. The flavor chosen is the one that gives as generic block:

$$G1 \quad X19.814 \quad Y-32.537 \quad E37.99705 \quad F45$$

The E address block can be related instead of the *[mm]* of filament to the volumetric flow rate or to the control action to forward to the device during the G-code writing phase in order to reduce real-time computation time. Fig.3.11 is the draft of the code implemented to print by reading a file (with extension changed to *.DOC*) uploaded on the controller via USB flash drive inserted on the teaching pendant. When called by the main routine, it reads line by line until it finds the first available *G1* block. It continues by getting the index of each address block within the char array; this information is used to convert the slice of string between to known consecutive address blocks into floats. Position, tool and speed data are updated accordingly. If adopted, the last part of the function manages the buffer. If the file has run over lines to be read, the controller executes the subroutine dealing with safe machine (robot plus tool) homing and stop.



```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC Interpreter()
     index_g :=1;
     len:=0;
     WHILE index_g>=len DO
       block := ReadStr(logfile);
       len := StrLen (block);
       index_g :=StrMatch(block,1,"G1");
       total_num_points:=total_num_points+1;
        IF total_num_points>=flag_tot THEN
                close_test;
                Break ;
        ENDIF
     ENDWHILE

IF index_g<len THEN

       index_x :=StrFind(block,1,"X");
       index_y :=StrFind(block,1,"Y");
       index_z :=StrFind(block,1,"Z");
       index_e :=StrFind(block,1,"E");
       index_f :=StrFind(block,1,"F");

IF index_e<len THEN
            ok := StrToVal(StrPart(block,index_y+1,index_e-index_y-2),Y);
            pos3.trans.y := Y;
            IF index_f<len THEN
               ok := StrToVal(StrPart(block,index_e+1,index_f-index_e-2),E);
               ok := StrToVal(StrPart(block,index_f+1,len-index_f-1),F);
               vel_feed.v_tcp :=F;
             ELSE
               ok := StrToVal(StrPart(block,index_e+1,len-index_e),E);
             ENDIF
ELSE
               E := 0;
               IF index_f<len THEN
                  ok := StrToVal(StrPart(block,index_y+1,index_f-index_y-2),Y); !
                  pos3.trans.y := Y;
                  ok := StrToVal(StrPart(block,index_f+1,len-index_f-1),F);
                   vel_feed.v_tcp :=F;
               ELSE
                  ok := StrToVal(StrPart(block,index_y+1,len-index_y),Y); !
                  pos3.trans.y := Y;
               ENDIF
            ENDIF
       ELSE
       ok := StrToVal(StrPart(block,index_z+1,index_f-index_z-2),Z_layer); !
       ok := StrToVal(StrPart(block,index_f+1,len-index_f-1),F);
       E := 0;
       vel_feed.v_tcp :=F;
       pos3.trans.z := Z_layer;
 ENDIF
       E_test{flag}:=E;
IF flag=buf_size THEN
         flag:=1;
         IF flag_buf = FALSE THEN    flag_buf := TRUE;
         ELSE              flag_buf := FALSE;
         ENDIF
       ELSEIF flag<buf_size AND counter < flag+1 AND flag_buf = FALSE AND flag_buf1 = FALSE THEN  flag :=flag+1;
       ELSEIF flag<buf_size AND counter > flag+1 AND flag_buf = TRUE AND flag_buf1 = FALSE THEN   flag :=flag+1;
       ELSEIF flag<buf_size AND counter < flag+1 AND flag_buf = TRUE AND flag_buf1 = TRUE THEN    flag :=flag+1;
       ELSEIF flag<buf_size AND counter > flag+1 AND flag_buf = FALSE AND flag_buf1 = TRUE THEN   flag :=flag+1;
       ELSE
       ENDIF
ENDIF
 ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Figure 3.11: Example of a G-code interpreter in *rapid* (*ABB* language)

3.2.2 *Data Link*

As said previously, this solution requires an additional node in the control chain that could be either a computer or a device with Ethernet interface that will send a line (or batch) of instruction to control the robot over TCP/IP. This solution (and broadly the family of remote control) is adopted every time that target positions keep changing during the job (for example pick and place with vision system to cope with a conveyor without well-defined distribution) and requiring computations that can overburden the PLC (i.e. image processing). It's self-explanatory that the number of code line required diminishes but managing the timing of the new target request and execution is crucial to yield smooth movement. Let's consider a specific case, a *Mitsubishi* anthropomorphic robot that has to work as an FDM 3D printer.
The data returned by the slicer (either G-code or raw trajectory) is post-processed and has to be send over Ethernet with TCP/IP protocol. The data packet, in the minimal

form, has 7 elements; it is indeed constituted by the X-Y-Z coordinates followed by the A-B-C to impose the orientation of the tool and the commanded velocity (feed). Orientation is dependent of the convention used by the controller (Roll-Pitch-Yaw, axis-angle or quaternion), in this case RPY.
For the remote controller node (a computer), all major programming languages can open a TCP/IP port. In *MATLAB* the communication is started using the function *tcpip(IP,port)* and by setting the terminator and the timeout. In Python and C++ with the *socket* and *winsock2* libraries.
For the robot, the main elements of the code can be seen in Fig.3.12. It is split in two

Task slot 1

```
*Repeat
M_00# = 1
Wait M_00# = 2

If M_01#=1 Then
   Ovrd 10
   Mov P_02
   M_01# = M_01#+1
   M_03# = 1
   Ovrd 100
Else
   Cnt 1,1,1
   Spd M_02#
   Mvs P_02 WthIf M_Ratio>99 ,M_03#=1
EndIf

GoTo *Repeat
```

Task slot 2

```
*WaitTrigger
Wait M_00# = 1

Input #1,MXTel,MYTel,MZTel,MATel,MBTel,MCTel,MsPEEDTel
P_02.X = MXTel
P_02.Y = MYTel
P_02.Z = MZTel
P_02.A=Rad(MATel)
P_02.B=Rad(MBTel)
P_02.C=Rad(MCTel)
M_02#=MsPEEDTel
M_00# = 2

Wait M_03#=1
Print #1, "OK"
M_03#=0
GoTo *WaitTrigger
```

Figure 3.12: Example of a DL control code for *Mitsubishi* robot

threads (parallel computing): one to move the robot and one for the communication. A variable is used to synchronize the operation. The *slot1* set it to ”1”, the value that the *slot2* has been waiting. This triggers the input function that reads the first block available from the incoming buffer; each variable (X,Y,Z,A,B,C,F) is stored in a temporary memory block before updating the position variable (*P_02*). The synchronization variable is then increased to ”2” that signals back to the *slot1* to continue with the preparatory and movement lines. If it’s executed for the first time then a specific routine is used (joint movement, *Mov*, at lower speed for the approach to the print area). All the other occurrences are linearly interpolated (*Mvs Target*) assuring a minimum smoothing operation on the path (*Cnt*). A request variable is set to ”1” when the robot has completed a percentage (user defined) of the motion (*WthIf M_Ratio* $>$ *%*); *slot2* therefore sends a message to the computer to issue a new target.
The overall printing procedure outline can be found in Fig.3.13. The flowchart also includes the preparatory phases with G-code loading, parsing and reference string encoding (both the robot and extruder).

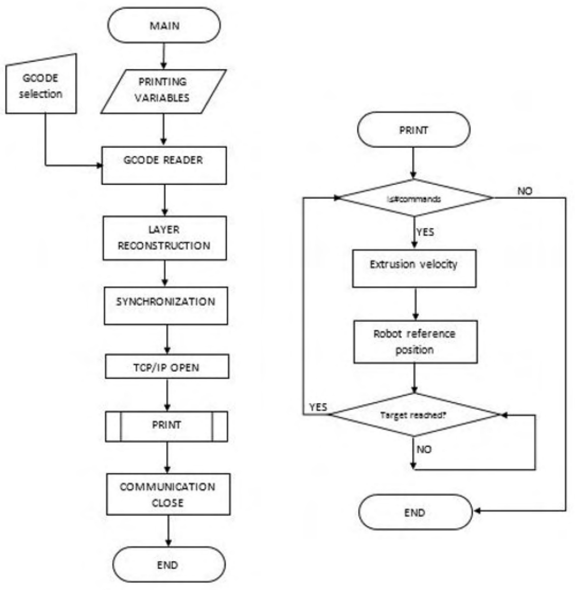


Figure 3.13: Simplified flow chart for DL

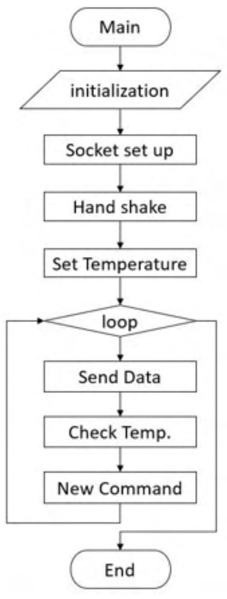


Figure 3.14: Flow chart for RT (C++)

3.2.3 *Real Time*

The *DL* approach requires to adopt a strategy to pace the target packages sent, causing plausible hiccups in the printing process. Moreover, this method prevents from the control of the actual profile, it's the robot's manufacturer that has indeed established the motion profile (trapezoidal speed or acceleration) and how to menage a series of movements. Hence, some robot controller allows the user to define the rise for each cycle time by retrieving the reference position at real-time and executing it via Ethernet.
For the *Mitsubishi* controller, the same of the previous example, the cycle time is 7.11 [ms] (for some version 4 [ms]). To operate in this mode, it is required the adoption of a specific UDP packet, determined by manufacturer; it is a c++ *struct*, containing all the data that can be overwritten or accessed, copied (or serialized) in a char array (byte representation of the original *struct*). The size of the data package is fixed to 196 bytes, either for monitoring or instructing and the ordering method is little-endian.
In this configuration, an executable (*.exe*, written in c++) is used to generate and send the data packet according to the prescribed structure. The trajectory sampled every 7.11[ms] and stored in a text file (although it could be streamed via localhost as well to interconnect software) is read one line at a time at every cycle. The trajectory has to be computed by the user in compliance with the specifics of each element of the cell. Real time (or low-latency) kernel guarantees a constant transmission frequency, although communication process is designed to cope with throughput-oriented kernels. The main lines of the robot program reduce down to opening the port (for example *Open "ENET: 127.0.0.1" As #1*) and following the incoming data (*Mxt 1,0*).

3.3 Offline simulation

In the previous section, the possible strategies to turn an industrial robot into a 3D printer have been discussed. When setting up the system or simply to check which solution better suits a given machine, the opportunity to work with a digital copy of the physical system is always advised. For these digital twins, it is not only important that they match the mechanical aspects of the real system, but especially those concerning movements and communication; this allows to debug the process without harm.
Here the possible combination, between physical and simulated systems in AM, are listed pointing out some remarks. The dissertation, afterwards, will vert solely on the robot.

- *Virtual robot - Virtual Extruder:* Consider each system on its own. Used solely during the early stage of the assessment of the robotic system. The offline extruder requires a code capable of reproducing the reading and the generation of the output signals (pulse-width modulation, PWM, of the stepper motor driver and of the MOSFET regulating the current flowing to the heat cartridge).

- *Real robot - Virtual Extruder:* This case could be used when the robot is reading the G-code and mastering the extruder. It's advised during the first robot tests to avoid that the heated element fortuitously collides with any object present in the cell.

- *Virtual robot - Real Extruder:* This combination can be used to validate the extrusion system during the print while keeping it still (see Fig.3.15).

- *Real robot - Real Extruder:* the system is printing.

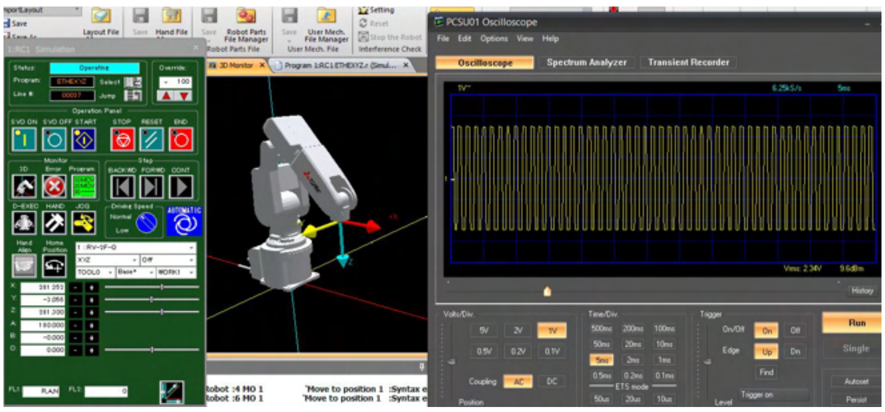


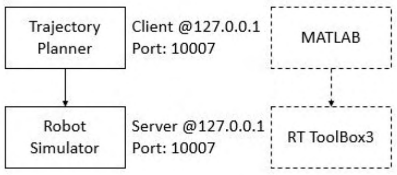
Figure 3.15: *Virtual robot - Real Extruder*: the PWM is read by means of an oscilloscope

The robot (Mitsubishi RV-2F-Q), is going to be used as example for the offline simulation of a control architecture. For other robots, the tackling strategies just need to be adapted accordingly.

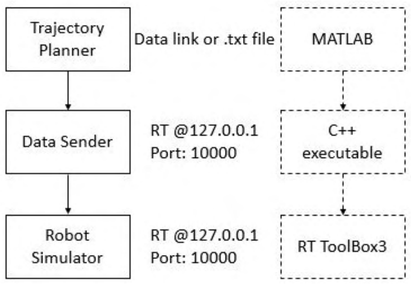
3.3.1 Mitsubishi simulation of *Data Link* and *RT*

The manufacturer software (*RT Tool-Box3*) is used as simulating environment because it offers, offline, all the functionalities that the real system possess (the one that are going to be extensively used are TCP/IP communication, motion planner and oscillograph to acquire the data). In order to mimic the control architectures offline, the only alterations required concern the LAN setup. The IP addresses identifying the robot ad the computer become one, that is, since all the talking has to happen internally to the PC, *127.0.0.1* (i.e. localhost). The communication is finally established by indicating the port, in both software, according to the scenario selected; this resolves into the following cases.

- Data Link: In *RT ToolBox3*, the *IP, Mode, Protocol* and *Packet type* parameters need to be customized to match the ones demanded by the network (port 10007 in Fig.3.16(a)).
- Real Time: The default port for real time is the *10000*, so only the network IP has to be set (Fig.3.16(b)).



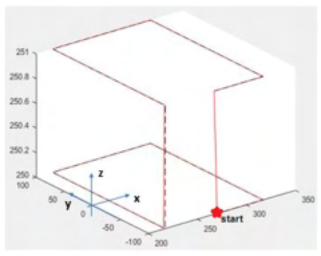
(a) Data link



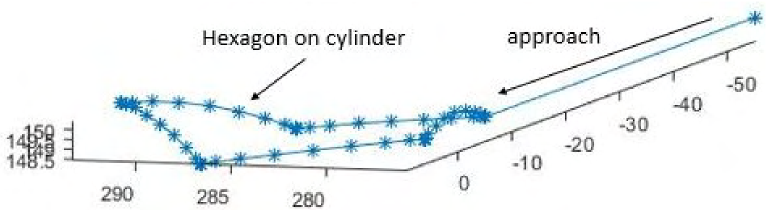
(b) Real time

Figure 3.16: Off-line control scenario: data link and real time

Both *DL* and *RT* are going to be deepened for both planar (conventional) and non-planar layer. The former is tested with a simple two-layer square (see Fig.3.17(a)) with 35.16 [mm/s] feed; the path is made up of a series of points positioned at the corners of



(a) Unidirectional trajectory



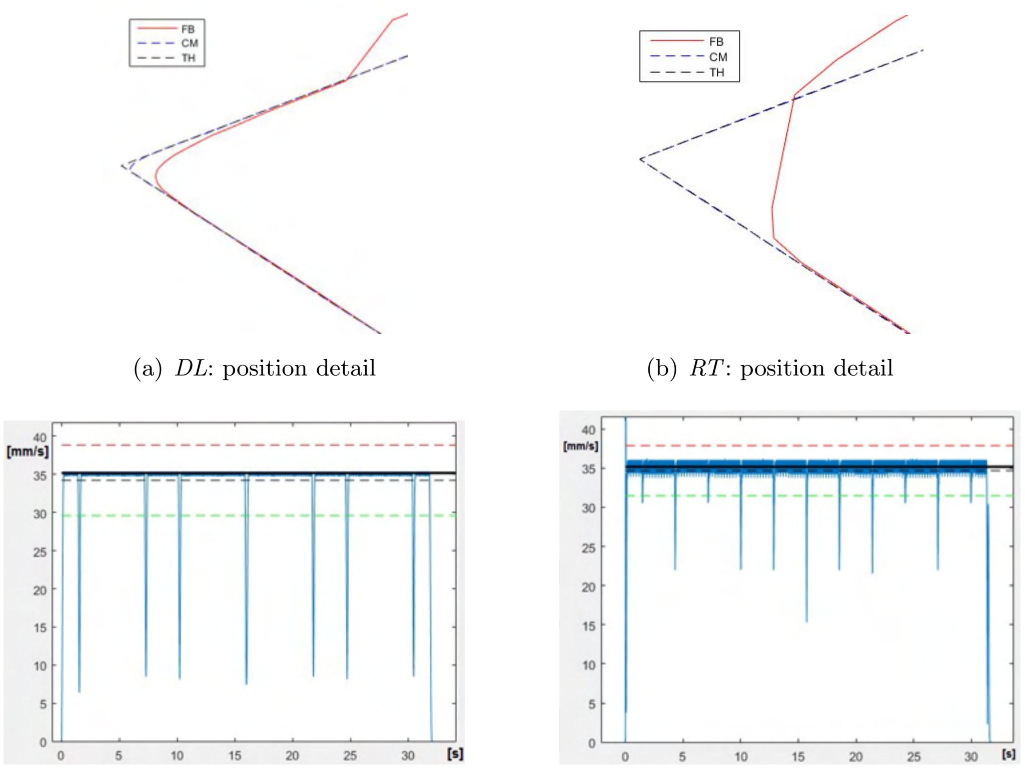
(b) Multi-directional trajectory

Figure 3.17: Representation of the test cases trajectories

the squares. The latter (see Fig.3.17(b)) has to be followed at 15 [mm/s]. The *RT* path is obtained by transferring (decal) a pentagon on a cylindrical surface (the procedure is detailed in appendix A).
Both motions have been acquired using the oscillograph function running in RT mode (thus sampling at 141Hz).

Planar-layer printing (in $\mathbb{R}^2$)

In Fig.3.18(a), it's possible to see a corner of the square using the *DL* approach (it is configured to transit from one instruction to the other using a 1[mm] radius circumference.); in Fig.3.18(b), the same feature, but with the *RT* strategy. What can be noticed is that even RT give rise to a smooth corner despite the fact that was not instructed, thus a motion planner is still applied to the data via UDP. In both cases a positioning error is present.
In Fig.3.18(c) and 3.18(d), the respective speed module profiles (evaluated along the path), obtained by numerical derivation, can be seen. The *DL* shows less bounded fluctuation ($\pm 1[mm/s]$) around the feed then the *RT*, although it tends to approach zero speed along the path more markedly (causing a loss of synchronization between the flow rates).



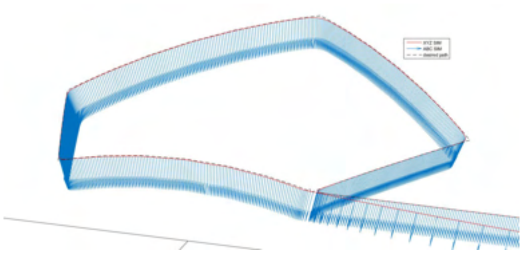
(a) *DL*: position detail (b) *RT*: position detail

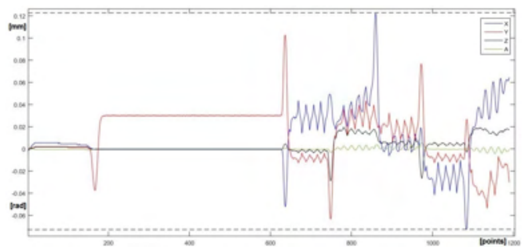
(c) *DL*: speed (d) *RT*: speed

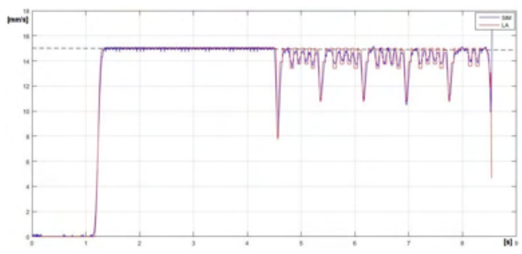
Figure 3.18: Unidirectional printing simulation: *DL* vs *RT*

Multi-directional printing (in $\mathbb{R}^3$)

Real Time case The motion law (at the TCP) is designed offline by means of a look-ahead algorithm (based on [33]) and discretized with a 7.11 [ms] interval.
In Fig.3.19(a) the position is depicted while in Fig.3.19(b) the positioning errors (only the rotation around the X-axis (A) are shown since the cylindrical surface is oriented in that direction, hence any normal vector on the surface will intersect the X-axis). In Fig.3.19(c)-3.19(d), the speed profile simulated is superimposed to the theoretical one and the error profiles are shown, as previously the robot controller performed a smoothing operation on the commanded points.


(a) Simulated curve


(b) Coordinates error


(c) XYZ resulting speed

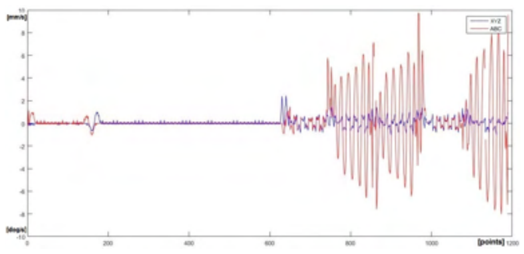
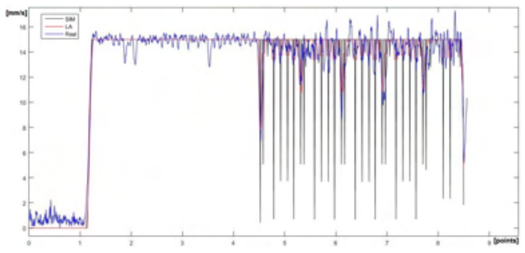

(d) Speed error

Figure 3.19: RT: simulation

Running the RT example on the physical system, outputted similar results to the one simulated, but with the introduction of terms unmanageable by the simulator, strictly related to the physical system considered (Fig.3.20).


(a) XYZ resulting speed

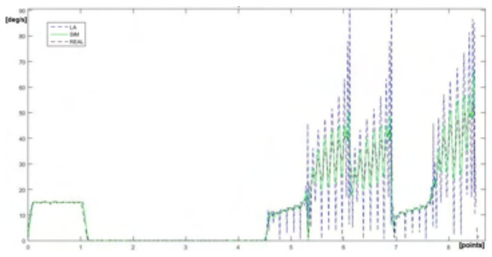
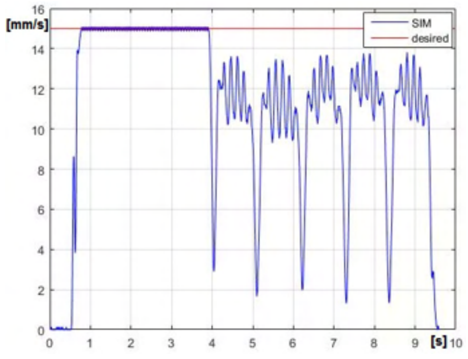
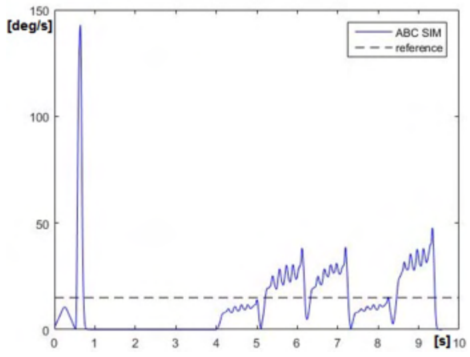

(b) ABC speed

Figure 3.20: RT: real

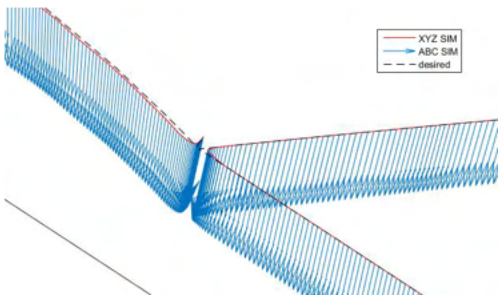
Data Link case Fig.3.21(a)-3.21(b) report the speed for the simulation for this method. What can be inferred is that the system is not capable of guaranteeing the feed speed during the build on the cylinder giving the sampling distance chosen. One possible solution could be to let the system know more than one point at a time, therefore developing a more sophisticated mean to synchronize the speeds. Another possible fix could be to use the simulation itself to predict the correct extrusion speed. A third solution could be to impose the speed only on the path, while discarding the orientation, for the computation of the speed profile. At the end in Fig.3.21(c), it possible to inspect the deviation from the one desired.



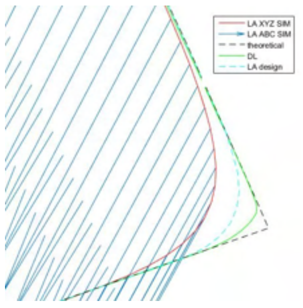
(a) XYZ resulting speed



(b) ABC resulting speed



(c) Simulated curve



(d) Trajectory commands vs simulation

Figure 3.21: DL: simulation

Comparison In Fig.3.21(d), the difference among the theoretical curve, the one evaluated with the Look Ahead algorithm (adding also smooth transition at the corners) and the results of the simulations is portrayed. A corner is enlarged because it's the most delicate situation since it has to be assured that the robot arm moves smoothly without under or over extrusion.

In conclusion to sum up what the simulation have shown: the *DL* method is less affected by speed fluctuations when performing unidirectional printing; the *RT* is able to follow more closely the speed profile designed along the path while reorienting the tool.

3.3.2 Stand-alone: Case study

It's worth reporting examples also for this remaining strategy for the sake of completion. A simulation for the ABB solution has been carried out analogously to validate the correct interpretation of the command blocks as well as the management of the buffer to synchronize TCP movement and extrusion feed. The virtual environment where the test is conducted can be seen in Fig.3.22.

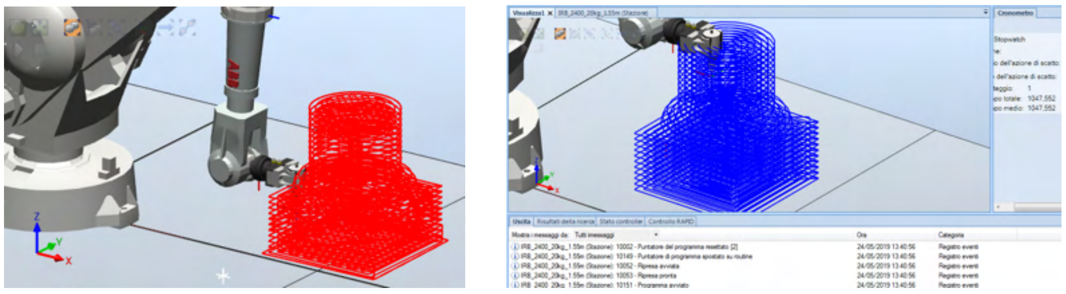

```
Mostra i messaggi da:  Tutti messaggi                                          Ora                   Categoria
IRB_2400_20kg_1.55m (Stazione): 10002 - Puntatore del programma resettato [2]  24/05/2019 13:40:56  Registro eventi
IRB_2400_20kg_1.55m (Stazione): 10149 - Puntatore di programma spostato su routine  24/05/2019 13:40:56  Registro eventi
IRB_2400_20kg_1.55m (Stazione): 10052 - Ripresa avviata                        24/05/2019 13:40:56  Registro eventi
IRB_2400_20kg_1.55m (Stazione): 10053 - Ripresa pronta                         24/05/2019 13:40:56  Registro eventi
IRB_2400_20kg_1.55m (Stazione): 10151 - Programma avviato                      24/05/2019 13:40:56  Registro eventi
```

Figure 3.22: ABB print simulation

Analogously, the simulation for the UR3 (using a strategy identical to the one presented for the Epson) has been done. In Fig.3.23, the path and speed profile for a given geometry are reported as a function of the motion parameters. Data is acquired using the Real-Time Data Exchange (RTDE) that generally outputs messages on 125 Hz (the real-time loop in the controller has a higher priority than the RTDE interface); it resembles the Mitsubishi's Real Time control/monitoring (subsection 3.2.3).
The simulations help tailor the post-processing algorithm of the trajectory planning to get the optimal configuration.

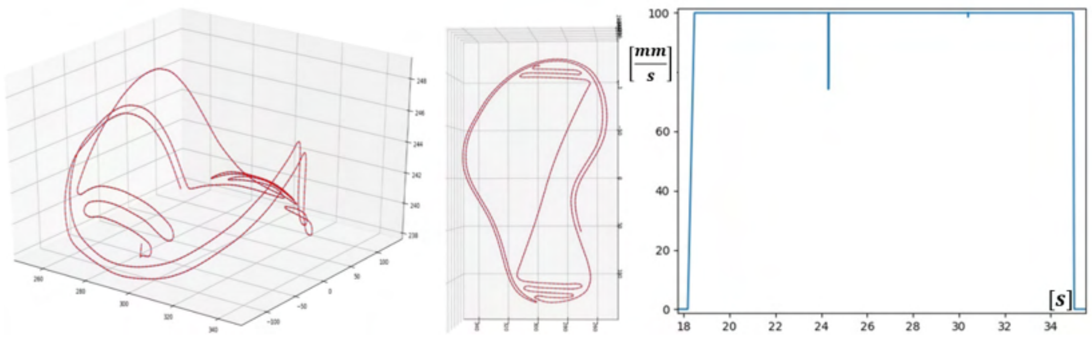


Figure 3.23: UR3 simulation with acceleration equal to 0.35 $[m/s^2]$, speed to 100 [mm/s] and blending radius of 5 [mm]

3.4 Testing phase

Once completed the designed the the extrusion system (hardware comprehensive of the flange and relative firmware) and the control of the industrial robot, few basic tests are conducted to validate the solution proposed in a practical application. To do so, the first test consisted in printing a simple shape (i.e. square) with a robot speed of 7.5mm/s and an extruder temperature of 220°C. In the second run a more complex shape (i.e. duck) is selected to test the behavior of the cell for longer times (approximately 4h) using the same process parameters; the temperature is checked every minute to monitor the print and promptly stop it if the temperature exists the optimal region. For both cases conventional printing is adopted, and the building is carried out using a client application (coded in Matlab) that masters both robot (Mitsubishi RV-2F-Q) and extruder (the method used is described in 3.2.2). The results are reported in Fig.3.24.

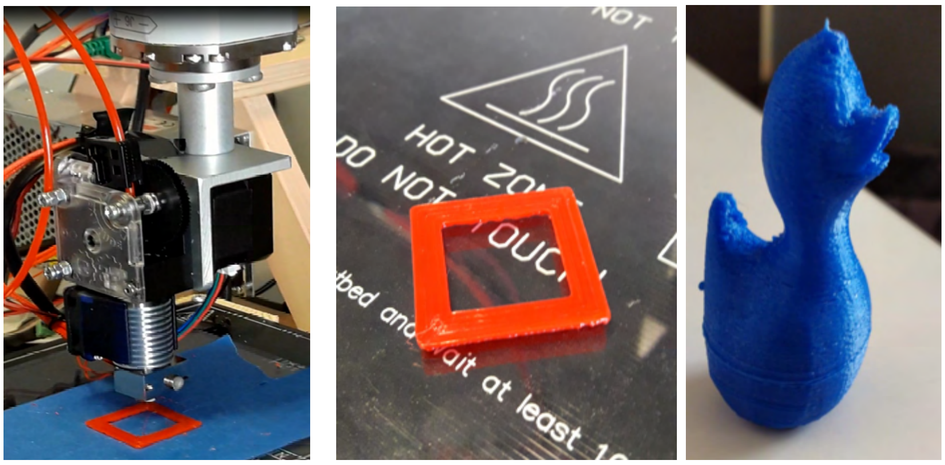


Figure 3.24: extruder first run: square shape

Chapter 4

Trajectory planner

The result of the two previous chapters is the physical medium to 3D print the part, but one last part is missing: how to plan the manufacturing processes. Without these components, it would not be possible to test conventional or novel building strategies on the setup.
As detailed in chapter 1, commercial slicers are coded to operate with the conventional *layer by layer* approach with 3-axis machines in mind; multi-directional planners, on the other hand, are still confined in the academic and research field. Therefore, even for standard printing, a piece of software is required to translate G-code into robot instructions and to check the overall process feasibility.
Hence, the planner required is not a mere slicer, but it has to account for the cell configuration (fixed or mobile tool), collisions, offsets and range of motion. This gives birth to *RSS* (Robotic System Slicer), a software whose intent is to offer an all-in-one solution to operate a 3D printer based on an industrial robot. The final user will be able to customize the manufacturing process or let the automated code (based on the experiences of this research activity) take care of it.

The main steps behind the version 0.1 of the *RSS* (both in Matlab and Python language) for the generation of the toolpath are here summarized.
The input is a CAD model of the geometry to be realized in the format of an *.stl* file. It then has to undergo to a pre-processing routine to repair possible defect in the model as well as to reduce the size of the variables storing the mesh (section 4.1).
Once completed the operation just related to the input data, the geometry needs to be positioned in the workspace to comply with the process chosen (section 4.2). The locus that represent the cutting surface has to be inferred according to the substrate and building strategy (section 4.3). Then, the contour resulting from the intersection between the model and the cutting surface is generated (section 4.4); for the same layer is possible to impose the desired infill (section 4.5).
The curve (point and normal direction), outputted from the previous stage, is the input of the following operations that are: tool reference system definition, motion law assignment (both belonging to the section 4.6), collision (comprehensive of self-collision)

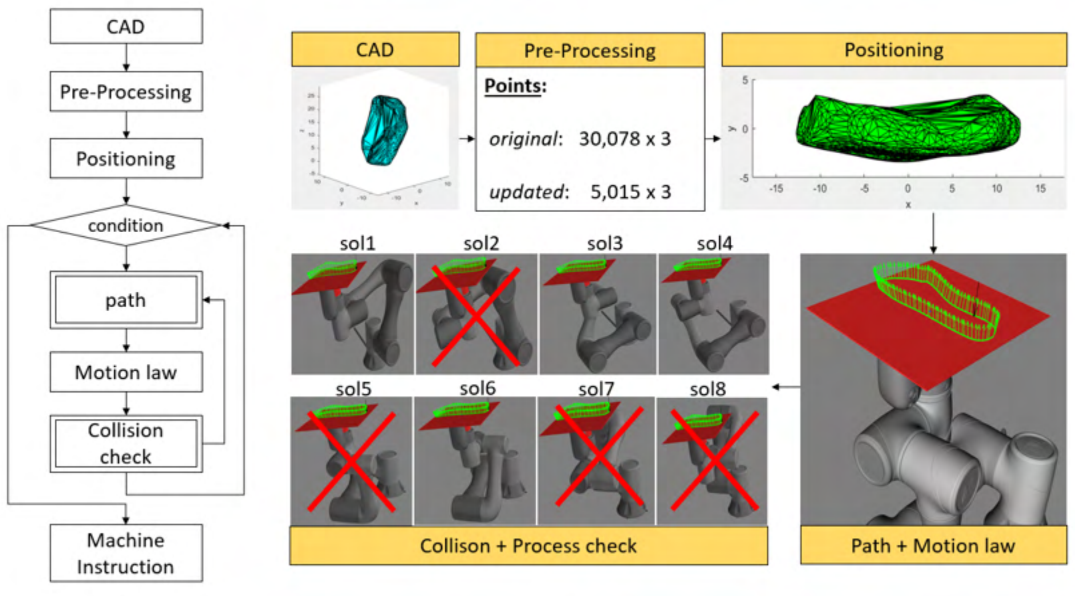


Figure 4.1: Flowchart of the generation of the machine instruction by means of Robotic System Slicer (*RSS*)

and process check (subsection 4.6.3). These phases have to be carried out until a feasible solution is obtained. Once defined the first deposition (this method can be used also for SM, hence subtraction of material) pass, the process is iterated until the entire geometry is processed. The end product is translated in machine instruction according to the control strategy opted from the previous chapter for the robotic system (section 4.7). Fig.4.1 depicts the phases of the *RSS*.

During the infancy of the activity, some considerations have been done to outline the development of the code.
Two are the main kernels of the software: path generation and collision check. It can be stated that the path generation (i.e. slicing) is nothing other than a collision between an object and a surface; therefore, it derives that actually only the collision routine has to be coded. The difference lies on the data type returned by the function; a Boolean to represent the hit/no-hit status or an array of floats storing the exact contact points (i.e. the contour and relative normal vector to the surface). This approach deviates from what has been found in literature about AM slicing, allowing a reliable tool to obtain the reference contour using the standard AM CAD format, suitable for all the possible cutting planes seen up until now and applicable to SM (for example milling) in Hybrid manufacturing process.
The software rarely relies on existing code to solve a specific problem to avoid limitation on the possibility to customize the routines at this or in future stages. The phases of the *RSS-ver.0.1* are here discussed.

4.1 Pre-processing

The first operation that the software has to carry out is the correct interpretation of the STL file.
Geometry stored in the STL might require repairs to be manifolds; several CAD software offer this function that should be considered to avoid uncertainties in the following steps (for instance, contour closure).
Matlab introduced the *stlread* function in the R2018b release. As concerns Python, there is the *numpy-stl* library, available in the *PyPi* repository, that manages the load operation. They both accept ASCII or binary STL file and returns the relative triangular mesh (connectivity list and vertexes). Normal vectors are also returned by the reader or they can be directly computed from the definition of each triangle.
The loaded data might contain repetitions of the same points (see chapter 1, Fig.1.10) that slows down the program, therefore it is processed to reduce it to a minimal representation. This can be done by finding the index of the repeated points for each position in order to update the connectivity list. If implemented, for a starting geometry of 1926 points only 325 unequivocally describes the manifold. As a rule of thumb, the points are decreased of one sixth (tested on 50 STL files).
If only a portion of the file is going to be interested by the following phases, cuts of the manifold can be a viable mean to reduce computational cost (this approach will be exploited in 7.1).

4.2 Positioning

Once imported the manifold, it has to be positioned in the workspace so that it is manufactured wisely. This resolves in placing the object on the print bed so that support is minimized, layers are oriented to yield good mechanical properties and enough region is in contact with the bed to assure proper adhering of the first layer.
This task is usually carried out by the operator manually that evaluates according to his experience and knowledge the suitable pose of the object in the operational space.
Some strategies have been studied to make the process automatic. They differ only on the technology that is going to be exploited, for instance milling might require the location to minimize the machined part and fit the starting block.
The simplest mean to orient the part is based on the minimization of the residual area of the projections on the XY and XZ plane. It starts by considering just one plane (i.e. XY) and in the specific the projection of the hull on such plane. The resulting polygon has to be *simple*, that is without self-intersection nor holes, in order to use Gauss' formula to calculate the area. A bounding box is circumscribed to the contour, that in 2D is just a rectangle. Now the area of the box and the one of the polygons are evaluated and subtracted. The difference is stored in an array. In order to minimize this residual area, the polygon is turned by a step increase (selected during the function call) around the Z axis. In this new configuration, the bounding box is update and

therefore their difference. The new value is appended to the array. Once completed the range indicated of possible rotations (180 degrees is the default), the pointer corresponding to the minimum value returns the angle that the manifold has to be rotated around the Z axis. Analogously, the procedure is done for the XZ plane and Y axis. Fig.4.2 depicts the main phases from the initial configuration to the re-positioned one. This algorithm is well suited for milling of prosthetic devices for craniectomy surgery

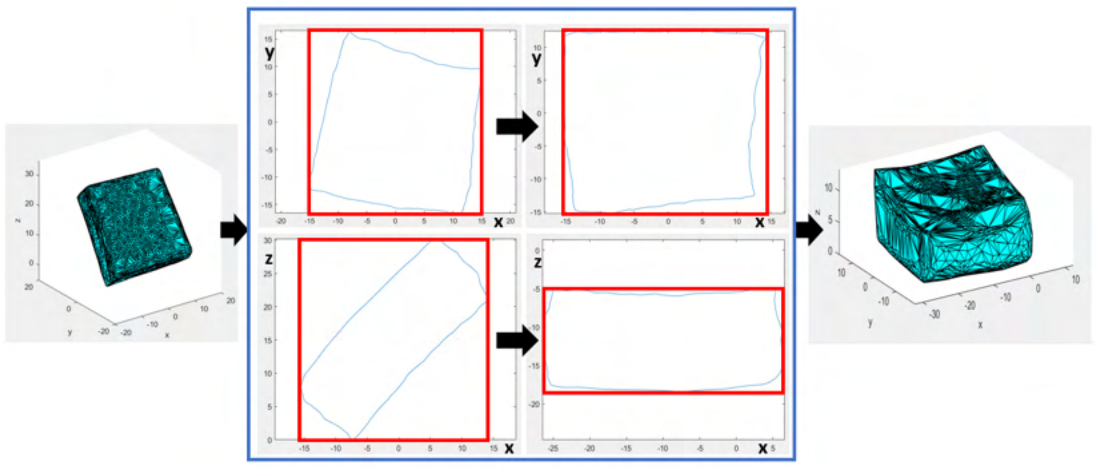


Figure 4.2: Positioning: residual area minimization method

(see section 7.2) because geometries do not have holes or overhanging features. In all the other cases minimization has to be backed up by other objectives to account for overhangs and holes.

4.3 Cutting surfaces definition

Before developing any line of the slicing algorithm, the types of cutting surface, it should be able to work with, have to be characterized. Their definition significantly impacts the computation required to extract the information requested.
In the section 1.3.1, the possible cutting surfaces have been presented. It's known that the digital model of the object to be printed is in the STL format. Therefore, the software has to have access to at least one library able to menage triangles and surfaces made up of this type of lattice.
As seen in chapter 1, layers in conventional printing are described using the equation (B.2), page 167. With this kind of representation, only the definition of one characteristic point, whose elements are null with the exception of the Z axis, is needed because the normal direction is constant throughout the print and defined ahead of times (that is the normalized normal vector of the substrate).
When the cutting surface stops being a plane, the substrate and the model have to be described in the same fashion, in this case the STL format (triangular mesh). Hence,

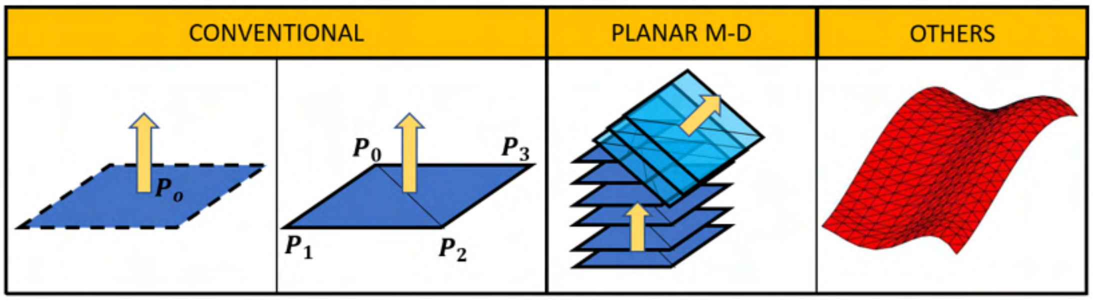


Figure 4.3: Cutting surfaces examples using meshes: (*Conventional*) planes are reduced to a portion delimited by four points, hence two triangles; (*Planar Multi-Direction*) the entire layer structure can be computed without the need to split the model or risking leakages from one direction to the other; (*Others*) the surface is described as a set of triangles.

to uniform the code and to make it independent on the type of input, even the planes are defined as a portion of the original space by means of triangles. This operation requires a greater number of points (4 instead of 1) but allows to customize the regions in order to yield only a local slice of the model.
This makes the planar multi-directional easier to implement; it is only necessary to define the quadrilaterals according to the identified planes where the change is taking place as can be seen in Fig.4.3. This is only possible because the final set of layers are not overlapping.

4.4 Surfaces intersection

As stated earlier, the *RSS* uses a slicing algorithm that is similar to a collision detection one. The intersection of surfaces is a common problem in the collision field as can be evinced in the appendix B. Let's review the software implemented.

4.4.1 Algorithm outline

Given two meshes of a first surface M_1 and of a second one M_2, their intersection can be obtained following the flowchart in Fig.4.4.
First, a mesh partition algorithm is used to reduce the computational effort by considering only a subset of the overall part for both the printing surface and object (using Bounding Volume Hierarchy, *BVH*, discussed in appendix B). On this subset, the actual intersection is found by running a triangle-triangle intersecting algorithm. It returns a segment for each intersection. They are then sorted and connected to generate close loops (the original triangulation might not be indexed to reflect neighbor triangles and consequently a close polygon).

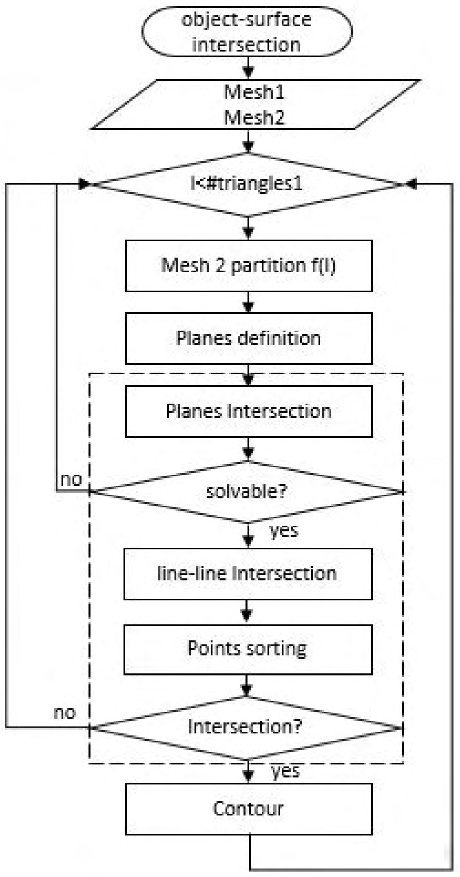


Figure 4.4: object-surface algorithm: flowchart. The part within the dashed rectangle is the triangle-triangle intersection algorithm.

4.4.2 Functions

Let's contextualize the subroutines hinted in the outline. They are *Mesh partition algorithm*, *Triangle-triangle intersection* and *Contour loop.*

Mesh partition algorithm: Each element of the triangulation is also describable as the circumscribed parallelepiped (with its maximum and minimum value for the x-y-z direction). This transformation allows to significantly reduce the computation time of pair-wise intersections by approximating a triangle with its bounding box because the overlapping criterion is simpler than the one for the straight triangles. These steps are depicted in Fig.4.5.
Having indicated with T the boxes of a first surface and L of a second, the hit criterion is the following:

1. If $T_{i,max}\Big|_{x,y,z} < L_{i,min}\Big|_{x,y,z}$ & $T_{i,min}\Big|_{x,y,z} > L_{i,max}\Big|_{x,y,z}$: no intersection.
2. *else*: the objects hit.

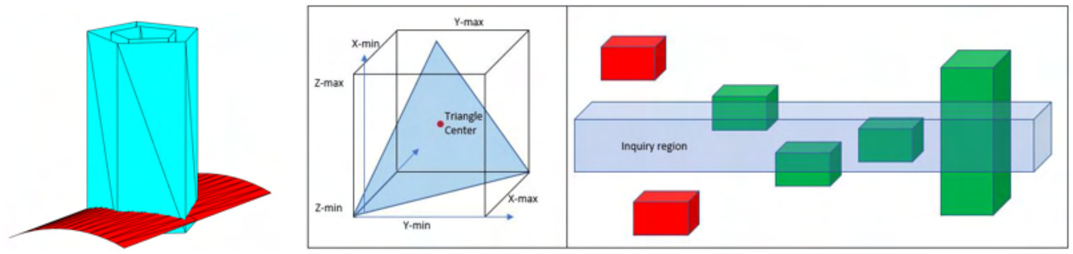


Figure 4.5: Triangle simplification and selection criterion of STL surfaces.

If the bounding box intersection returns a hit, it means that the two triangles might overlapping (necessary condition, but not sufficient). It is therefore required to further investigate the problem by solving the actual triangle-triangle intersection.

Triangle-triangle intersection: It can be divided into the following stages:

1. **planes definition**: both triangles, defined by the vertexes, are seen as planes instead of bounded regions.
2. **plane-plane intersection**: the shared line between the two planes, if it exists, is computed.
3. **line-line intersection**: the shared line is intersected with all the segment to get the common points.
4. **point sorting**: the point belonging to the two segments are sorted along the shared line to yield the final intersection.

The shared line between the two planes can be computed using linear algebra (for those interested, by solving the equations (B.5) and (B.6) at page 171). The existence of such solution depends on the non-singularity of a 3x3 matrix, $[\vec{n}_1; \vec{n}_2; \vec{n}_3]$, storing the normal vector of the two planes and their cross-product.
Each side of the two triangles is intersected with the shared line (solving equation(B.3), page 171). This resolves in:

$$[\vec{n}_1; \vec{n}_2] \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = [A]\underline{t} \quad = \quad dP = P_2 - P_1$$

Using the left pseudoinverse (because the matrix A is not square, $[A]_{3x2}$) the solution exists and it's unique (from literature) if:

$$A((A^T A)^{-1})A^T dP == dP$$

and equal to:

$$\underline{t} = (A^T A)^{-1} dP$$

If $\underline{t}_1 \in [0, 1]$, it means that the point found belongs to the segment. This algorithm requires the inversion of a matrix, that is time consuming; a valid alternative is to treat

the two lines as in a 2D space (either XY, XZ, YZ plane) to find the intersection point and to check if it holds in the remaining dimension. It is possible that all three projecting planes have to be considered before finding the solution otherwise the intersection does not exist [97].
There are three possible relationships between the two segments, one for each triangle (provided that the two triangles are not coplanar). The intersection can be evaluated by just considering the difference (Δ) of the index[1] of the first two elements of the sorted vector (the ordering direction does not influence the result).

1. If $\Delta = 0$: the segment between two points of a triangle are completely separated or meeting in one point (this case is discarded because will simply burden the following stages of the slicer).

2. If $\Delta \neq 0$: the segments overlay.

If *case 2* is met, then it's possible to determine the intersection as the portion of line between the inner points of the sorted vector. Fig.4.6 illustrates these possibilities.

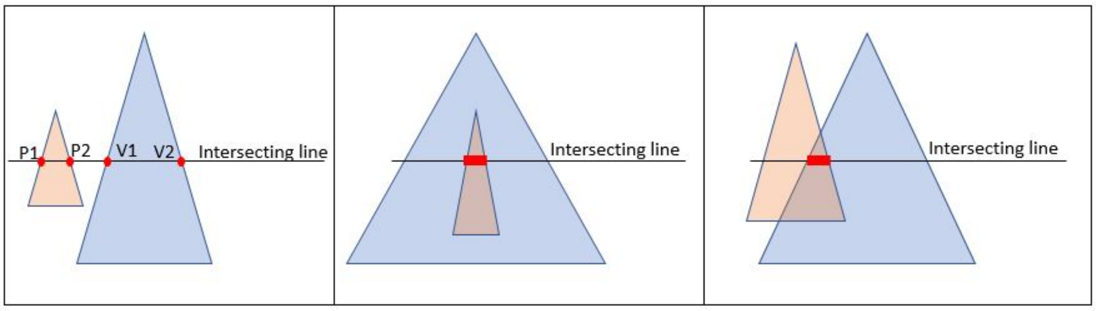


Figure 4.6: Possible scenarios of the triangle overlapping: *left*, no intersection; *center*, intersection with points belonging to the same triangle;*right*, the intersection is found between two points belonging to different triangles.

Contour loop The fact that the triangles have been processed progressively (in the order they've been uploaded) doesn't necessarily implies that the intersection segments form a sorted loop. The vector containing all these points is unconnected, randomly distributed and for a minimum of m points able to describe the profile it contains twice more (triangles sharing a side will generate at least two points), as can be corroborated in Fig.4.7.
To overcome this issue a contour reconstruction algorithm is required. The basic concept exploits the existence of double points (belonging to neighboring triangles) to determine the connecting segments. So given any initial segment, the algorithm finds the *j-th* points that minimizes $\|P - P_2\|^2$, where P_2 is the last vertex of the segment itself. It keeps iterating until the last found matches the first one. Therefore, one loop

[1]points belonging to the 1st triangle are indicated with 1, the others with 2.

must have been completed. The algorithm restarts with the remaining points. Fig.4.7 also depicts the result of such sorting operation.

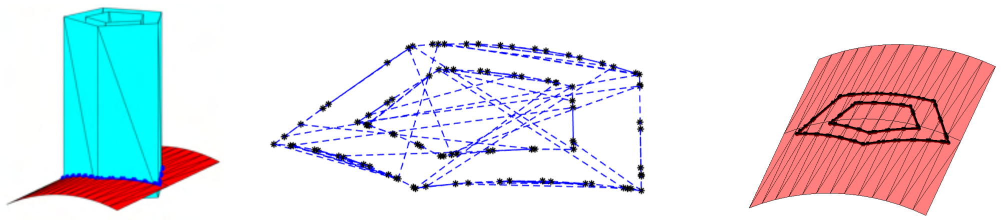


Figure 4.7: Triangle-triangle intersection result

Validation Let's apply the functions presented till now to simulate the print on a generic surface.
The two examples share the same random generated surface but differentiate in the model considered. The first example has a cube with a hole, Fig.4.8, that validates the contour reconstruction of two geometries. The second case, Fig.4.9, is used to debug the routines against a generic object.

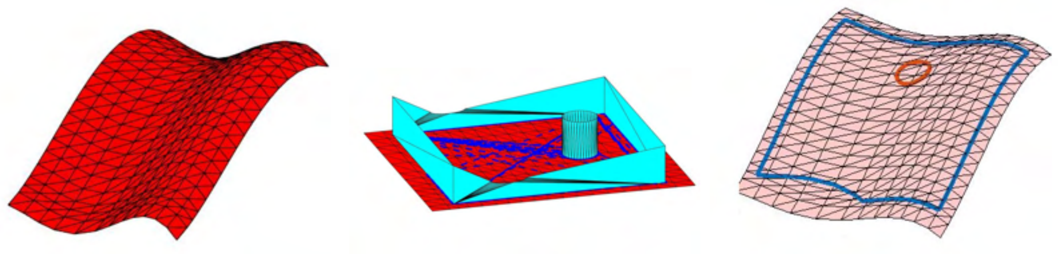


Figure 4.8: Intersection loci definition for a generic curve and a cube using the algorithm proposed. *right*, random surface; *middle*, both partitioned meshes; *right*, contours found.

4.5 Infill surfaces definition

Until this point, we have seen how it is possible to get the contour of the intersection between the cutting plane and the model, but unless interested in the build of the only hull it's not enough to carry out the AM process.
The print is in fact not only defined by the contour of the shape, but also how the inner part is going to be approximated. As observed in the state of the art, the generation of such geometries is not as manageable as in planar substrates.
A philosophical question arises: *is the inner part a characteristic of the original model or not?*
If the inner part is a property of the original shape, then it should be defined a priori by the designer of the CAD model without any knowledge on the building surface. This

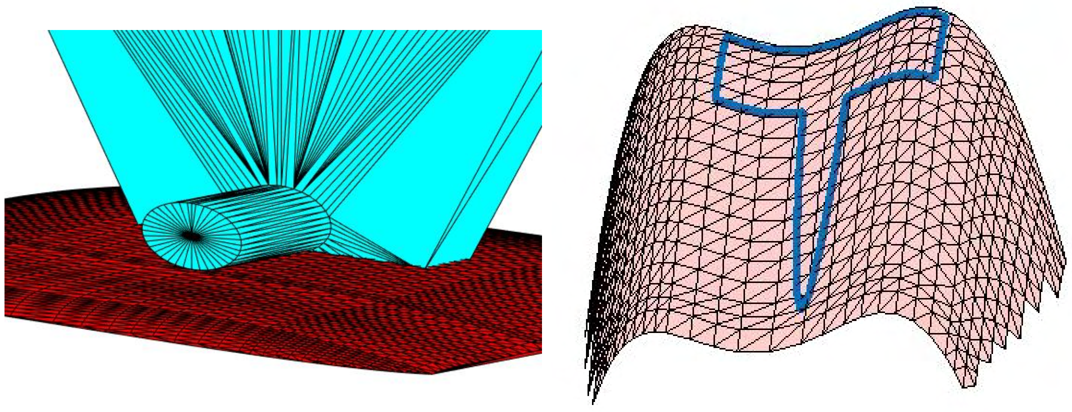


Figure 4.9: Intersection loci definition for a generic curve and object using the algorithm proposed. *left*, representation of the build; *right*, contour found.

implies that CAD software should be able to allow the user to define not only the hull but also the inner part. This leads to the definition of multiple CAD models by the designer, at least one per feature (i.e. hull, inner perimeter, infill), that can be sliced as seen previously.
If the inner part is not a feature of the original shape, then it will depend on the substrate and model. This requires algorithms to derive the CAD models of these features depending on the information at hand.
The latter case (because it requires additional stages) will be addressed hereafter through this section.

4.5.1 Additional perimeters

Perimeters can be duplicated to improve the surface quality, stability and toughness of the build. They are defined as an inner or outer wall, depending on the starting feature, that distance a given value (that remove the risk of air gaps between the passes) from the original geometry.

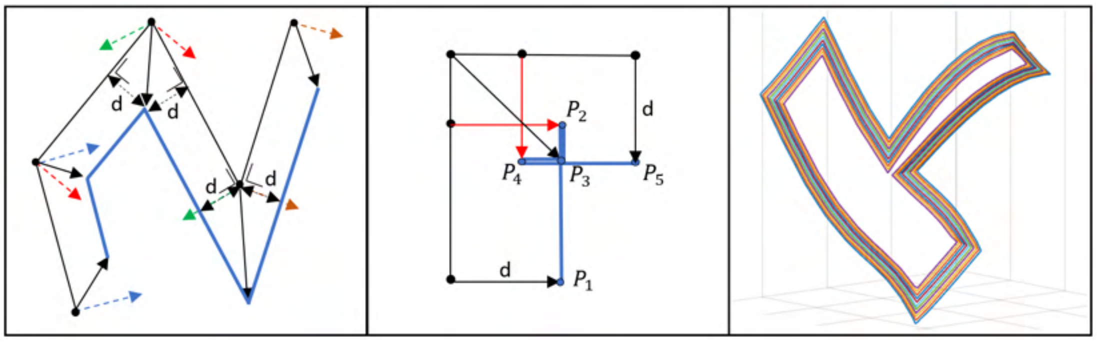


Figure 4.10: Definition of additional perimeters

The new intersecting surface is generated by the algorithm shown in Fig.4.10 and it works like this. The normal vector, $\overrightarrow{n}$, at each point of the contour is known as well as the tangential one, $\overrightarrow{t}$ (all the segments obtained retains information of the original triangle). A point $P_i^{(1)}$ is transformed in $P_i^{(2)}$ by the knowledge of two neighboring point (*i-1* and *i+1*). Solving the geometrical problem, it yields:

$$P_i^{(2)} = P_i^{(1)} + \frac{d}{\sin(\beta)} \frac{\overrightarrow{v}}{\|\overrightarrow{v}\|}$$

Where β and $\overrightarrow{v}$ are defined, having indicated with w the normal unitary vector to the $t - n$ plane, as:

$$\beta = 0.5 \times acos\left(\frac{\overrightarrow{t}_{i+1}}{\|\overrightarrow{t}_{i+1}\|} \cdot \frac{-\overrightarrow{t}_{i-1}}{\|\overrightarrow{t}_{i-1}\|}\right) \quad ; \quad \overrightarrow{v} = 0.5 \times (w_{i+1} + w_{i-1})$$

The points obtained require to be post processed to get rid of possible undercuts using self-intersection detection algorithms for polygons.
A new solid can be then created from the new path preserving the original normal vectors to get the exact profile (position and normal direction).
Results of the addition of extra perimeters for the contours presented previously are depicted in Fig.4.11, these are intended as explanatory of the process just described and they do not necessarily represent the one used to determine the actual toolpath.

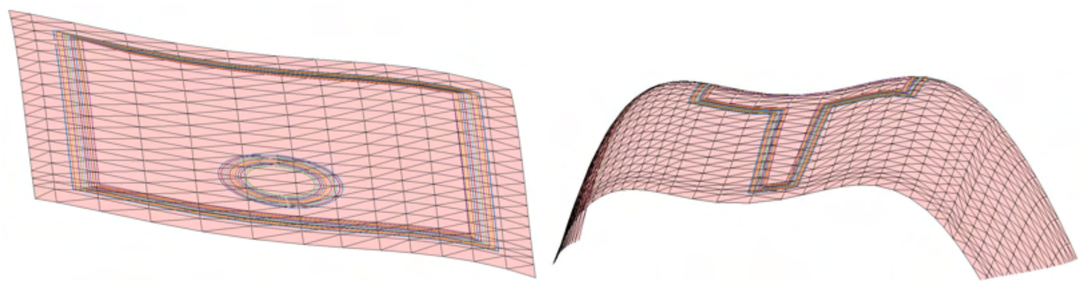


Figure 4.11: Additional perimeter examples for the examples in Fig.4.8 - 4.9

Hatch distance The previous algorithm is based on a local approximation of the curve of the surface (Taylor series truncated after the first derivative); therefore, the greater the distance d the greater the error. When dealing with perimeters, its exploitation is justified by their distance, since it is expected to be in the same order of magnitude of the nozzle diameter (to give birth to a unique wall). In general, when we're dealing with non-planar surfaces, searching a neighboring pass only relying on the information on the current position will lead to some error.
Let's consider the case in Fig.4.12,*A* as an example in $\mathbb{R}^2$. If it is moreover supposed that the first pass is oriented as if in the planar case, it is possible to witness how defining the second pass, only based on the knowledge of the first one, leads to errors.

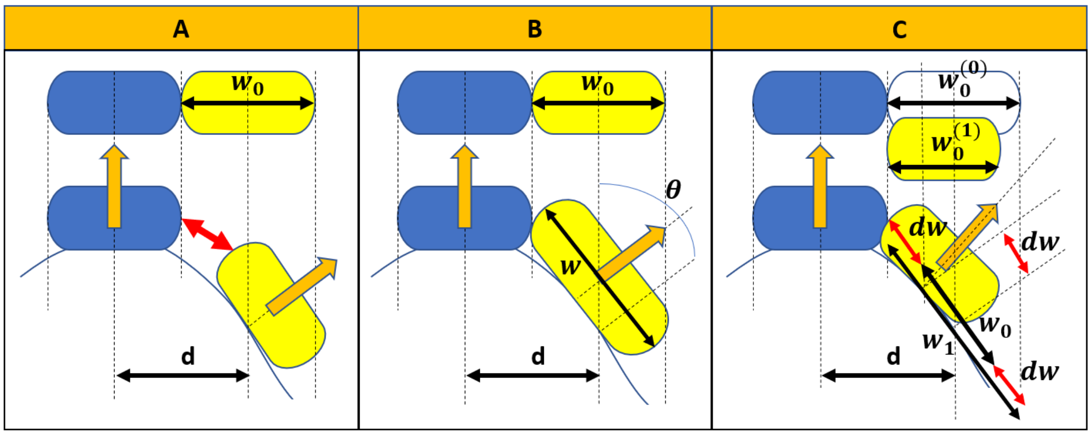


Figure 4.12: Definition of additional perimeters

In Fig.4.12,*A*, the second pass, intended to be juxtaposed to the first one, once brought on the actual curve, leaves an empty gap in contrast with what was desired. This shows that also additional perimeters need to be checked according to the nozzle diameter and surface topology.
Fig.4.12,*B*-*C* report possible countermeasures. The two cases differentiate on the width (the desired one is w_0) of the final pass. Knowing the new orientation of the tool with respect to the surface and its previous position, the angle θ can be computed and consequently w_1 that is the distance between the rounded sides of the starting cross section along the new tangential direction.
Now two possible solutions can be thought: reshape the section or shift it close to the original one. Reshaping can be done considering the relative bounding box that will have same height, but a length estimated to $w_1 - h\tan(\theta)$. In the introduction chapter, it has been reported how the width of a strand on the substrate has and should have a length that belongs between the nozzle diameter and its outer dimension. Hence, the new length has to be smaller than the outer nozzle diameter to be sure to press the material against the surface. Of course, not only the material flow rate has to be updated, but also the waypoint has to be updated, as in Fig.4.12,*B*.
On this matter, the second solution keeps the length equal to w_0, but shifts the waypoint using $dw = 0.5(w_1 - w_0)$. This is equivalent to a modification of the width before the evaluation of the position and orientation on the surface.

The two solutions still avoid information on the topology of the substrate to compute the position of the additional perimeter (or infill). The substrate can be analyzed at the beginning of time, only if it is known that it will not change throughout the build, to store the variation of the normal field according to the position. If the substrate is updated from one layer to the other, then a local study of the curve in the direction where the second wall will be placed has to be done every time.
This can be achieved by intersecting the surface with a rectangle having the base with a

length equal to the hatch distance, the height parallel to the normal vector of the point on the contour and the normal vector of the rectangle parallel to the tangential vector in the same point. The hull path has to be sampled with a suitable spatial interval (to avoid gaps or superimposition) to yield better control of the neighbor contour. The curve imprinted on the cutting rectangle can be used to determine the position of the next point.

4.5.2 Infill

The infill is generated following the same principles used for the secondary perimeters. If specific patterns are considered like honeycomb, they have to be generated on the substrate to preserve the distance as seen in the previous paragraph. Here the rectilinear infill is discussed.
The main steps to obtain the infill is here listed:

1. Plane spacing and orientation
2. Surface slicing
3. Curve reconstruction
4. Contour-infill intersection

Plane spacing and slicing is a direct implementation of what has been said and done for the contours and perimeters, but using a geometry made of planes instead (a practical example can be found in section 7.1, Fig.7.3). The curve reconstruction rearranges the segment belonging to the same curve and it also make sure that only m of the $2m$ points are considered.
The contour-infill intersection is the last step, but it is not strictly necessary if the planes have been generated using a mesh that approximates the geometry within the hull. An object might be formed in fact by several loops (outer and inner) that causes on a particular curve (belonging to the infill) to have segments where the material is deposited and other where is not. Therefore, the cutting plane not only has to intersect the surface but also the contours.
The points, obtained after this second intersection, are then sorted before distinguish-

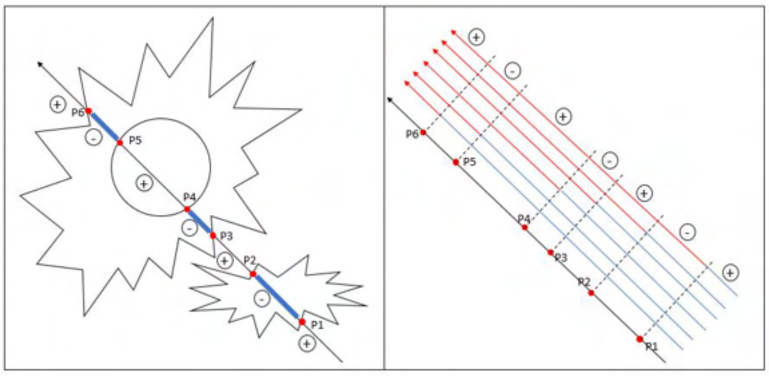


Figure 4.13: Print region detection criterion: *left*, example of generic infill with final result;*right*, visual implementation of region identification

ing if they require the extruder on or off. In Fig.4.13, it's possible to see a visual representation of the algorithm to obtain the printable region, that is:

1. calculate the distance of the infill points from the one intersected with the contour.
2. take the product of only the sign of these distances
    (a) if $\prod sign(d) > 0$: no print
    (b) if $\prod sign(d) < 0$: extrude

The infill results for the contours presented previously are depicted in Fig.4.14 for the sake of completeness.

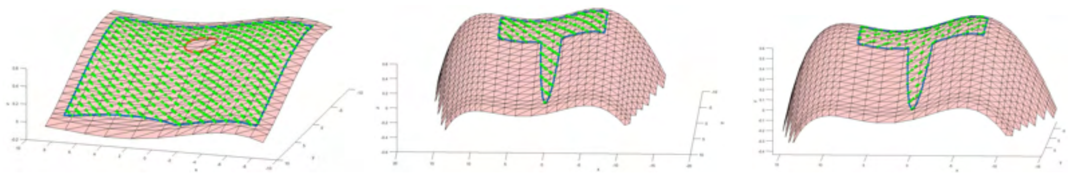


Figure 4.14: Infill result for the examples in Fig.4.8-4.9

4.6 Trajectory definition

After the slicing procedure, the path yielded is constituted by a series of points with the associated normal vector to the surface. This information has to be used to define the position of the End Effector, EE, throughout the process.
Being the EE a 3D body in space, its location is defined by the position and orientation of a reference frame attached to the body itself. Hence, the toolpath has to resemble the one in Fig.4.15 where the desired local coordinate system of the EE at each point is known.

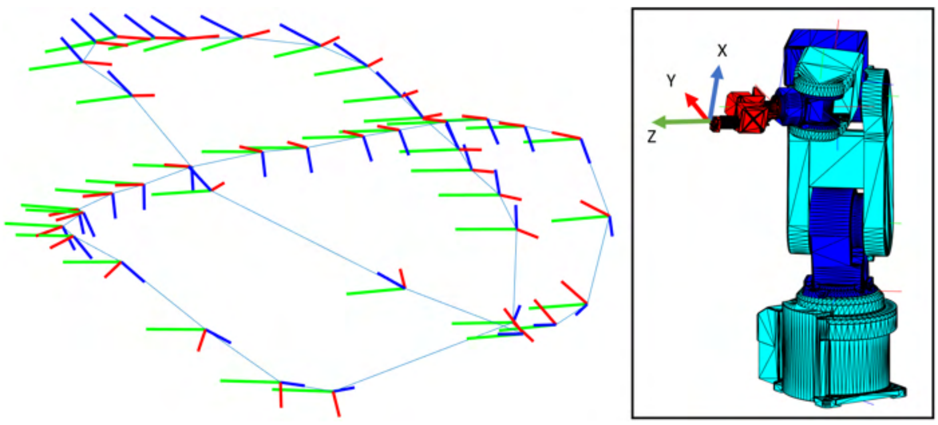


Figure 4.15: Example of a path with reference frames associated to each point that the EE will meet during its motion.

The information, extrapolated by the slicing software, is able to characterize the origin of such moving frame and one direction cosine. The convention is to assign the EE frame with the Z axis exiting the body, parallel to the direction of approach of the gripper (Denavit and Hartenberg's conventions). The X and Y unit vectors are unknown. Therefore, after the slicing procedure, for each point, the EE has an infinity of possible poses that it could take (the EE frame can rotate around its known Z-axis, hence infinite solutions). Of this infinity, only a subset is feasible.
The final frame describing the orientation has to be computed taking into account several constraints, the main ones are:

- **Range of motion**: apart from the fact that a waypoint has to be reachable, rotation around the tool axis might be limited in range for a given machines. So, paths that leads to continuous rotation of J6 through consecutive points have to be checked. The final path has also to be free from sudden changes of the solution adopted. These points lead to movement at high speeds and accelerations to reconfigure the manipulator that prevent the system from completing the task.
- **Collision**: no overlapping has to occur among the bodies in the cell, particularly in this application in which hot surfaces might damage the robot, the substrate and electrical wires.
- **Tool cabling**: tool wires can limit the range of motion of the robot either due to their pulling action (thus increasing the payload and reducing the accuracy) or due to their impossibility to get excessively twisted.

An EE frame can be tentatively assigned to each point and later tested for collision and the infringement of the range of motion. Simulations are therefore used to run the job to check and eventually get the information needed to fix possible issues.

4.6.1 *Fixed* or *Mobile* tool cell

The type of cell has to be known before proceeding because it shapes the steps to take (in Fig.4.16 the original path refereed to the substrate with the resulting kinematic loop closures for the two types of cell is reported to clarify the concepts). In case of the extruder mounted on the robot, the normal vector is substituted by the one going in the opposite direction, based on the convention that wants the tool Z-axis to exit the wrist (thus, entering the substrate). The path can be left on the workpiece, leaving the controller to compute it at the EE using the definition of the tool geometry.
For the fixed tool cell, the norm is unchanged for the computation of the rotation matrix (because it exits the substrate that coincides with the EE), but it's required to define the points as the path of the *tool 0* (wrist of the robot). This is done to transfer the solution to robots with controller not able to process this kind of trajectories. The points, that the *too 0* has to follow, are obtained as:

$$P_{i,tool0} = P_{fixed} - R_i^T P_i$$

where P_i is the position of a point on the toolpath accounting also for the dimension of the substrate. R_i is the rotation matrix that takes the tool coordinate system to the one along the path.

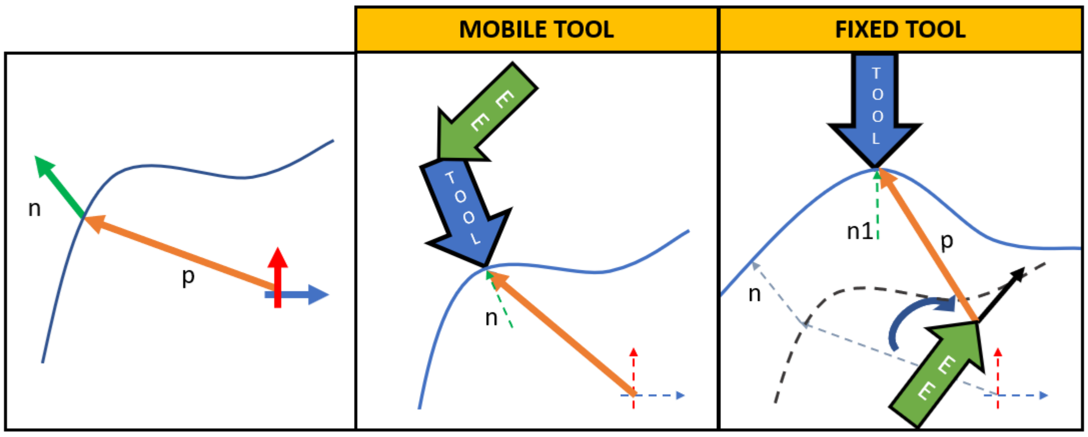


Figure 4.16: *Fixed* or *Mobile* tool cell path update.

4.6.2 Orientation assignment

Given the type of cell, the tentative frame can be assigned to each point. Three possible methods have been used during this research activity: *tangential vector*, *world director cosine alignment* and *Rodrigues' rotation formula.*

Tangential vector alignment: using the knowledge of neighboring points, the tangent vector, $\boldsymbol{t}$, can be approximated. Imposing a director cosine (x or y unit vector) to be aligned with the tangent, the rotation matrix can be computed. This can cause multiple revolution of the wrist.
For example, if the tangent vector is aligned with the x unit vector, the following equation can be written:

$$\underline{Y} = \frac{\boldsymbol{t} \times \boldsymbol{n}}{\|\boldsymbol{t} \times \boldsymbol{n}\|} \qquad \underline{X} = \frac{\underline{Y} \times \boldsymbol{n}}{\|\underline{Y} \times \boldsymbol{n}\|} \qquad R = [\underline{X}, \underline{Y}, n]$$

where normalization is used to make sure to obtain an orthogonal matrix with unitary determinant.

World director cosine alignment: the reference frame on the path can be engineered to keep one axis always parallel to a director cosine of the world reference frame (at least the closest solution with a proper tern). This limits the range of motion on J6. The computation can be done analogously as for the tangential vector alignment case.

Rodrigues' rotation formula: A third solution uses Rodrigues' rotation formula to compute the rotation matrix to align the normal vector to the Z axis of the tool as the result of an axis-angle rotation. Considering an initial vector $\boldsymbol{n}$ in $\Re^3$ and a unitary vector $\boldsymbol{w}$ of the rotation axis, around which $\boldsymbol{n}$ rotates by an angle θ, the rotation matrix R, that gives $\mathbf{v_{rot}} = \mathbf{Rn}$, can be computed as:

$$\mathbf{R} = \mathbf{I} + (\sin\theta)\mathbf{K} + (1 - \cos\theta)\mathbf{K}^2 \tag{4.1}$$

In the case that n and v_{rot} are known, as the one here discussed, the term $\sin(\theta)\mathbf{K}$ of the (4.1) can be determined has:

$$\sin(\theta)\mathbf{K} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}$$

where the elements of the matrix are the components of the k vector, computed as: $k = \mathbf{n} \times \mathbf{v_{rot}}$. The remaining term to be computed is the $\cos(\theta)$. This is equal to the dot product between the two vectors after being normalized. The rotation matrix is computed substituting the calculated terms in (4.1).

These methods are depicted in Fig.4.17 with some practical examples either taken from the *mobile tool* or *fixed tool* setup.

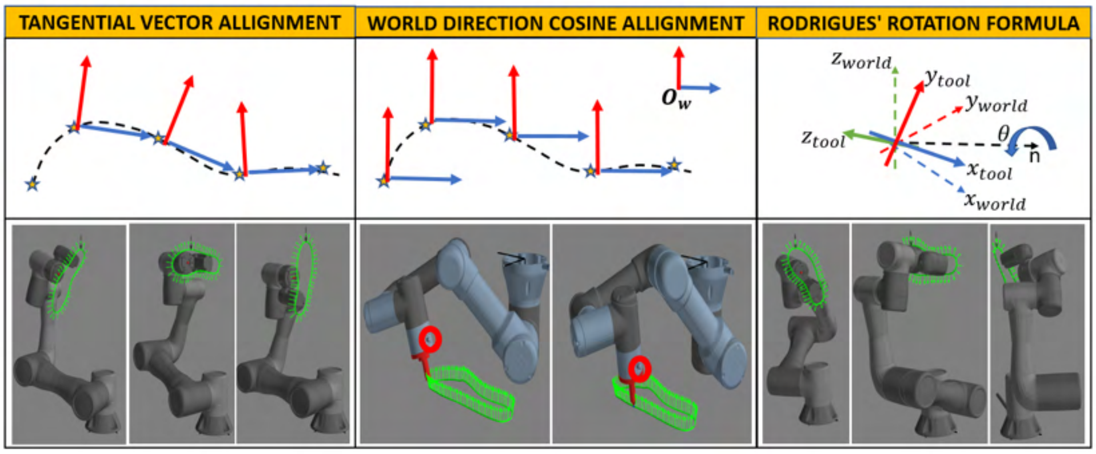

TANGENTIAL VECTOR ALLIGNMENT
WORLD DIRECTION COSINE ALLIGNMENT
RODRIGUES' ROTATION FORMULA
O_w
z_{world}
y_{tool}
y_{world}
z_{tool}
θ
n
x_{tool}
x_{world}

Figure 4.17: Examples of initial tern assignment along the path: *Tangential vector alignment* in 2D space points for example the x unit vector to the next point; *World director cosine alignment* keeps it always parallel to the x direction of the world reference frame; *Rodrigues' rotation* in 3D space.

Motion law assignment

It is now possible to use this information (geometrical path, segment with or without extrusion, cross-section of the pass and so forth) to assign a motion law to the robot and

extruder. The common practice sees the definition of the TCP feed with the extrusion speed calculated accordingly (using the equation (2.1) in subsection 2.1.1).
The algorithm used to generate the motion laws is a *Look Ahead* with trapezoidal speed profile that tries to guarantee the commanded feed along the path, provided that is feasible, and compliant with the extrusion process and reorientation of the tool occurring at the same time. Of course, when dealing with existing controllers, time dependent checks (for instance, collision between moving machines) requires the exact replica of the interpolator to foresee the behavior of the cell.
Prior to the toolpath checks, a coarse time discretization allows to reduce the computation time required to perform them. Only after the response is positive, the actual cycle time can be used to interpolate the points (only for the real time control case section 3.2, in the other cases the feed is sufficient).

4.6.3 Trajectory feasibility check

In the previous section, it has been discussed that to confirm the trajectory generated checks have to be performed on range of motion and collision.

Range of motion check

To check if a point is reachable (or at a singularity), the inverse kinematic (IK) problems has to be solved to find all the possible solutions. The IK algorithms have been developed using [47] and [98] as reference for the machines adopted (section 3.1). Once obtained the joint-space coordinates for all the 8 (2 in the case of the SCARA) possible configurations (or solutions to the inverse kinematic problem), the ones that have points unreachable (i.e. coordinates degenerate to a Not A Number) can be entirely or partially discarded. An unreachable point with one configuration, can be obtained changing the configuration itself (see 4.18); this can imply fast movement of the joints with a small TCP motion that is not compliant with the ISO standards. Therefore, if all the solutions have at least one unfeasible point, configuration change and blending is advised and used (split the reconfiguration on multiple points both ahead and after the one not feasible to reduce the maximum speed); otherwise if solutions that do not present unfeasible points exist alongside those that do, then the latter can be discarded altogether.

4.6.4 Collision check

The configurations that survived the first cut, before being sanctioned, need to pass the self and collision check (in the appendix B the theory on collision is mentioned). This test can be divided in two types according to the number of moving machines in the cell. Single-machine cells only require carrying out self-collision (if not already accounted in the range of motion check), mainly between the tool and the rest of the moving parts and the collision of these parts with other bodies in the cell.

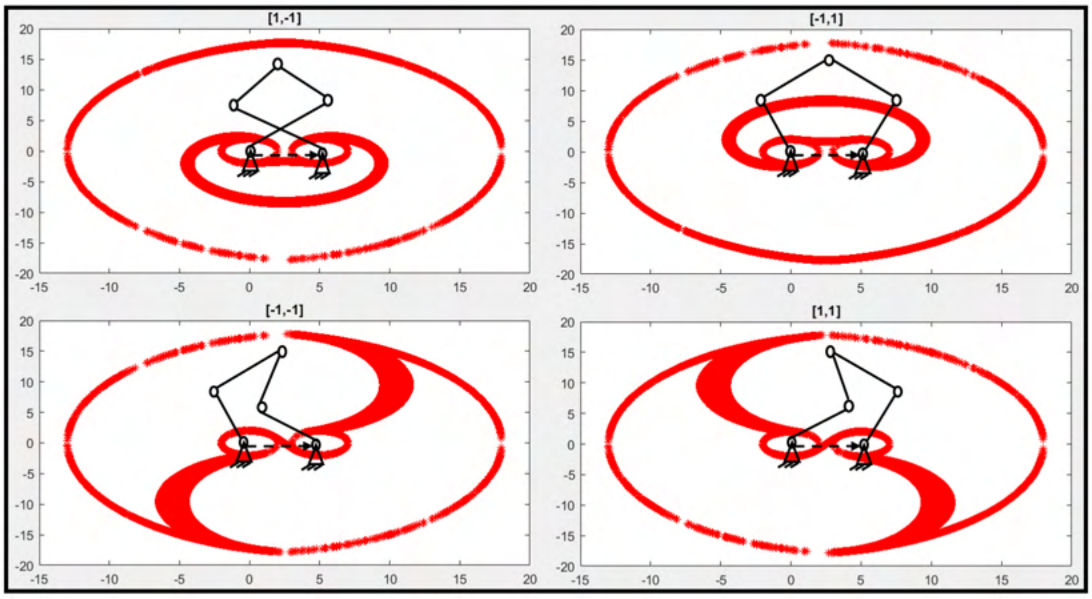


Figure 4.18: Examples of the singularity points for a planar PKM 5R robot (double SCARA) with the left leg, Ll, kinematic parameters equal to [10; 8] mm, while the right, Lr, [10; 8]; the coordinate system is placed in the left actuated joint and the second actuated joint is placed at E = [5; 0]. The unreachable points are obtained mapping the operational space for each of the four possible configurations (the initial poses are depicted for the sake of clearness) and computing the ratio of the eigenvalues of the matrix $J^T J$ for each point to put aside those that goes to 0 or ∞ (that represent the boundaries of the metric space identified with $J^T J$). The matrix J is computed as $J = \begin{bmatrix} -K_\alpha \sin(\mu_\alpha) & -K_\beta \sin(\mu_\beta) \\ K_\alpha \cos(\mu_\alpha) & K_\beta \cos(\mu_\beta) \end{bmatrix}$, where $\mu_\alpha = \alpha + \theta_L$, $\mu_\beta = \beta + \theta_R$, $k_\alpha = \frac{Ll_1 \sin(-\theta_L)}{sin(\mu_\beta - \mu_\alpha)}$, $k_\beta = \frac{Lr_1 \sin(-\theta_R)}{sin(\mu_\alpha - \mu_\beta)}$ and θ_L is the angle between the second and first link of the left leg (θ_R of the right leg).

Multi-machine cells in addition to their single-machine tests have to cope with the presence of other robots sharing the workspace. Fig.4.19 depicts these two families.

Behind these categories, there is a different type of collision that echos back on how to avoid them. Self and collision with fixed parts (*Single-machine*) is independent of the simulation (and in the physical world of the job) time. In fact, this type of collision depends only on the positions of the bodies and will not change with a different time stamp. Collision among moving machines is time dependent, so it could either be fixed by changing the path or the time stamp.
Let's see how the tests are done.
The first step is to manage the description of the cell. The geometries of the bodies are reduced to a minimum form, where possible, in order to reduce the computational cost. As can be seen in Fig.4.19 links can be approximated with boxes, not to be confused

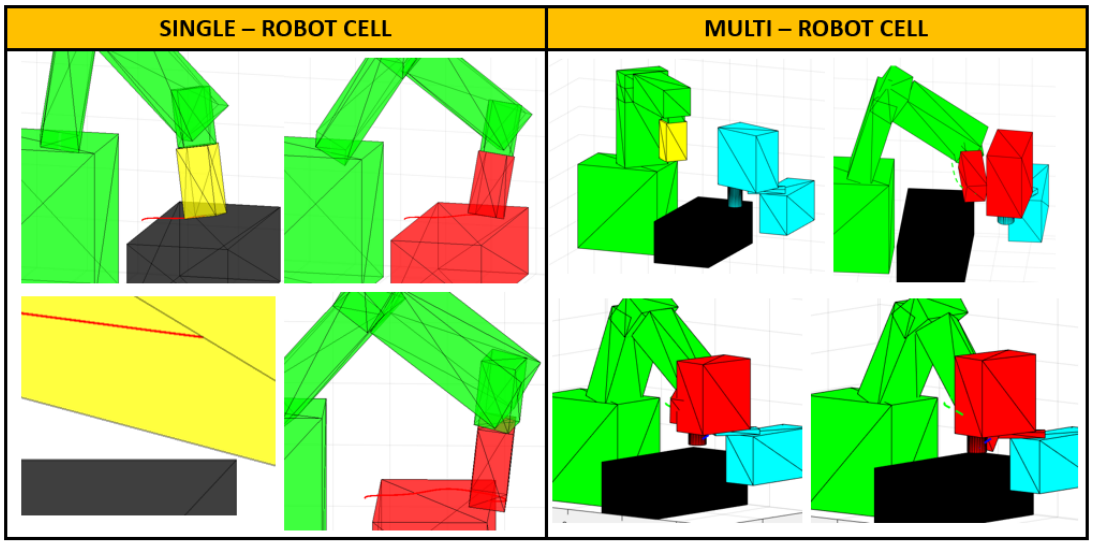


Figure 4.19: *Single-machine* and *Multi-machine* cells collision examples.

with the actual bounding box substitution used to preliminary evaluate if the poses lead to overlapping. In the same image, the mobile extruder is also approximated as a box, but this is an oversimplification that can cause false positives. In this case, a sensible simplification is to split the extruder in two subparts one relative to the motor plus flange-heat sink and a second relative to the heatblock and nozzle.
The algorithm is then identical to the one used to get the toolpath with the difference that the exact intersection point is not calculated. What is returned is the Boolean relative to collision and eventually the index of the bodies overlapping (the bodies to check are extrapolated from a standard collision matrix defined by the user in which the index of the rows and columns represent a specific body while the elements are Booleans indicating if they have to checked or not). Knowing the bodies in a faulty position, it's possible to tell if they are dependent or not on time.
Time invariant collision are impossible to be overcome while keeping unchanged the toolpath engineered. The first decision, the algorithm takes, is to discard these configurations. If all are eliminated, then it checks if there are points that causes the machine to always crush; these points have to be redesigned. All the other paths can be rearranged stacking portions of curve with different configuration (as did for the previous case when discussing the range of motion).
Collision, that can be prevented assigning another timestamp, allows to anticipate, delay or reorganize one of the two moving objects, if the final result is still compliant with the task processed. For movement such as *point to point*, characterized by the importance of the start and end position over the journey, the path can be significantly altered to avoid the impact. Fig.4.20 is an example of four SCARAs sharing the same workspace where two of them crash. The one on the top-left is following a linear interpolated motion, while the one in the bottom-left is unconstrained. Information returned from the forward pass is used to automatically add waypoints that assure a

safe run and a correct task completion.

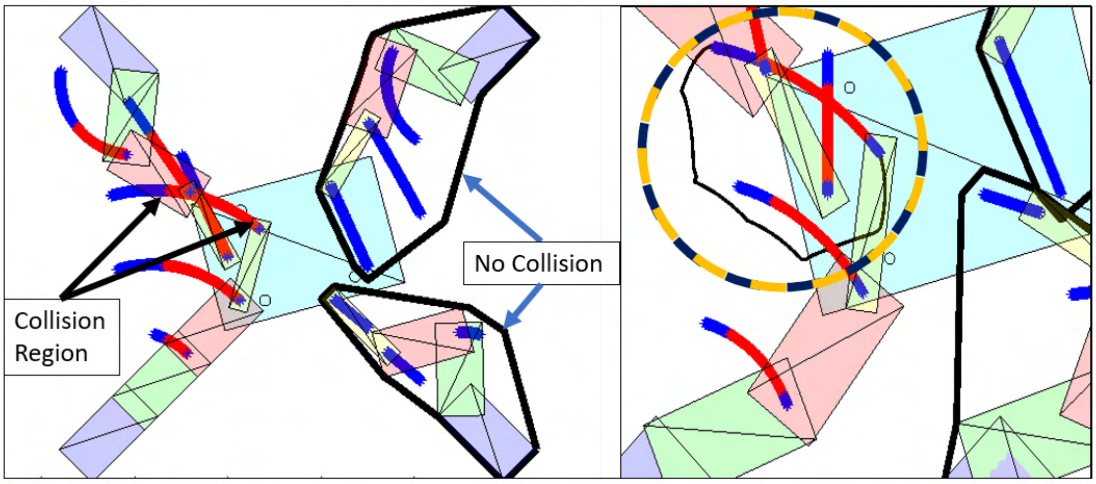


Figure 4.20: Example of a 4 SCARA cell sharing the same operational space (left) where collision check is in indispensable tool to make sure to plan collision free trajectories (right).

4.7 Machine instructions

Until now the step starting from the raw digital model to the final toolpath has been discussed. The information in this format needs to be processed to the specific machine. As seen in section 3.2, the control architectures implemented demands the trajectory to be prepared in a particular format and delivered with a certain timing.
Stand-alone can either be adopted generating a G-code file that is then interpreted by the controller or directly the machine instruction in the native programming language of the robot (c++ or python based). Execution is carried out by the interpolating functions of the robot.
Data Link must receive a package that is string with the target positions and speed (the message is user defined); extrusion can be controlled on a parallel TCP/IP port. Synchronization is achieved by parallel tasks in the controller to separate position update from the actual motion.
Real Time resembles the *Data Link* case but with a stricter cadence and data massage (byte format of fixed length).

4.8 Future entries

Towards the realization of the version 1.0 of the software, the built functions need to be improved and additional ones have to be inquired and implemented.
The possibility to directly generate the CAD in the software will allow to track on the idea of how the part is realized by the technician. This will serve as starting point to

automatically and easily discern between AM or SM.
The *RSS* should also be able to directly detect errors in the STL file and provide to the relative repairs.
Part orientation within the cell has to be improved and generalized. A possible mean to achieve it is to use Machine Learning (for instance a Deep Convolutional Neural Networks).
A database with the most widespread robots' configurations (kinematics and controls) will attract a larger interest. Moreover, the possibility to add a new model or a custom machine will make the *RSS* retain its origin in research activities and propel innovation in the machine for AM.
The software, lastly, has to cope with additional tools inside the cell (such as grippers, spindles, drillers, human operators) process and control-wise.

4.9 Build phase

The machine instructions can be uploaded or sent to the cell to obtain the physical object. Examples can be found in the next three chapters. The first one revolves around simpler case studies used to validate and test the part of the algorithm here discussed; the second when introduces novel building strategies (chapter 5). The third chapter reports actual industrial project where the algorithm has been adopted for subtractive manufacturing and 3D multi-directional printing (chapter 7).

Chapter 5

Experimental tests

Once established the foundation of the activity (tool, machine, software), experimental tests can now be performed on the test rig.
The matter is discussed starting with some examples using conventional printing involving setup checks and large areas print experimentation (section 5.1). Being this the standard building philosophy, it is mandatory for any new system to be at least equal to it and not a regression to early stages of the technology. Therefore, some geometrical tests are conducted.
Multi-directional printing is then the subject of section 5.2. Here, three basics building strategies (considered because they admit a well-known algebraic formulation) are highlighted that leads to prints on a gable roof and on a cylinder. The latter is performed in a dual manner, either by keeping the substrate constant throughout the process or adjusting its curvature to transition to a different layer type (1.3.1).
On this trail, a special declination of the multi-directional that allows to print rods and tubes without the limitation of the standard layer building approach (section 5.3). This method (indicated with lathe-like printing) is the first case in which the slicer shown in the previous chapter is deployed to get the toolpath. This choice has been taken to separate the technological issues with the ones that could have possible risen from the newly coded slicer.
All cases presented here differ from the one of the following chapters (7) for they geometrical simplicity compered to actual industrial case studies.

5.1 Conventional printing

Conventional printing (2.1.1) is a viable tool to check the final setup without introducing additional process variables. Two sets of tests have been carried out, one to inquire the variation of dimensional accuracy as a function of the increasing length when using industrial robots and one to menage prints using builtin functions in preparation for unconventional printing.
The *Epson T3* (see chapter 3) have been chosen for this phase since it is a SCARA,

hence it has the least number of degrees of freedom (4 Dofs) among the one considered required to define the position in space of the tool (3 Dofs are required for the position in space of a point).

5.1.1 Dimensional test

Once completed the setup, i.e. the integration of the extruder with the robot, it is required to evaluate the optimal process parameter extensively pointed out throughout the discussion (chapter 1 and chapter 2). After that, the effect of the manipulator on the final build can be analyzed.
Robotic systems are mainly intended to be programmed in manual mode; so, an operator teaches the target positions by physically bringing the TCP to that positions. This approach, therefore, hides some accuracy issues resulting either from the control (such as kinematic parameters, encoder resolution, position control) or the physical system. From literature, this is resolved by calibrating the machine.
These initials test also allow to assess if a calibration process (it is advised to opt for an off-line compensation of the reference positions rather than tamper with the controller parameters) is in order.

Varying length The first test aims to study the effects of an increase in length of the object to print (industrial robots tend to have a larger print area than Cartesian available on the market).
A geometry like the one in Fig.5.1 has been adopted; it has two holes, a circle with constant radius (d1, 10[mm]) and a rectangular one with fixed width (W1, 5[mm]) and variable length (L2). The outer perimeter is a rectangle with fixed width (W, 20[mm]), variable length (L) and one rounded vertex. The height of the object, H, is 5[mm].
The distribution of L is [50,100,210,260,310,360] privileging dimension over 200[mm], but less than the maximum print length of 400[mm] along the considered axis; L2 distribution is [10,60,160,210,260,310] [mm] instead.

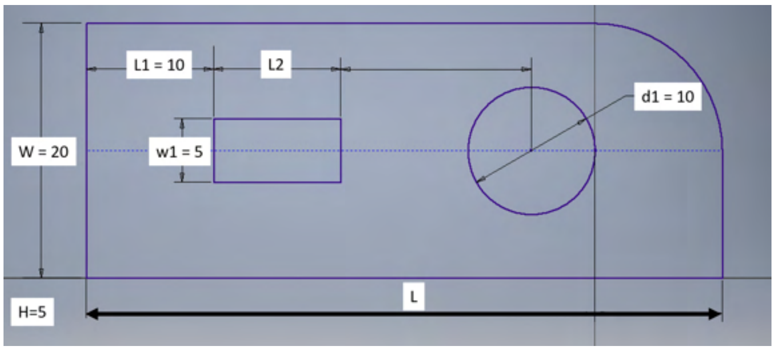


Figure 5.1: Dimensional accuracy: sample geometry.

The CAD is sliced by means of *Slic3r* to avoid any corruption of the trajectories due to the newly implemented slicing software. The position and orientation of the part inside the operational space is unchanged throughout the campaign. Moreover, the print parameters are kept constant from one sample to the other; as concerns the material, PLA is used starting from 1.75[mm] filament and extruded at 220[°C] through a nozzle diameter, d_n, of 0.4 [mm]. The print speed is 13 [mm/s]. Prints are done on a unheated bed, using glue to promote adhesion of the first layer.

The specimens are preliminary measured using a caliper (dimension less than 200[mm]) and a meter. In addition to geometrical parameters, the eventual presence of warping and other deformation has been recorded.
The maximum errors (in absolute values), in millimeters, are [0.25,0.15,0.25,0.40] respectively for W, W1, L1 and d1, 0.30 for H and [0.80,0.70] for L, L2. The average values are respectively [0.10,0.10,0.0667,0.25], 0.15, [0.4083,0.1667]. These errors are within $\pm 2d_n$, that could be due to robot or the slicing procedure (strands width is usually within 1÷2 times the nozzle diameter). Percent errors (expressed as $\frac{X_{ref}-X}{X_{ref}}$) do not show any significant expanding trend (plotted in Fig.5.2) related to the increase in L; values are bounded between ±4% with the exception of the height.

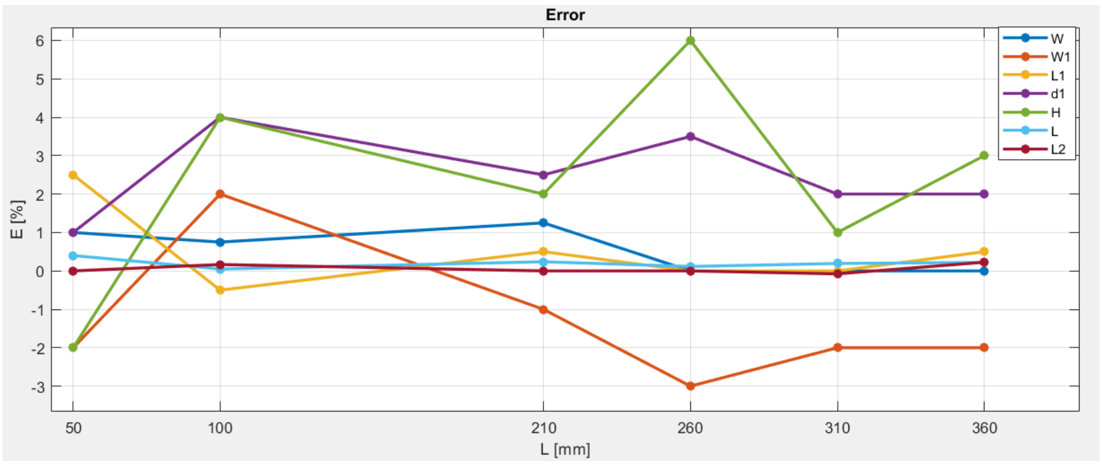


Figure 5.2: Test on dimensional accuracy: percent errors.

No warping has been found during the prints (that toked from 41 (shortest) to 217 minutes (largest)). The estimation of other deformation, in the XY plane have to be carried out using suitable measuring devices (non-contact like by means of cameras). L and $L2$ do tend to increase, but with the measuring devices at disposal for the test it is difficult to assert if it related to the system or to the measuring device resolution. Final parts are reported in Fig.5.3 that will require better measuring devices (and one for all the specimens) to extract exhaustive conclusions. Preliminary, it can be asserted that being the errors within the $\pm 2d_n$ the cell created does not constitute a significant regression with respect to current machines.

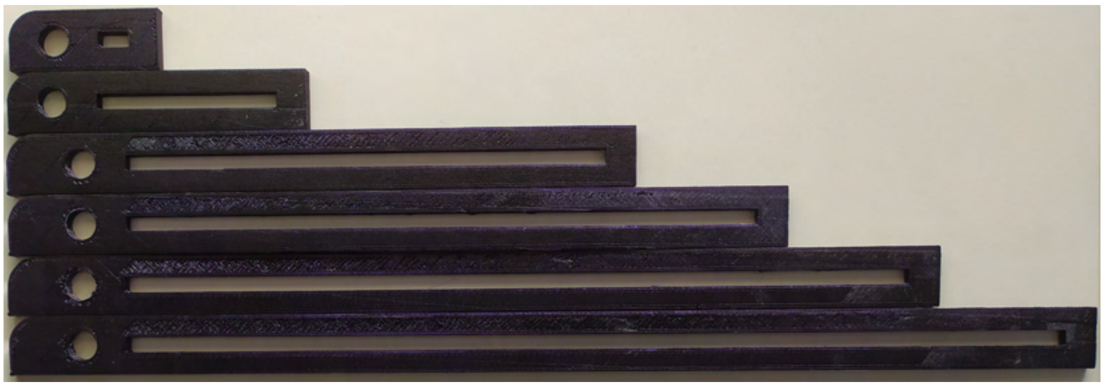


Figure 5.3: Test on dimensional accuracy: final specimens.

5.1.2 Additional test

Here two additional trails performed on this setup are reported.
The first tends to exploit all the printing area as can be seen in Fig.5.4, where the SCARA is printing a circular crown. Not significant variations from the previous samples have been found. The machine was able to work for the required time without hiccups (hence reinforcing its reliability).

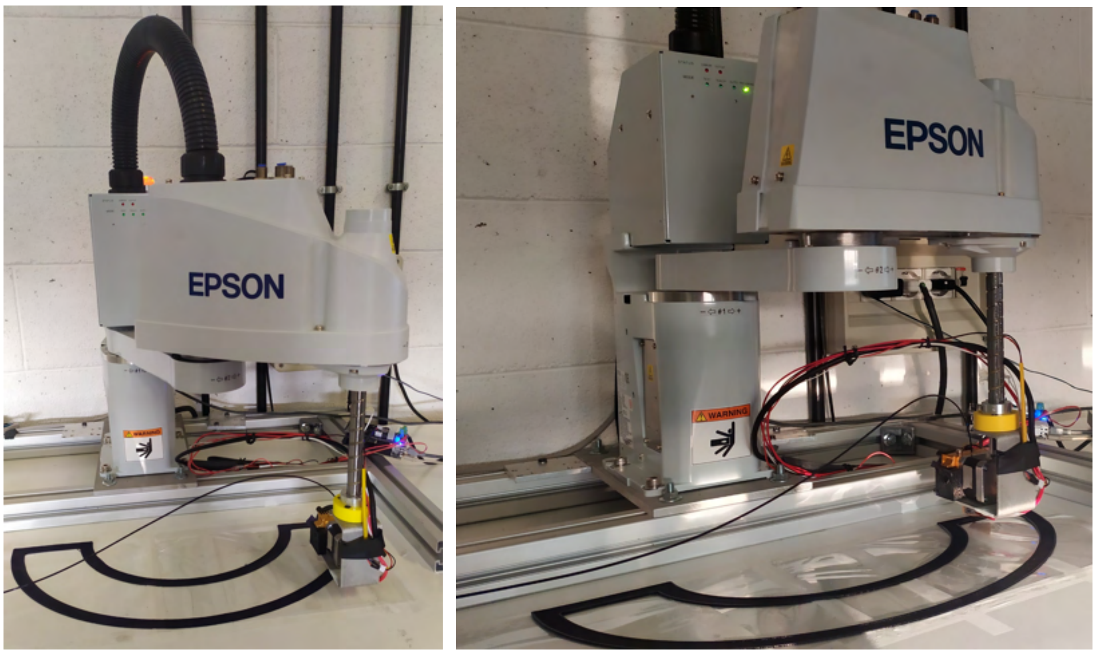


Figure 5.4: Robotic large print bed example (SCARA)

A second build has been done to integrate the change of the local reference during the task. This methodology allows to implement, in few steps, part decomposition and trajectory planning for planar-layer multi-direction printing. Indeed, by just defining a set of local coordinate systems to each layer change, the trajectories yielded by the

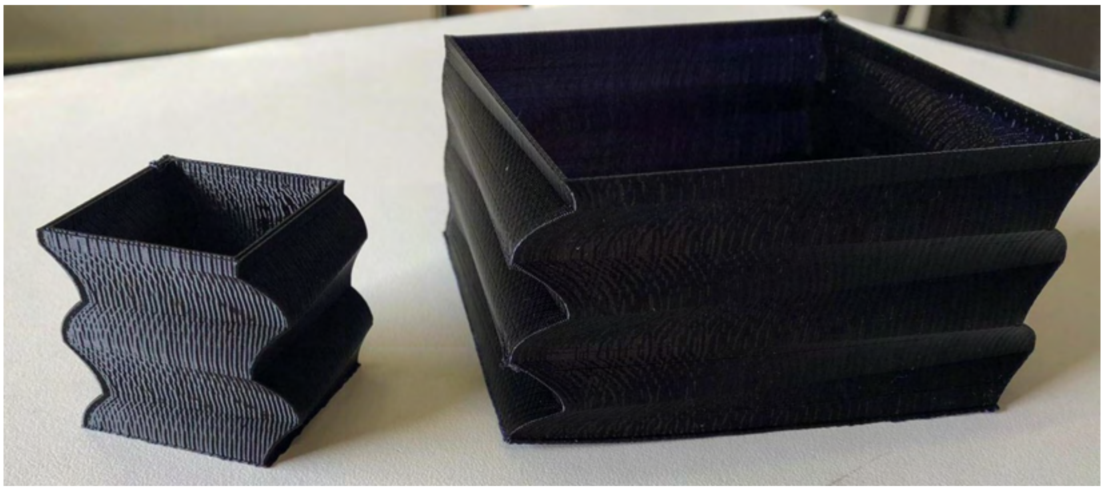


Figure 5.5: Print test updating the referencing system of the layer.

slicer can be straightforwardly uploaded on the controller.
Fig.5.5 depicts an example of parts obtained by changing the orientation about the Z axis of the previous layer, clockwise for 5 [mm] and then counterclockwise for other 5 [mm] of a step angle that guarantees no overhanging features.

5.2 Multi-directional: Custom trajectory

Abandoning the conventional building philosophy, it is required to use 6 Dofs machine to be able to locate the EE into the engineered position. The case studies in this section have been faced and implemented using the Mitsubishi manipulator (see chapter 3). The toolpath is obtained by generating customized curves using information available on the substrate considered (that admits a trivial mathematical representation).
Three cases will be addressed in this section, the first case study is a reinforcement rib on two inclined planes resembling a gable roof. The following two uses a half-cylinder as building surfaces, with the first replicating the same rib while a second a pentagon characterized by variable thickness layer calculated to obtain a flatter surface.

Fig.5.6 depicts these three cases. All of them are obtained in two distinct time instants (separated even by weeks); the substrates are indeed printed ahead in one batch (following the "layer by layer" philosophy with a white PLA filament) and only when needed used for testing purposes (in order to show better results, the new builds are printed with a black filament).
The aim of such trials is to validate not only the means to use these printing philosophies, but also to inquire the possibility to use FDM to customize and repair existing parts. The additional features can indeed be seen as logos to brand parts, customize reinforcements and fixes on damaged goods.

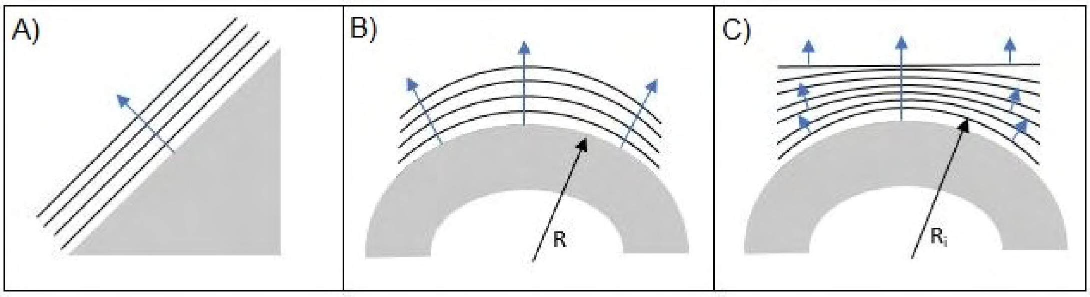


Figure 5.6: *A*, fixed orientation; *B*, variable orientation, fixed curvature radius;*C*, variable orientation, variable curvature radius

5.2.1 Gable roof

The building strategy in Fig.5.6-*A* can be tested by building a rib on top of a triangular prism. This is a straightforward case belonging to the planar multi-directional printing (1.3.1) constituted by two inclined flat surfaces meeting in the top.
The new feature is done by starting on the lower side of one of the slopes and moving towards the tip where the tool turns to lay down the material on the other side. The path, one arrived at the bottom shifts sideways to start a neighboring pass. Fig.5.7 shows the gable roof during the realization process.

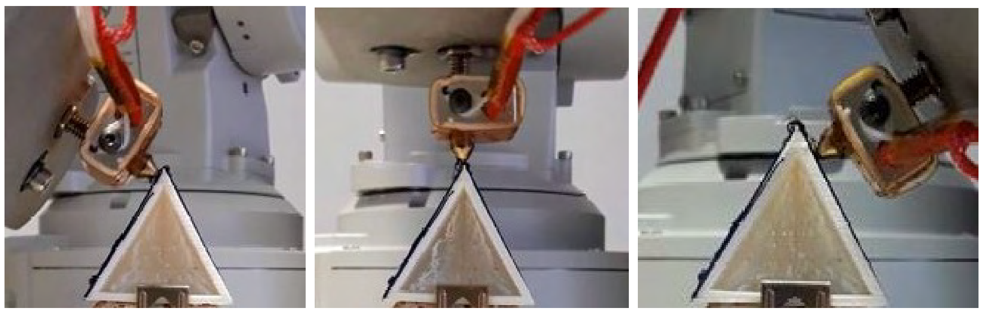


Figure 5.7: Double rib: turn execution

Table 5.1: Custom trajectory: rib parameters

$\#w_{lines}$	10	d_f	1.75[mm]	L_{base}	40 [mm]	h_z	0.2[mm]
$\#h_{lines}$	10	h_0	0.25[mm]	d_n	0.35[mm]	w	$2d_n$

Due to it triviality, the mathematical expression of the toolpath is omitted although the parameters used can be found in table 5.1. $\#h_{lines}$ is the number of layer, h_z corresponds to the layer height, L_{base} is the length of the base of the equilateral triangle and $\#w_{lines}$ is the number of parallel passes for each layer (the other parameters have been already introduced).

From this trial some important aspects have been witnessed. The deposition of the first layer requires to have an approximation of the substrate as close as possible to the actual one (the layer can function as a leveler for small imperfections); adhesion of the first layer is not as challenging as the one on a different material. The filament retraction is crucial to yield good results at the top, where to perform the reorientation, the robot has to slow down, hence spending more time in the same spot, to avoid exceeding the limitations imposed by the safety measures (i.e. fast movement with TCP in the same position).

5.2.2 Rib on cylinder

Analogously, a reinforcement rib is printed on the half-cylinder defined by the parameters in table 5.1, where L_{base} is this time the diameter of the circle.
The toolpath on the actual substrate (since it might different from the theoretical one due to the layer approximation during its built) can be designed (or taught by physically bringing the TCP to the surface) following these remarks (see appendix A for additional information about printing on cylinders):
Given 3 points (y_i, z_i), it's possible to define a circle with center in (a,b) in the y-z plane:

$$(y-a)^2 + (z-b)^2 = R^2 \tag{5.1}$$

By developing (5.1) and substituting (y_i, z_i), a system is obtained:

$$\begin{bmatrix} y_1 & z_1 & 1 \\ y_2 & z_2 & 1 \\ y_3 & z_3 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = - \begin{bmatrix} z_1^2 + y_1^2 \\ z_2^2 + y_2^2 \\ z_3^2 + y_3^2 \end{bmatrix} \tag{5.2}$$

Where $A = -2a$; $B = -2b$; $C = a^2 + b^2 - R^2$. Solving this system leads to the definitions of a, b, R. By solving (5.1) as a function of y, the following equation can be found:

$$z(y) = b \pm \sqrt{R^2 - (y-a)^2} \tag{5.3}$$

where only the equation with the plus sign is acceptable.
The points considered (either deriving from a priori or a posteriori knowledge) should be uniformly selected along the path, since (5.3) is a function of y, Chebyshev nodes can be used.
Defined n as the number of points, $k = \frac{\pi}{2(n+1)}$ and $\chi = \cos{(2j-1)k}$ for j=1...n-1, then:

$$y = \frac{y_{first} + y_{last}}{2} + \frac{y_{first} - y_{last}}{2}\chi \tag{5.4}$$

The new layer is easily determined by swelling the points in the y-z planes considering the same circle, but with a radius increased by the layer height, or by adding to the points a vector characterized by the modulus equal to h_z and directed as the unit normal vector (V_n), that in compact form is $\overrightarrow{P}_{new} = \overrightarrow{P}_{old} + h_z\overrightarrow{V}_n$. For the x-y plane the points are mirrored and shift according to the hatch space defined.

In Fig.5.8, the robot is depicted while printing on the rib. The solution implemented is an approximation of the curve (using chords instead of arcs), this leads to a polygonal chain that within each segment is closer to the substrate allowing for an increase of the adherence (provided that the maximum deviation from the actual curve is less then layer height).

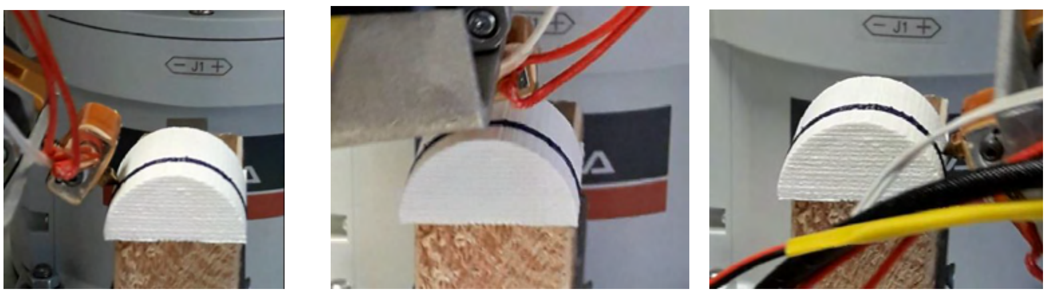


Figure 5.8: Custom trajectory on half-cylinder: path execution

5.2.3 Multi-directional: Variable curvature printing

The third sample (Fig.5.6-C) differentiates from the previous one because it starts the printing process on a surface with a given curvature to end up with a desired one.
A hollow pentagon, inscribed in a 30 [mm] diameter circle, is converted into machine instructions using a slicer and a post-processing algorithm to generate the reference commands. For simplicity' sake, the pentagon is modeled for conventional printing using a CAD software, sliced with *slic3r* (with a 20% infill) and then bend on the surface (see appendix A, where the method is detailed).
It is possible to end the new feature with a flat surface by deriving the updated equations for each layer given the previous radius (r) and the desired heights both at the top (h_1) and at the edges (h_2). Graphically speaking, the solution can be found in Fig.5.9.

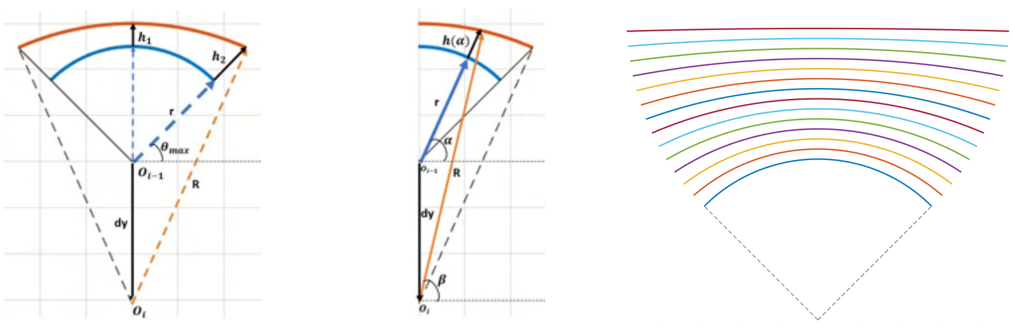


Figure 5.9: Graphical depiction of the half-cylinder curvature update with explanatory results.

This is translated into mathematical formulation as:

$$\begin{cases} R = r + h_1 + dy \\ R^2 = (r + h_2)^2 cos(\theta_{max})^2 + [(r + h_2) sin(\theta_{max}) + dy]^2 \end{cases}$$

Solving this two-equation system in R (the new radius) and dy (the vertical motion of the center), we get:

$$\begin{cases} dy = \frac{h_2^2 + 2r(h_2 - h_1) - h_1^2}{2(r+h_1) - 2(r+h_2) sin(\theta_{max})} \\ R = r + h_1 + dy \end{cases}$$

To adjust the material flow rate during the manufacturing of the layer, the height as a function of the angular position, (α in Fig.5.9), needs to be evaluated, that is:

$$\begin{cases} k_1 = r + dy \sin(\alpha) \\ k_2 = dy^2 + r^2 + 2rdy \sin(\alpha) - R^2 \\ h(\alpha) = -k_1 + \sqrt{k_1^2 - k_2} \end{cases}$$

The relationship between α and β is: $\alpha = atan2(R\sin(\beta) - dy, R\cos(\beta))$.
Result can be seen in Fig.5.10 in which the print was stop prior to meet the flat condition, but still manages to remark the concept.

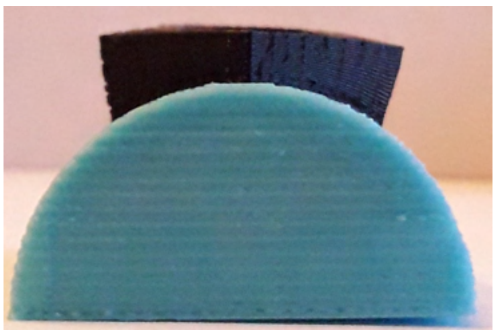
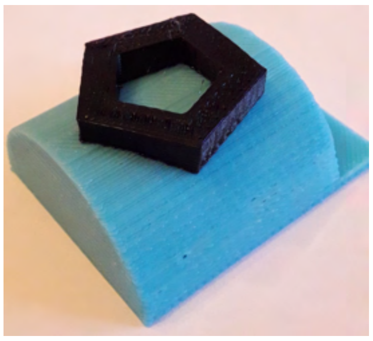



Figure 5.10: Non-planar multi-directional print with variable radius

5.2.4 Multi-directional: fixed tool setup

Up until now all the reported cases saw the use of the tool mounted on the wrist of the robot. An alternative solution adopts a fixed tool while the robot carries the printing surface.
Fig.5.11 captures a robot in the intent to realize an overhanging feature by tilting the substrate. The geometry is realized trivially (the objective was to test the setup) by generating a G-code of a parallelepiped and replicate the geometry on the side (as seen for the Epson example when working with reference frames).

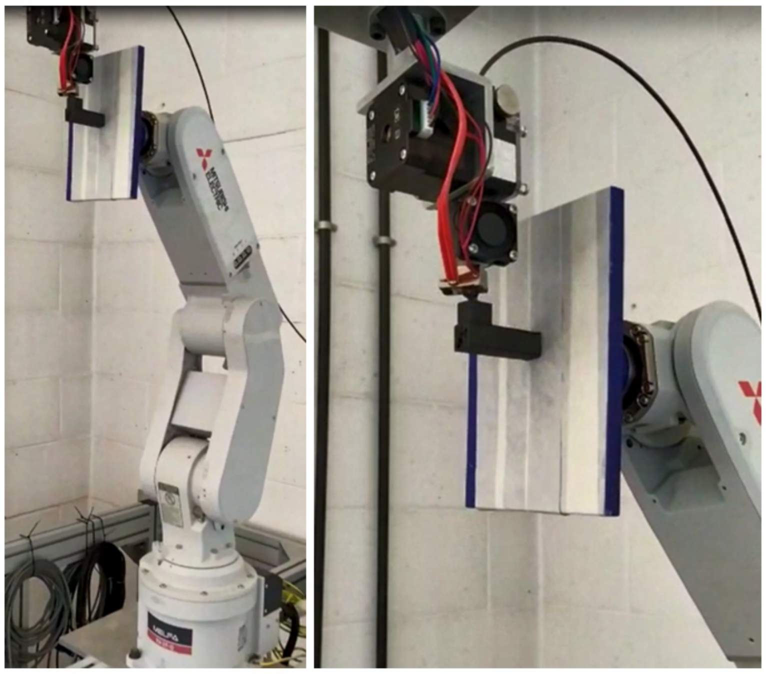


Figure 5.11: Fixed tool print example for multi-directional printing

5.3 Lathe-like printing

An explanatory case of a more complex building strategy is the lathe-like (introduced in 1.3.1), that allows to print on turning rods (as lathes do to remove material). Unlike the half-cylinder case seen previously, the CAM software used is the one discussed in chapter 4. It uses the complete lateral surface of the cylinder to extrapolate the contours and consequently the infill. It's self-explanatory that it's not mandatory for the starting rod to be of a circular cross section; it could be threaded, hexagonal or any shape according to the application but still take advantage of the generated woven structure.
Two cases will be reported; one to show the manufacturing process of tubes, while the other shoulders to mount bearings on an aluminum rod (speed profile test rig at page 53). First a brief description of the additional tool is due for clearness' sake.

5.3.1 Turning substrate design

The turning bed is made of an Aluminum rod with a 5[mm] diameter hinged at both sides to guarantee the rotation axis and to avoid possible axial motion, due to the force (although small) exchanged with the extruded material, ending up to the motor shaft and causing positioning errors. The rod is linked to a stepper (NEMA17, 0.42Nm holding torque, 200 steps/rev and driver up to x16 microsteps/step) by means of an elastic coupler. This choice is done because the housing of the rod already takes care to retain its positions, therefore it just compensates possible angular misalignment.

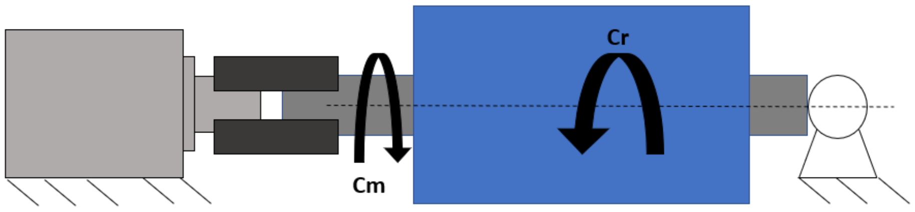


Figure 5.12: Examples of the lathe-like slicing procedure.

The system is designed considering the following data: the length of the rod is 20 [cm], the maximum radius R, that can be printed, is 5 [cm], the density of the rod is 2700 [Kg/m^3], while the one of material, PLA, is 1240 [Kg/m^3] (approximated as 0.45 of the one of the rod).
The inertia of the rod is: $J_z = 0.5\rho\pi L r^4$
The one of the printed material is: $J_z = 0.5 \times 0.45\rho\pi L(R^4 - r^4)$
Summing the two: $J_z = 0.5\rho\pi L[0.45R^4 + 0.55r^4]$
Neglecting all other possible forces acting on the system, Newton's second law can be written that states that for the data considered and the entire work volume filled, the angular acceleration at 0.35Nm is 147 [rad/s^2]. The result gives enough room to increase the print speed or the microstepping (thus lowering the motor torque) without the need for adding a reduction stage.

The deflection of the shaft under its own weight has to be calculated (using [99]) knowing that E is 69[GPa]:

$$f_m = \frac{PL^3}{384EI} = \frac{0.25\rho\pi g L^4}{384E \times 0.0491D^2} = 0,416\frac{\rho L^4}{ED^2} = 1.04\mu m$$

The deflection will not affect the print during the first layer (subsequent ones are not an issue since the only term changing and increasing while keeping ρ, L and E constant, is the squared diameter of the rod in the denominator, thus causing a reduction of the fraction).
Concerning the control of the angular position of the turning bed, the software is analogous to the one discussed about the speed control, but with the addition of lines of code to keep track of the number of high-low transition of the PWM stepping signal and its state to properly compute the commands.

5.3.2 Tubes

Let's consider a hollow cylinder having a diameter equal to 10[mm] and 50[mm] long, modeled using a commercial CAD software. Using conventional strategies, it could either be manufactured laying down or standing up, in both cases compromises have to be done between geometrical accuracy and mechanical properties. Using the turning substrate several issues can be mitigated. In order to get the machine instructions

suitable to control the cell, the CAD model is uploaded to the slicer coded.
In Fig.5.13, the main steps, highlighted by the user panels, are shown for clearness' sake. The STL is first oriented manually or automatically to suit the convention adopted by the algorithm; then it is sliced to get the contours that will constraint the infill and processed to get the final Gcode. The machine instructions are uploaded to the robotic system that controls both the device to turn the substrate and the extruder. In the same image, it is also possible to see the two superimposing deposition patterns one revolving around the shaft axis and the one along it.

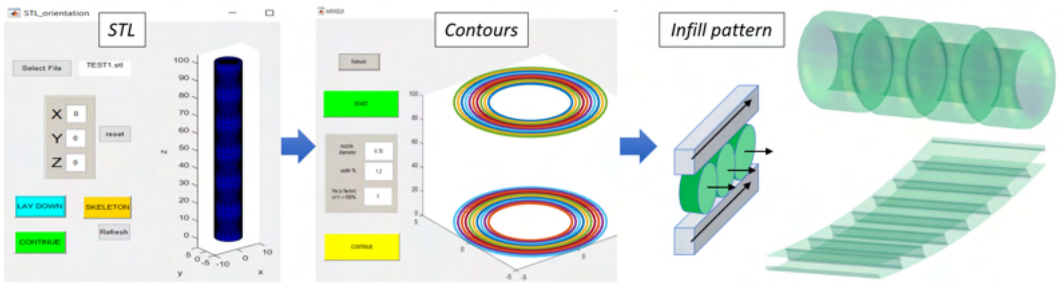


Figure 5.13: Examples of the lathe-like slicing procedure with infill patterns.

In Fig.5.14 the final product is shown; two versions have been printed to remark the infill pattern (all cases reported lack the deposition of the contour, to test the infill generation). Two additional cases have been appended that manifest the possibility to deal with different geometries: one with shoulders and one with variable tapered ends.

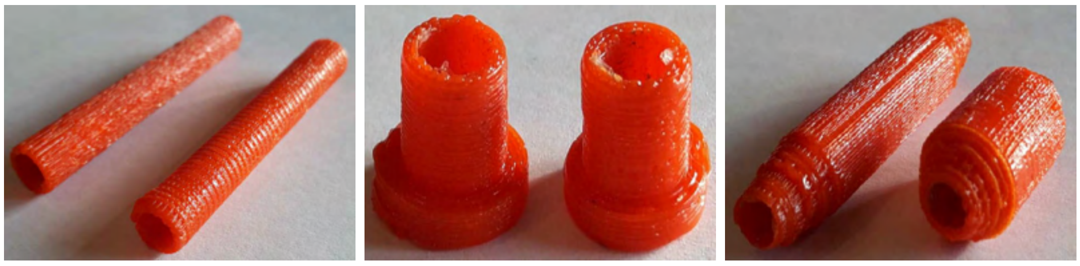


Figure 5.14: Final tubes; Shouldered and Tapered rods

5.3.3 Shaft customization

Let's consider a case study in which its analogy with the conventional manufacturing is highlighted. For instance, the design of a test rig for the control of stepper motors (the analysis of the speed profiles of the extruder at page 53).
In Fig.5.15, it is reported the concept behind the design; the housing features a NEMA 17 flange on both sides to mount it on devices or stands; the rotation axis is obtained by means of two ball bearings (625ZZ) where axial preloading is used to get rid of possible

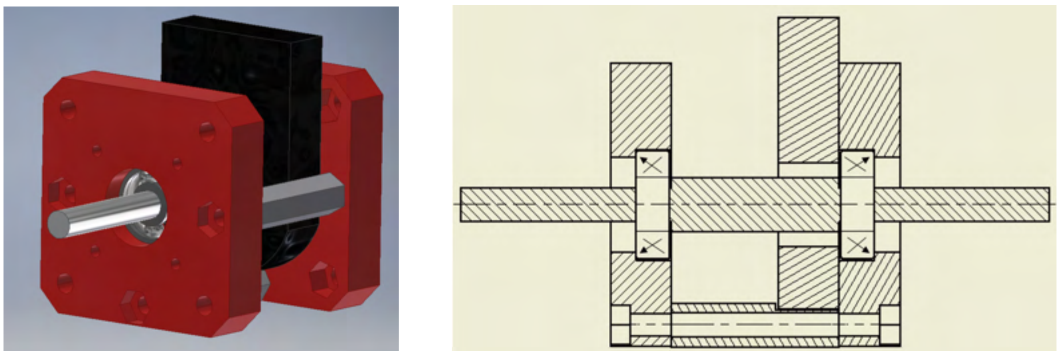


Figure 5.15: Tentative design.

backlashes in the final rotation. Hence, the shaft should resemble the cross-section in Fig.5.15 where the diameter is brought from its initial dimension of 5[mm] to 8[mm]. In order to obtain the double shoulders, the print is done directly on the lateral surface of the aluminum shaft. Using SM technology (i.e. turning) this part could have been obtained by starting from a 8[mm] shaft and removing chips at the sides to reduce the diameter to 5 [mm] (AM method reduces the scraps at the end of the process).
The layer pattern is the same as the one of the first example. Fig.5.16 contains these two distinct moments of the build (along the shaft rotation axis and, in the next layer, revolving around the shaft).

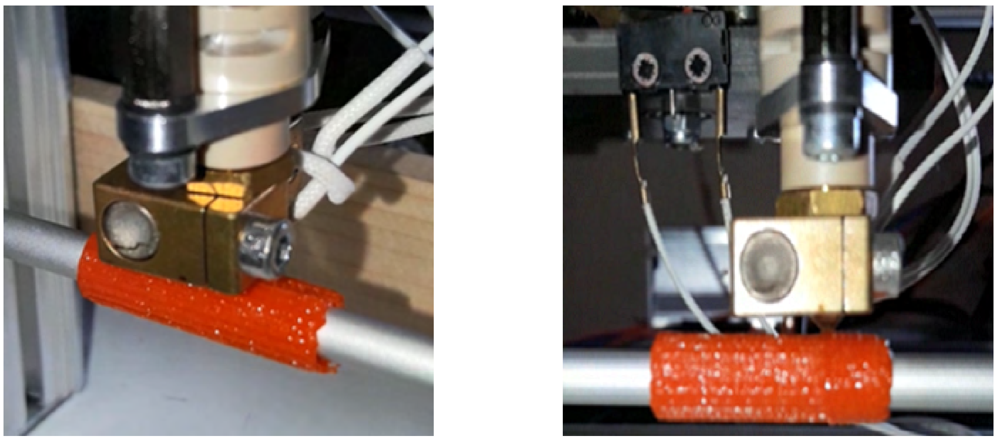


Figure 5.16: Example of shaft customization.

The adhesion of the new feature to the shaft is corroborated by the experimental campaign showing the shaft synchronized to the motion of a stepper motor (feed-forward control). Further investigations are required to investigate the torque transmission capability.

5.4 Conclusions

These trials had the double function to validate the setup during the actual process and to highlight the weak and strong point of the current trends in AM building strategies. All the robots have been successfully converted to AM machines using the most suitable control strategy to simultaneously master the robotic arm and the external device (i.e. extruder and turning substrate). From an aesthetic point of view the parts do not present excessive defects (except for those due to the absence of filament retraction mechanism during the gable-roof testing, although implemented in the following case studies).
They also have shown the possibility to customize builds with reinforcements or additional features (that could lead to repair partially damaged object) by adapting the print to the substrate while being able to shape the cutting surface between passes to end up with a desire curvature.
Another consequence of this test is to remark that, despite being able to tailor the print according to different surfaces (hence eliminating need of support structures), issues related to the weaknesses at the interface of two surfaces is still present. The only example that guarantees a behavior close to isotropic is the lathe-like declination of the cell thanks to resulting texture; yet, its application is limited to specific types of geometries. The others generate parts whose deployment in action has to be as close as possible to the one intended; unaccounted and unexpected loads component could act on the weak boundaries causing separations, hence failure of the object.
New building strategies are therefore required to avoid (or at least limit) these issues. Feasible solutions, integrable with building surfaces as seen up until now, will represent a step closer towards stronger, more reliable and performing parts.

Chapter 6

Bioinspired building strategies

In order to widen the sources of inspiration in the search of novel manufacturing AM methods, the natural world is glanced at. A brief overview of bioinspired solution found in AM is done in section 6.1.
Example of manufacturing philosophies (waving, subtraction, addition and growth) adopted by living creatures are reported throughout the chapter. The first one, based on *Waving*, inspired a building method named *Stitching* (6.2) whose main characteristics are tested in section 6.3 to assess its feasibility.
In subsection 6.4, two other novel building strategies are also theorized starting from different philosophies (respectively *addition* and *growth*). They are namely *Layer Shaping* and *TreeD printing*.
All of these solutions can be traced back to a particular behavior found in living creatures, but without sharing any physical link to its original process (they are not an exact replica of the actual phenomenon neither they are meant to be). These adding methods are discussed and theorized pointing out the devices (hardware) and software required to develop the corresponding cells, that namely are: the robots (number and type), the tools (for AM, SM or other forms of manipulations), the toolpaths (2.5D or full 3D) and trajectory planning. A synthesis on the main aspects of these processes is put to serve as an easy to access source of information (section 6.5).
This building strategies also influence the structure of the robotic cell (section 6.6) and how the elements inside interacts with each other.

6.1 Novel building strategies: Biomimicry Manufacturing

Several examples of bioinspired solutions applied to manufacturing and especially to the additive world can be found in literature [100].
As stated in the aforementioned reference, a bioinspired product design can be faced in two ways: “biology-to-design approach” in which nature inspires an engineering concept or “design-to-biology approach” where a specific problem is untangled by looking for a solution in nature.
The contextualization of these biomimetic principles is found in the form of freeform,

simulation-driven and lattice design. Freeform design engineers surface that compromises shape fitting and functionality (bone infill pattern [42] and tree-like support structure [101]). Simulation-driven design optimizes the part by iterative process of simulation to redistribute and empty the material to achieve the stiffness to withstand an expected load (mimicking the evolutionary process). Conventional AM building strategies might not be able to keep up with the geometries yielded by this optimization. Last, lattice design copies the tendency of natural systems to exploit modular structures for their dwellings. Lattices leads to lightweight, high specific stiffness, fracture resistant and crack growth arresting structures.
Solutions found in literature are revolving on topological mimicry of the natural world, but not so much on the manufacturing strategies engineered by leaving things. Although shapes and forms are commonly adopted, the building strategies still retain connection with its parents CNC subtractive technologies.
In nature there are three main building strategies that living creature use: weaving, subtraction, addition and growth. They constitute a source of inspiration for possible AM building strategies that will be articulated in the following discussion.
Let's start with the analysis of the weaving approach.

6.2 Nest building (Weaving) and *Stitching*

Ploceidae is a family of birds that weaves fine leaf fibers and grass to make their nests. For this reason, many of these birds are called weavers or weaverbirds.
Fig.6.1 shows a bird in its intent to build a dwelling. The building process can be split into two main phases [102]. Ploceidae starts by anchoring strands to the tree (*initial attachment*); then they proceed to form a central ring at the bottom (*ring phase*) that complete the bearing structure. They finish by enclosing the frame with additional material to protect the eggs.

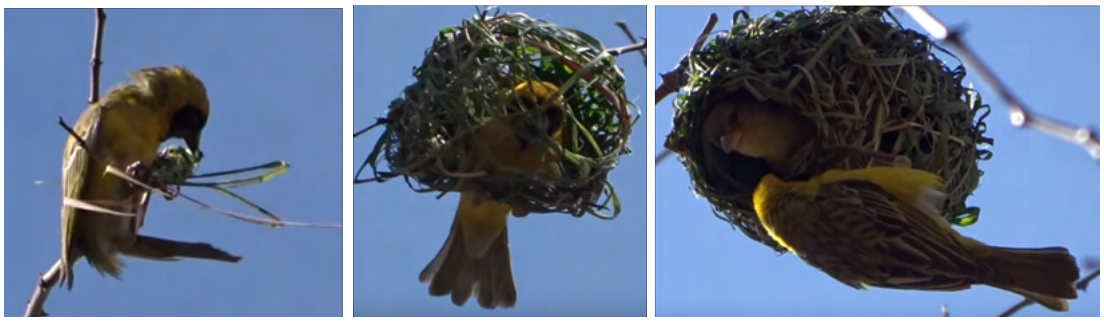


Figure 6.1: Weaving birds building a nest [103]

This approach inspired an AM building strategy named *Stitching*. The idea is to 3D print single blocks of the overall part that are then merged together by printing connecting elements that could act as actual stitches, binding material or generate weaved-like structure (Fig.6.2).

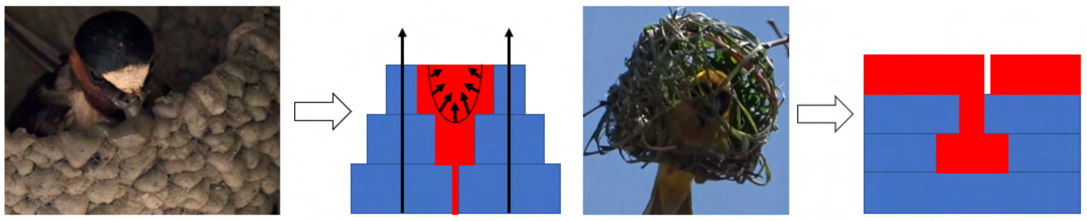


Figure 6.2: From nature to *Stitching*

Stitching can be therefore declined into two categories. The first one builds the object using small subprints that are then bond with additional material following a different building orientation (sharing some connection also with the *addition* manufacturing philosophy found in 6.4.2). The change is proposed to act as discontinuities in the part to reduce crack propagation. This typology requires at least one 6 dofs robot to make sure to reach all possible gluing regions. CAM software should deal with the definition of a feasible subdivision in addition to slicing techniques. Fig.6.3 shows a possible 3D model subdivision in sub-elements (*cyan* blocks) that can be either filled (*top branch*) or contained (*bottom branch*) first. The differentiation depends on the possibility to reach the binding points dictated by the blocks shape and organization. In order to improve strength of the final build different building directions can be advised. For instance, blocks could be generated using planes parallel to the bed, contours (*red*) orthogonal and infill (*green*) following a part-defined surface in space.

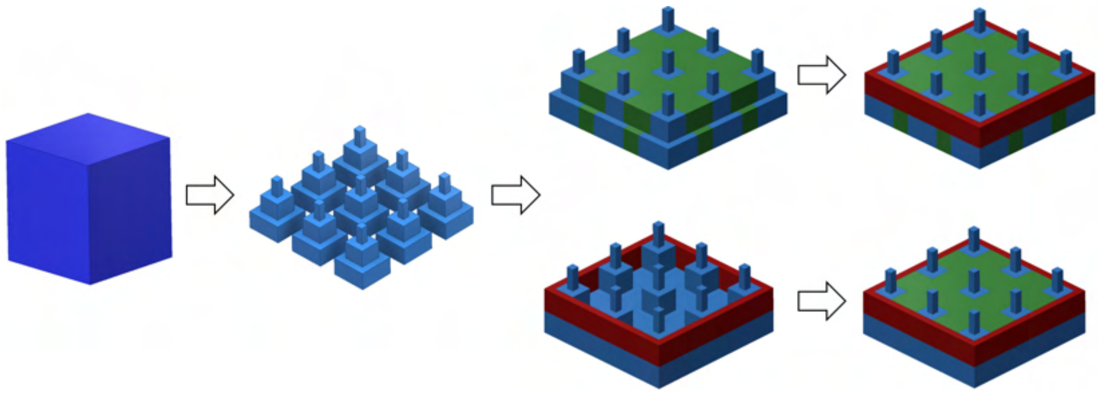


Figure 6.3: *Stitching*: subprint blending

The second one builds object by generating pockets during the print that serve as anchors among layers. They could act as starting point of a layer (Fig.6.4-*top branch*) or acting as actual stitches (Fig.6.4-*bottom branch*). The increase in strength along the printing direction is given by a resembling dovetail joint; shear action is mitigated by interrupting a layer superimposition (weak bond) with rods of material. The geometry of these chambers has to be investigated as well as the filling process. This operation should not last long to avoid excessive heat transfer to the substrate that could cause uncontrolled transformation.

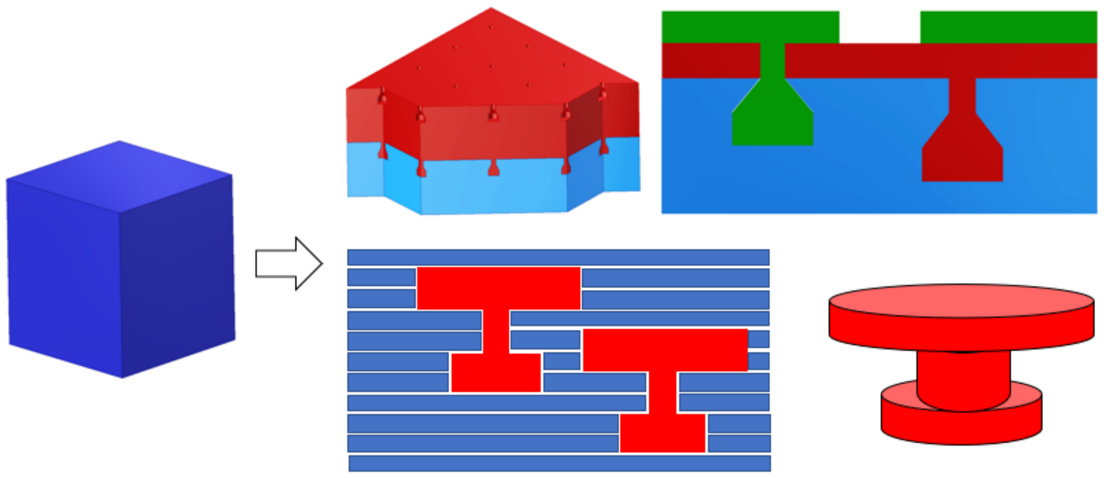


Figure 6.4: *Stitching*: layer joining

Path planning does not deviate significantly from standard algorithm. It requires to define the original CAD model subtracted by the pattern of joints that also gives information on the starting points of the layers/stitches. A stitch can be realized by positioning the TCP as close as possible to the hole to be sure to fill the chamber (the outer nozzle diameter should generate enough pressure to stuff the mold). Extrusion of a quantity of material equal to the volume of the dovetail precedes a combined movement-machining instruction that fills the sprue. Circular pattern can be used to end the joint before proceeding with the rest of the layer.
Cartesian robots are more than enough for this application (experimentation pending). Extruder with tapered nozzle could provide easy insertion in the chamber (like in IM applications) although this could represent several technological issues (increase in pressure required to extrude material due to interaction filament-walls, substrate adhesion to the outer walls).

Stitching as has been presented could evolve in a form that uses pre-built object (either by means of AM or IM) that are then “glued” together, this will significantly reduce build time. One additional advantage, once developed the process, could be the insertion of structural blocks containing sensors, devices and such to make ready-to-use parts. It will be referenced as *Stitching-2* to differentiate it from its parent approach.
Stitching-2 will require at least two manipulators, one for printing and one for insertion (tool: gripper). This will impact the trajectory planning to be able avoids collision and menage the time chart of the process.
If the print volume can be placed on the EE, then one robot could be used. The cell could feature a fixed extrusion system as well as a feeder-dispensing blocks that the robot (featuring the print bed) reaches from underneath.

6.3 *Stitching*: experimental test

In this section some key aspect that might prevent the feasibility of the *Stitching* building approach is analyzed.
The core of the *Stitching* method is the possibility to build intertwined structure by creating inter-layer links (stitches); this is achieved by leaving holes in the previous layers that then serve as starting points of the new layers, thus creating passes that are not limited to their respective planes. The feasibility to fill the anchoring hole and to continue the print is a must, if not possible the method will have to be dismissed.

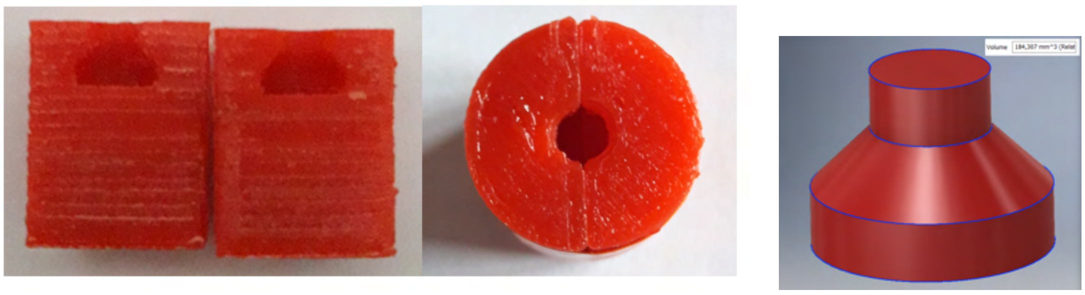


Figure 6.5: Test case to validate the *Stitching* approach

6.3.1 Qualitative test 1: volume filling

To inquire the correct stuffing of the dovetail, it has been decided to split the starting block in two halves, so that it would have been possible to separate them to see inside. The mold is made of two half-cylinders with a cavity that, when juxtaposed, forms a known volume. The cylinder is manufactured using conventional 3D printing (90% infill, 0.15[mm] layer height, PLA printed at 210°); its dimensions are $\varnothing 16 \times 20[mm]$. The hollow feature has a volume of $184.3[mm^3]$ ($\varnothing_{max} 8 \times 2$, $\varnothing_{min} 4 \times 2$ and overall height 6[mm]). They can be seen in Fig.6.5.
Here, the trial just focuses on extruding material inside the chamber. This allows to separate the print of the half-cylinders (juxtaposed to form the chamber) from the filling of the dovetail. The actual procedure is reported in table 6.1 with every line of the G-code commented; the important points are here discussed instead.
The mold is placed in the workspace in a known position; the nozzle is heated up and some material is extruded to fill the extrusion chamber. It is then positioned using a reversed U shape into the sprue where the material is extruded.
The quantity of material to extrude is evaluated, knowing the diameter of the filament (3[mm]) and the volume of the chamber ($184.3[mm^3]$). It is possible to write:

$$E \times A_f = V \quad \rightarrow \quad E = \frac{4V}{\pi d_n^2}$$

it yields that 26[mm] of filament has to be extruded. This value is tentative increased by 15% to be sure to fill the chamber.

Table 6.1: volume filling: G-code analysis

M104 S215	Temperature is set to 215°without pausing the code
G28	Home position is performed
G1 Z5 F5000	and the Z axis is increased by 5mm from the plate.
M109 S215	Program is put on hold until the temperature is set.
G21	Unit of measure chosen is millimeters.
G90	Points are defined in the absolute reference frame,
M82	same is done for the extrusion.
G92 E0	Extrusion current position is set to 0.
G1 F480	Speed to use is 8 [mm/s]
G1 E30	30[mm] of material are extruded in air
G92 E0	Extrusion current position is set to 0.
G1 Z22 F3000	Extruder is lifted to 22 [mm]
G1 X180 Y80 F3000	TCP is positioned in (180, 80) at 50[mm/s].
G1 F600	Speed to use is 10 [mm/s].
G1 Z19.4	TCP is lowered inside the specimen to 19.4.
G1 F480	Speed to use is 8 [mm/s].
G1 E30	30[mm] of material are extruded inside the piece.
G1 Z22 F3000	TCP is removed at 50[mm/s].

Attempt 1 succeeded to top up the empty space, leaving some space in the sprue due to presence of the nozzle. The half-cylinders got glued together, manual attempt to separate them was unsuccessful. A blade was then used; this operation showed that only the region with the dovetail was keeping the two parts joined. Forcing with the blade resulted in an earlier failure of one of the halves, but not in the region of the hole. By insisting on the smaller region obtained, it was eventually possible to see the root of the stitch. It had completely filled the desired region. Pictures taken throughout this process are illustrated in Fig.6.6.

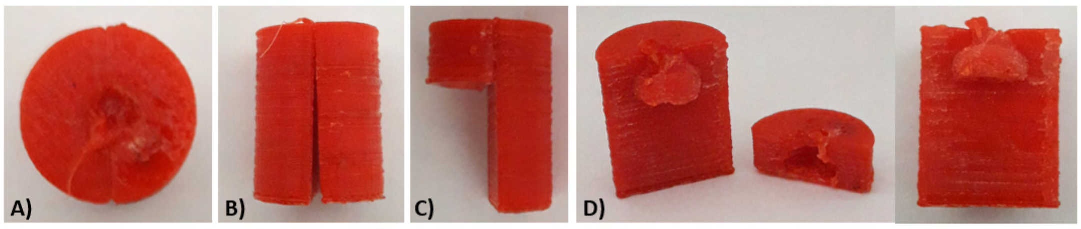


Figure 6.6: Volume fitting result: (A) top-view after test; (B) attempt to separate the two half with a blade; (C) failure of one half before the stitch root; (D) final exposure of the filled chamber.

6.3.2 Qualitative test 2: new layer

Once evaluated the possibility to fill a chamber, test needs to be carried out to check if it is feasible to continue the print. The same chamber is considered as well as for a simplified version of it (truncated cone with a volume of 49.5 [mm^3]). The specimen is created this time in one single print: the machine creates the base with the dovetail hole, once filled, it continues with the generation of a cylinder with a smaller diameter to remark the transition and make it easier to spot. This trial stands to mimicry the stitching process during the print.
Test have been performed using extrusion lengths equal to 6, 12, 18, 22, 30 [mm] to fill the chamber before starting the new feature on top of the base.
Block address equal to *E6* and *E12* have been used to study the formation of the stitch. The minimum value lead to a poor final geometry of the anchor, although the expansion of the filament once inside the chamber could still prove to be effective (the material injected wets the hull surface and expands inside but without completely filling the room). Doubling the length, the additional material reaches the bottom, but instead of expanding inside, it tends to reemerge as soon as the nozzle leaves (compliant with the fact that it's not possible to seal the mold with the extruder while injecting). They are reported in Fig.6.7.

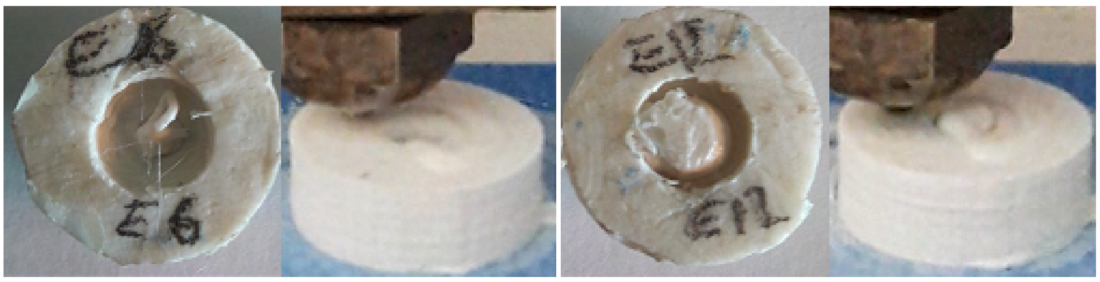


Figure 6.7: New layer: two case with under-extrusion (6mm and 12mm) to witness the anchor formation

Values equal to 18 and 22 are the next considered, with *E22* being the actual quantity required after shrinkage to carry out the process. Last, the extrusion length used for the first qualitative test is used.
In all these trials, a partial remelt of the last layer due to the presence of the nozzle (the pouring is done at 8 mm/s, thus a maximum permanence time of 4 seconds), combined by the presence of a leaking nozzle, leads to a poor final geometry at the discontinuity. An excess of material causes it to spring back, adding more plastic to the new layer and a consequent loss of precision on the first few new layers. Variation in the expected behavior could also be associated with a different filament used (white instead of red one from a different manufacturer). Final results are at Fig.6.8 in which it is possible to see how the interface between the two parts connected by the stitch is affected by a not complete fill of the chamber and equivalent over-extrusion.

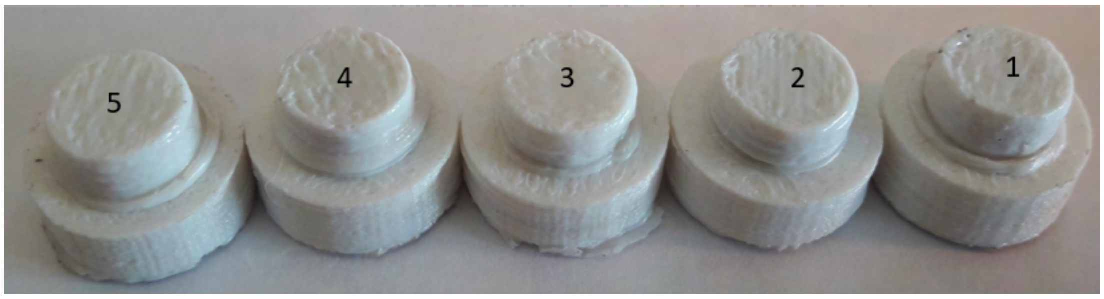


Figure 6.8: New layer with different extrusion length: *(1)* 30mm, *(2)* 22mm,*(3)* 18mm, *(4)* 12mm, *(5)* 6mm

A simplified hollow shape has then been considered; as mentioned earlier, it is a cone with the same sprue exploited this far. The reason behind the change is to speed up the exposure of the lateral surface with the new material while keeping the dovetail like shape (i.e. to yield the same joining principle). The volume is approximately equal to 49.5 $[mm^3]$, already accounting for the shrinkage.
Test conducted, using the evaluated volume, shown a proper wetting of the lateral surface of the cone and limited recoil affecting the following layers (using the red wire instead, test to assess the dependency on the filament used have to be carried out). In Fig.6.9, the shape of the cone chamber, a frame taken during the injection and the final result (up to the first new layer) are shown.

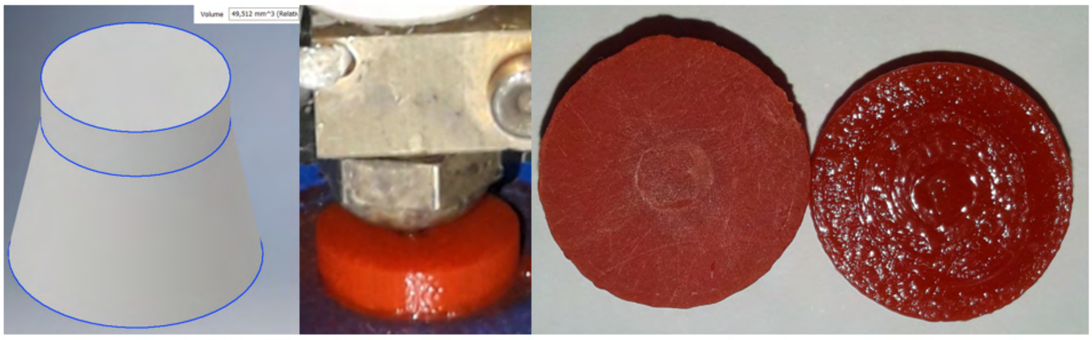


Figure 6.9: New layer: cone case

6.4 Other bioinspired solutions

Let's inquire the remaining three constructive philosophies found in nature to build shelters, colonies and menage self-growth that will lead to the outlining of other two novel strategies.

6.4.1 Subtraction

An example found in nature of this widespread mechanical manufacturing strategy is ants burrows.

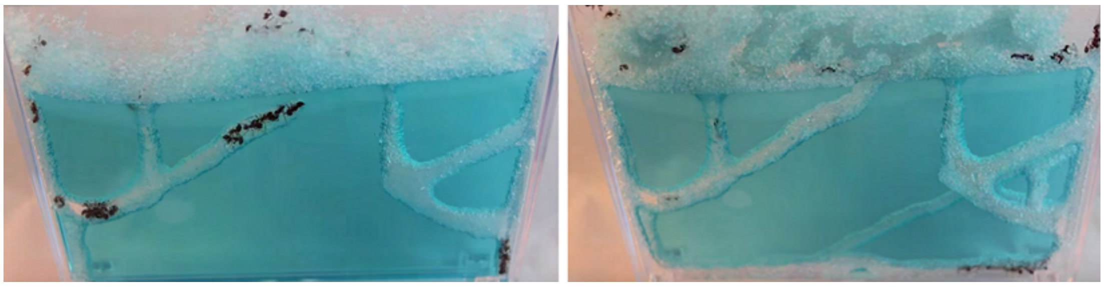


Figure 6.10: Ants building a nest [104]

Fig.6.10 shows a colony of ants lifting and moving material to form tunnels and branches [105]. The study suggests a density-dependent drive to excavate (the lower the nest density the higher the inhibition to expand); it also shows a dependency on time (to lift the grains back to surface) and substrate texture.

6.4.2 Addition

Example of building by material addition can be found in the animal world.
The Mud Wasp grows concentrically each cell, where an egg will be placed, of the nest before enclosing the entire structure [106]. Fig.6.11-6.12 shows a wasp in the attempt to create a cell. The material deposited (as the name implies, this insect uses mud retrieved in the near area to construct the cell) is squeezed and kept in position while drying to retain the desired shape.

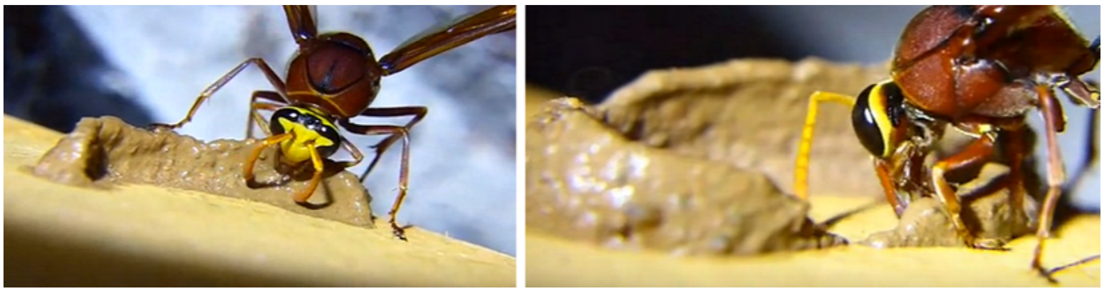


Figure 6.11: Mud Wasp building a nest [107]

Same trend can be found among other wasps [108]. The building material might differ, but analogies have been found on the "growth" strategy. They also enlarge the nest radially by adding 3D cells.
Rufous Hornero is an example of a bird that manufactures its nest by adding material;

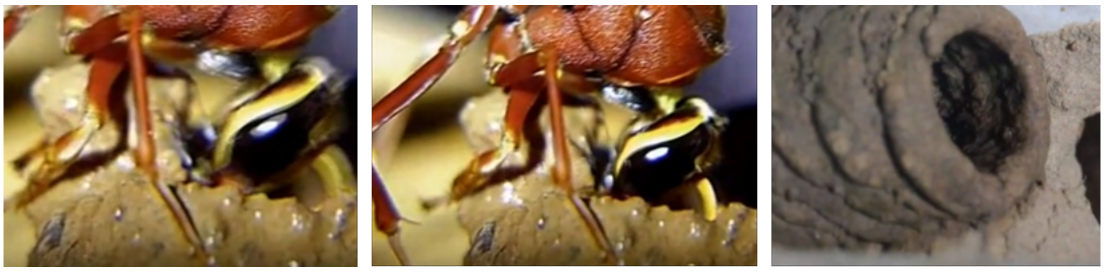


Figure 6.12: Mud Wasp adding a new part and final result [107]

swallows do the same (Fig.6.13) [109]. Their approach consists in the positioning of initial pellet to draw the base of the nest. The arrangement of the following pellets is performed to shape the nest accordingly (usually dome-like structure). The arrangements of the pellets are done in function of the part of the dome that is being built.
A last example is the Cathedral Termite. They build mounds above the subterranean nest to regulate the humidity and ventilation inside the burrows.

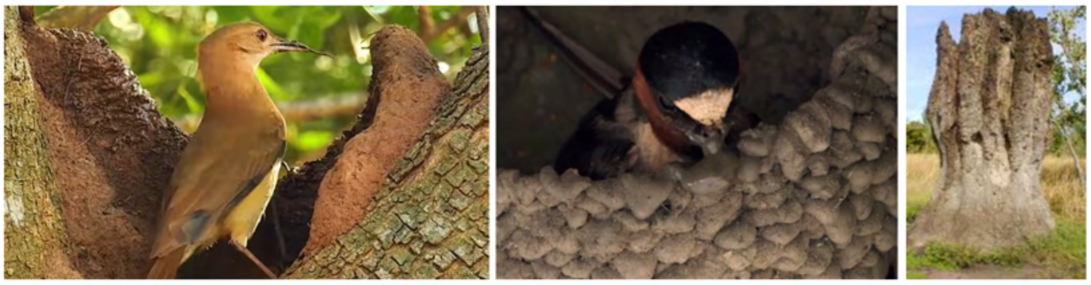


Figure 6.13: Rufous Hornero (*left*, [110]), a Swallow (*center*, [111]) and Termites (*right*, [112]) making their lairs.

6.4.3 Growth

Growth is the process used by living organism to define the size and shape to better strive in the environment. This adjustment can interest specific region (for example, plant shoot) or the whole cell system (for example, embryos) [113]. Mitosis is the mechanism in which a mother cell is split in daughter cells. It can be described as process of growth-division (cell increases size before being cut in two) or division-growth (cell splits and then the siblings cells get bigger).
Let's consider the case of plants and particularly trees. In Fig.6.14, cross-section of a trunk is depicted as well as branching formation. The trunk has an outer layer (*bark*) that protects the tree; moving towards the center cells that manage water transportation form a second layer (*sapwood*); at the center old sapwood layers forms the core of the plant (*heartwood*) that gives mechanical strength to the final organism.

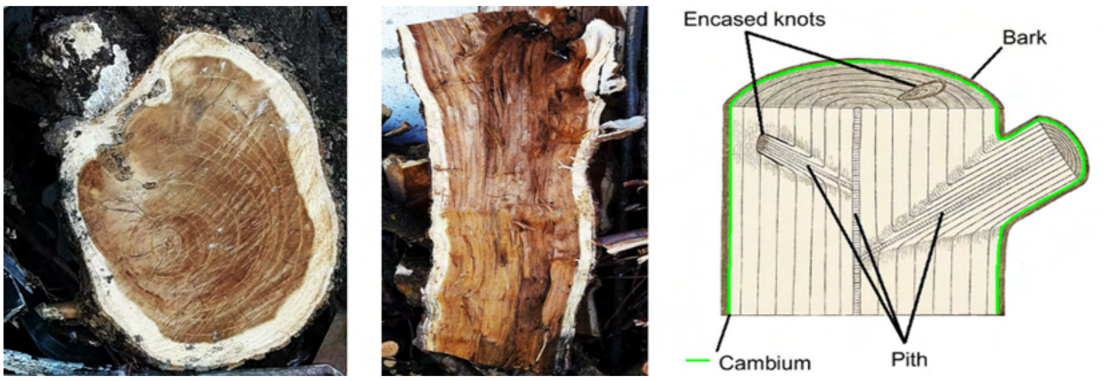


Figure 6.14: Tree structure (right image [114]).

6.4.4 *Layer Shaping* and *TreeD printing*

Analyzing these behaviors found in the living kingdom, two possible AM building strategies can be derived. For the sake of clearness, they have been tentative named: *Layer Shaping* and *TreeD printing*.

***Layer Shaping*:** This approach is inspired by nature additive experts (Mud Wasp, birds and such) that builds by first placing the material and in a second instance shaping it into form (Fig.6.15). They do not limit themselves by just adding and shaping a layer but making sure that adherence on the old substrate is strong enough to bear the structure.
Current trends in additive manufacturing sees the intervention of other devices (for example, drills and milling bids in Hybrid manufacturing) during the process to compensate for geometrical errors (e.g. staircase effect) in positions that otherwise will not be accessible after completion (for example, inner chambers and conduits).
Layer Shaping consists in extruding a draft of the desired layer before, as the name suggest, shape it by means of tools (contact or contactless). Tools hypothesized are subtractive (hybrid system) such as drills, cutters and lasers; material, instead of being removed, could be spread by the EE (heated rods and plates) in the surroundings (in this case, the complexity in generating a feedforward plan to print the part increases). This method suits better large prints in which shaping will significantly reduce lead time while, of course, increase scraps and nonreusable material (if the material is not

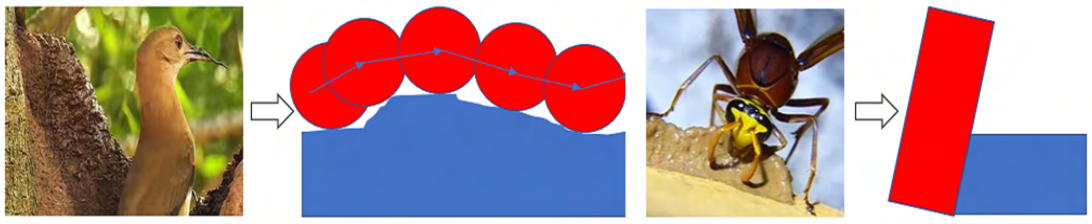


Figure 6.15: From nature to *Layer Shaping*

recyclable).
The robotic cell will require at least one robot with multi-tool on the EE or more likely a binary machine system with two distinct manipulators (one for the addition and one for the subtraction). Cartesian (or Gantry) and Arm robots can be selected among the serial manipulator for their dexterity and large volumes. 3-axis machines can be used for the AM part since they will have to just create a draft, while for the SM (or reshaping) task, machines with more than three degrees of freedom allows to obtain the final 3D shape. SM can be done using machines with serial kinematics chains, but PKM are more rigid and suitable to bear potential high exchanged forces (if this strategy would have been transferred to metal-based AM technologies). The development, or at least the integration, of the two tools is required to carry out the process.

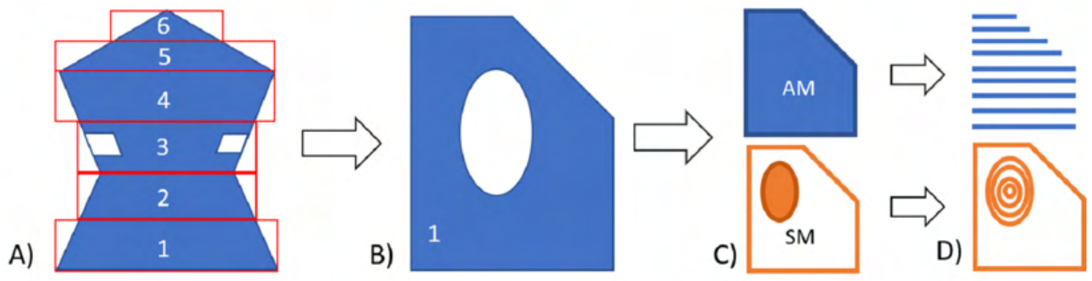


Figure 6.16: Layer shaping: (*A*) layer division according to accessibility; (*B*) single layer analysis; (*C*) AM - SM part identification to generate the path; (*D*) path generation.

Fig.6.16 shows the ideal workflow from the CAD model to the path generation. The possibility to intervene in a second time allows to slice the part not by means of planes, but parallelepipeds (Fig.6.16-*A*). Their heights will depend on the accessibility for the tool to do its job; if possible, some feature could be machined at the end (*layer 3* of the figure) to provide support to the other layers. The analysis can be then carried out layer-wise. For each layer (Fig.6.16-*C*,*D*) the two operation has to be told apart; the part obtained by means of AM will draft the layer, while the SM one will specify it according to the design. It's expected to finish the sector by performing holes and machining the contours. Chips removed might remain in the build region if not properly expelled, therefore means to keep the area clean is ought to be integrated (air might cool excessively the part that needs to be reactivated to promote interlayer bonding).

Once determined the toolpath, the actual timing chart and motion law has to be sought. Addition and subtraction can be carried out independently (*discontinuous* process) or altogether (*continuous*). The former (Fig.6.17-*left*) treats one operation at a time. Therefore, no collision between the two machine is expected throughout the build; operations are paced by each own technology process; transition could be zero-lagged or delayed promoting optimal condition for the following step (temperature wise). How the abscissa is placed along each path does not affect the two stages.
The latter (Fig.6.17-*right*), on the contrary, tries to superimpose the two phases to

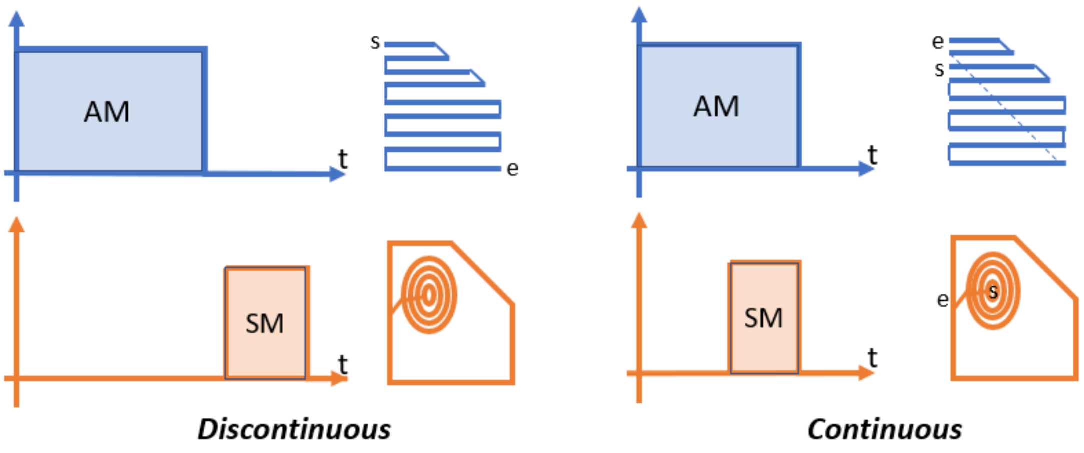


Figure 6.17: Trajectory planning possibilities of *Layer Shaping*.

minimize built time. Therefore, trajectory planning should account for both possible collisions and feasible routing. This last concept can be easily understood by an example; referencing the same figure in which an elliptical hole has to be milled, starting extruding material from this region could be sensible because it will give time for the extrusion system to move elsewhere leaving room to the bid to remove the material in this region.

***TreeD printing*:** This approach, as the name suggest, is inspired by the tree trunk structure and formation (Fig.6.18), although other living things show same principle (for instance human bodies with bone-tissue-skin cross-section at the limbs). Looking at the picture, it's possible to notice how the structure is made up of a rigid core (pith) surrounded by lower density material and enclosed by a rigid skin. The bone frame is remodeled by possible secondary branches that defines the final structure so to better cope with the environment (i.e. gravity compensation, weather etc.).
Analogously, a component could be build defining a skeleton representation of the form (as can be seen in Fig.6.18) that will serve as base for the actual manufacturing.

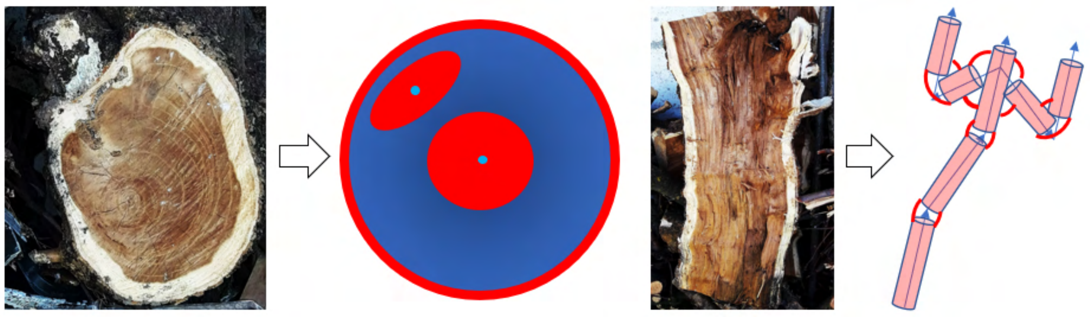


Figure 6.18: From nature to *TreeD printing*

It's trivial to comprehend that several problems might arise. Accessibility, being one of them, can prevent or constraint the optimal wire deposition direction especially in concave areas. Skeleton generation should account for the geometry of the tool and broadly of the robotic cell. The mechanical system has to be of at least 6 Dofs to orient the part with respect to the TCP; additional Dofs could be used to operate in a wider range of feasible solutions. Skeleton, even if build following the pith direction (thanks to the complete set of degrees of freedom), might give birth to weak bond among layer. Hence, the addition of a "sapwood" layer should be carried out to strengthen these points, if manageable.
Fig.6.19 tries to point out the "heartwood" generation by referencing a generic shape that unfolds initially inside a plane and later splits in two branches. The root grows according to the skeleton direction until a first deviation is required; here an increase of the cross section provides better support for the child branch. If it jets out, the core extends to the border. This generates solid bodies right to the edge of the main part before it extends to the needed branch.

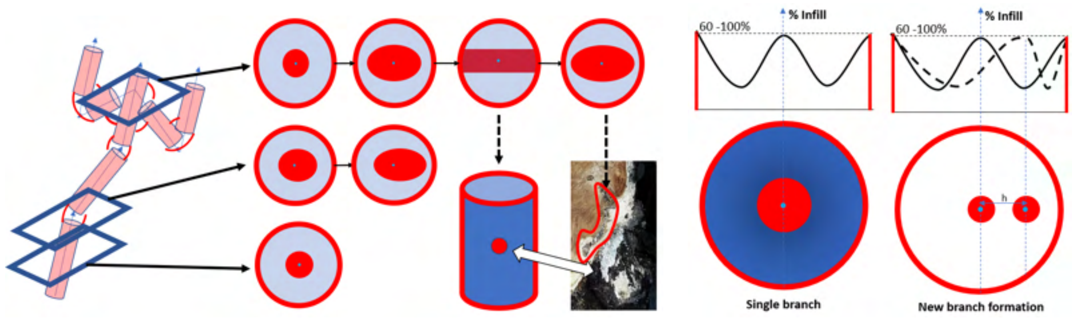


Figure 6.19: *TreeD printing*: skeleton (branch) formation.

The cross-section of the to-be-built part can be generalized as a solid contour followed by infill (lower density) and at the center the skeleton (Fig.6.19). The infill percentage resembles a cosine wave that, in polar coordinates centered in the pith, propagates along the radial distance. A new branch, propagating from the spine itself, can be seen as a perturbation of the original one leading to a new infill percentage distribution.
Once completed the skeleton, the rest of the build has to be managed to assure a strong nodes and body and density distribution sought. Fig.6.20 reports a possible solution for a shaft with a significant variation in the section (for instance a shaft with the gear already printed on it). Having terminated the spine (anticipating also the new section by having a variable diameter along th height, Fig.6.20-*A*) using standard printing strategy, the piece is tilted 90 degrees with respect to the TCP. The red strands in Fig.6.20-*B* are then placed to reinforce the core by turning the rod on its axis (*lathe-like printing*); sub-figures *C-D* show the new branch being formed (using 4 cylinders as skeleton of the overhanging feature) combining different building directions. The final

part is the result of branching and ribbing with variable density.
Other possible branch formation and growth require additional study to improve and generalize the process.

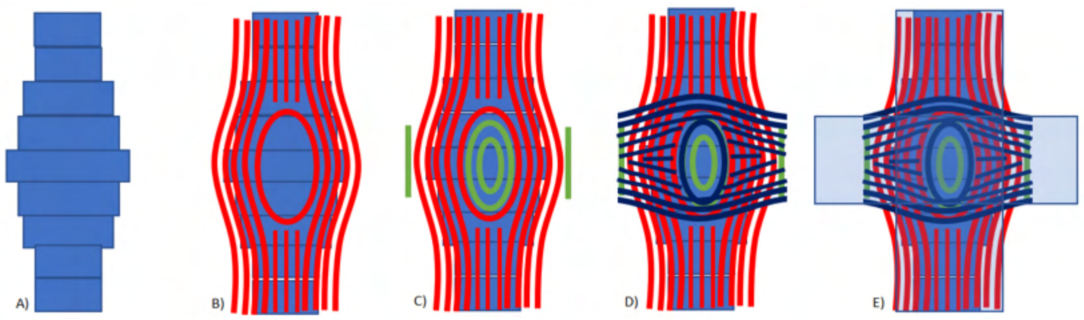


Figure 6.20: *TreeD printing*: possible building strategy.*A* reports the spine formation with conventional FDM, *B* depicts the addition of a reinforcement layer along the building direction; *C* the propagation of the bone structure along the new direction; *D* reinforcement are then added; *E* the final part is the superimposition of pith and "sapwood" layers.

TreeD printing requires CAM software able to generate the skeleton of the model (several algorithms can be found in literature) and infill, both of them, relaying on multi-directional/curved-layer building. Generalization of this process requires a thorough analysis. The resulting toolpath has to account also for possible collision between object-robot (tool included) since it belongs in the complete 3D space. This impacts the robot selection (of course also the work-volume significantly affects the cell layout). As anticipated, redundant system allows to mitigate possible limitations in the positioning. The extrusion system could be either fixed on the ground or on the EE.

6.5 Recap of the methods

Up till now, four possible novel building strategies have been theorized: *Stitching*, *Stitching-2*, *Layer Shaping*, and *TreeD Printing*. Table 6.2 condense what stated for the robotic systems and tooling. The manipulators range from 3 to 6 degrees of freedom, serial and/or parallel kinematics. Extremity devices are "auxiliary" (for instance a gripper) or fundamental (i.e. extruder, spindle).

Table 6.3 summarizes the significant aspects concerning toolpath generation and trajectory planning. It possible to infer that the toolpath planners need to access to standard slicing and milling algorithms to generate the geometrical curves; 3D models have to be processed in advance to decompose and assign each part to different manufacturing technology (AM and SM). Trajectories, on the other end, heavily relies to collision check and avoidance to allow different machines and moving parts to coexist; hence, time charts to schedule the job are mandatory.

Table 6.2: Biomimicry AM manufacturing approaches: *robot* & *tool*

Name	**Robot**	**Tool**
Stitching	• 3 Dofs robot	• Extruder
Stitching-2	• 2 manipulator with 6 Dofs (arm robot) • 2 robots (3 Dofs + 6 Dofs) • 1 manipulator with 6 Dofs	• Extruder • Gripper
Layer Shaping	• 1 manipulator with 6 Dofs (arm robot) • 2 robots (3 Dofs + 6 Dofs)	• Extruder • Spindle / Others
TreeD	• 1 machine with more than 4 Dofs	• Extruder

Table 6.3: Biomimicry AM manufacturing approaches: *Path* & *planning*

Name	**Toolpath planner**	**Trajectory**
Stitching	• Part subdivision • Slicing (standard - generic surface) • CAD model manipulation	• Process control • Collision
Stitching-2	• Operation manager • Slicing (standard - generic surface) • CAD/part subdivision	• Time charts • Collision
Layer Shaping	• Slicer (standard) • CAM • Operation manager	• Time charts • Collision
TreeD	• Skeleton • AM CAM (general problem)	• Collision

6.6 Novel building strategies: AM cells

These plausible solutions require consequently the upgrade of the robotic cell to cope with multiple machines and or tools. An AM (SM as well) cell has three main component that needs to be known, from the most important to the lesser: tool, robot and substrate.

Tools define the technological mean of how the material will be added, thus constraining the possible outline of the cell. For example, a filament based extruder (see 2.1) is able to function from vertical down to horizontal, but not against gravity (it could be feasible, but the risk to clog the extruder up is higher); its low weight (cartridge included) can be easily moved by an industrial robot. A screw-based extrusion system (see section 2.2), on the other end, can be bulky; the hopper (or generally the pellet feeding system), if directly connected to the body, prevents the tool from any or excessive reorientation. Broadly speaking for all the type of extrusion-based AM tools, the material exiting the nozzle will be subjected to gravity and stresses related to gradient temperature (boundary in contact with air against a melted core) that will bend and reshape it. Relevant is the distance between the TCP and the substrate, that mitigate these forces (the scale and forces interaction changes from the micro to macro scale).
Tools and cells, consequently, can be therefore set up in two configurations: *mobile* or *fixed*. A *mobile tool* means that the TCP will be repositioned at least one time throughout the building process; a *fixed tool* will never change its position with respect to the world reference frame.

The robot can either belong to serial or parallel kinematics machine family. Both have their pros and cons that should be accounted for when choosing witch one to choose. PKMs tend to have larger payload capability, lower inertia and higher stiffness than their serial counterparts, but at the cost of a limited workspace (lower useful workspace - machine volume ratio).
From the robot point of view, the EE could either move the tool (*mobile tool* scenario) or the substrate (*fixed tool* scenario). It is self-evident that collision could occur between the tool and the platform if it is not accounted for during the trajectory planning phase or the design of the cell. The location of the fixed part has to be done sensibly, even by using simulators to digitally replicate the job. Fig.6.21 depicts these two configurations. Another aspect to mind is the payload that has to be moved. The *mobile* tool requires the robot to be able to shift and orient it. In the *fixed* tool configuration, the robot has to carry the substrate and the build throughout the entire process.
The substrate can either be the print bed, the object on which the print is performed or a combination of the two. In any case, the knowledge of its geometry is important as much as the one of the tool, if not more. It is indeed the parent manifold that generates the initial slicing surface (planar or generic) that is repeated during the build

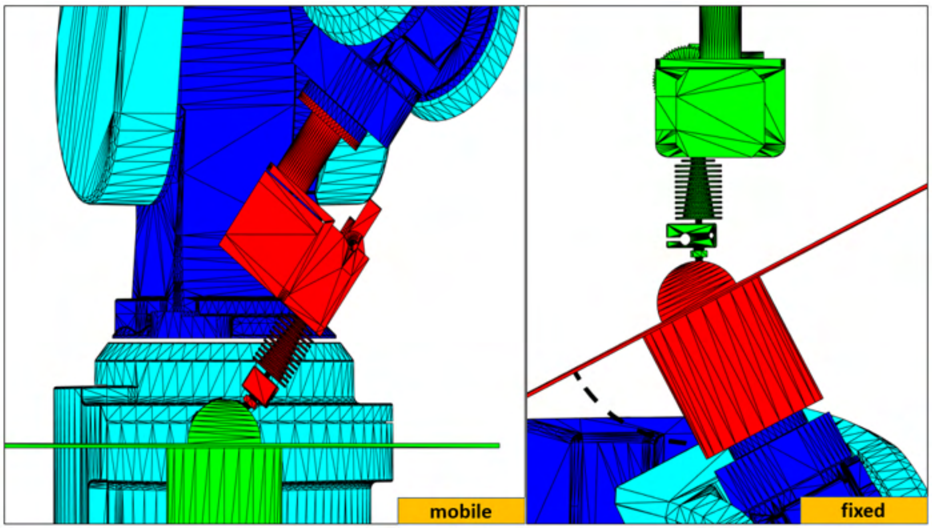


Figure 6.21: Possible implementation of a *mobile* and *fixed* tool cell.

or changed to comply with the nature of the object itself. The example in Fig.6.21 uses a substrate that is a half-cylinder fixed on a platform, on top of which a given geometry has to be built; so, the initial cutting plane has to be cylindrical.
The to-be-manufactured parts share a strong relationship with their substrate, so they should be considered as a single entity when opting for which cell type to implement. If the part to create is small, but the object on which has to be done is large and/or heavy (or part itself is bulky), a *fixed* tool cell cannot guarantee the expected result, if the robot is not able to handle the substrate (or substrate with the work-in-progress). Robots, especially PKMs, might feature a suitable payload capability, but lack the motion range to create complex 3D shapes. Hence, as a rule of thumb, the *fixed* tool has to be preferred unless the weight and size of the moved substrate cannot be handled by the system (Fig.6.22).

For small/medium build, a *fixed tool* cell allows to have a single robot inside the cell (therefore reducing the overall cost of the installation) attended not only by an AM tool, but also by SM tool (spindle), measuring devices (contact or contactless to get the actual printing surface) and standard robotic tools (for instance a gripper). *Mobile tool* cell, to be identical, requires robot capable to carry multiple tools or change them during the task.
For large build (for instance concrete wall printing), the *fixed tool* cell cannot be considered. The AM extruder, operating with molten material (or non-Newtonian fluids), could have a limited orienting range; therefore, large and complex 3D shapes could not be feasible at all with the sole AM tool. In these cases, it's better to use hybrid manufacturing technologies by adding and sequentially removing (or reshaping) the material.

Single robot cells could be improved by adding additional reprogrammable machines

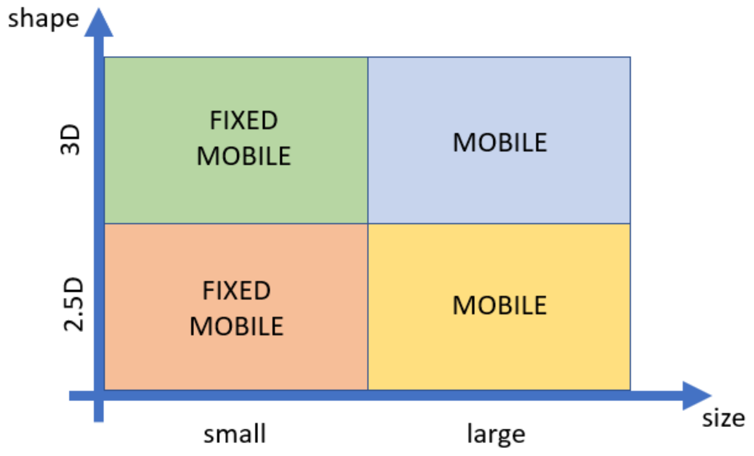


Figure 6.22: Cell type (*fixed or mobile tool*) selection according to size and shape of the substrate-part combination.

to carry out parallel tasks, include inserts and cooperate to the final goal.
A last plausible actor in the cell is an operator that intervenes to adjust the process or to actively contribute to the print. Due to the hazardous nature of the process (i.e. tool at a temperature greater than 200°C) that can cause serious damage even if contact prolongs for less than a second, the cell needs to feature all the safety measures necessary. *Fixed tool* cell can easily integrate a self-deploy protecting cover that shield the hot extruder from the reach of a person as soon as he or she enters the cell or upon crossing an unsafe region mark. The latter allows to pause the print only if mandated, letting the operator servicing other devices in the meantime.

Chapter 7

Industrial Case Studies

In this last chapter, what has been discussed and realized so far is applied to two actual industrial applications.

The first application presented is an example of adding material on a freeform surface (3D multi-directional printing). It sees an industrial manipulator depositing glue on a shoe upper to join it with its sole (section 7.1). After discussing the framework of the research, the devices that can be found in the cell are indicated pointing out their functions. This application uses the slicer in its general form, with some additional routines required to deal with the specific case at hand (such as referencing the path using available marks on the upper). Interaction between the TCP and the substrate is guaranteed by means of a load cells and force control (the one already available on the controller) that substitutes along the normal direction the position control. Trajectory planning is done for both *fixed* and *mobile tool* setup. Offline simulations with respective physical experimentation are reported.
This case study shows, moreover, how hardware/software can be generalized to other fields than 3D printing with relative no effort.

The second one (section 7.2) tackles the issues concerning subtractive applications and the need for a higher degree of automation of the final system. It reports the automatic generation of skull prosthetic devices for cranioplasty by means of SM and in particular milling (by means of a CNC machine). After reporting the framework of the activity, how it has been automatized is transcribed, pointing out the few interventions required by the medical personnel to get the physical counterpart of the digital skull reconstruction. This application requires a customized version of the *RSS* (using planes as cutting surfaces) presented in chapter 4 to menage the process; its stages are enumerated and detailed using practical examples taken from the research. Final specimens are shown.
This case study shows how the software implemented is not limited to its original goal and that can be used for HM (Hybrid manufacturing and *Layer shaping* discussed in 6.4.4).

7.1 Freeform deposition: Shoe Upper gluing

This project treats the engineering of the deposition of a glue filament, to join shoe upper and sole by means of an industrial robot. One of the main challenges of the proposed application is the target speed of 200 mm/s, which should be reached guaranteeing a proper contact force between the extruder tool and the shoe upper to be glued.
The adhesive exploited is specifically developed for filament extrusion systems and has to wet the shoe upper surface in order to guarantee a stable and durable bond. The intrinsic 3D nature of the upper surface requires to rethink the material spreading strategy found in analogues application, like first layer deposition in FDM-FFF 3D printing, which are based on the gap between the nozzle and the substrate. Glue deposition is an operation to some extent similar to roughing, since it requires to move the robot tool both on the side and on the bottom of the upper, depending on the shape of the shoe sole.
In this application, a six-dimensional path must be followed over a surface, whose geometrical description may suffer from a certain degree of uncertainty. A strictly geometrical approach would require the accurate knowledge of the geometry of the shoe upper and of its placement in the robotic cell, thus requiring faithful CAD/CAM models and calibration after every change of setup. The latter would be too time consuming and therefore unsuitable for customized mass production, for which all production phases need to be prepared in advance during the design phase of the specific shoe model.
To overcome this issue, the automatic procedure proposed relies on a force feedback control. The glue extrusion system, mounted on the robot, follows a tentative trajectory previously defined offline by means of a CAM software, which is then adjusted online through a force feedback control with constant reference force along the normal direction of the path.

7.1.1 Cell setup

The principal subsystems of the gluing cell proposed are a reprogrammable machine (i.e. industrial robot) and an extruder. As for the robotic system, the application has been developed on three commercial manipulators with 6 Dofs, commanded with a reference path for nominal positioning, and with a further feedback loop for the control of the contact force between the tool and the upper.
The machines adopted are the Mitsubishi (*RV-2F-Q*), Universal Robots (*UR3*) and a Techman robot (*TM5-700*). In the former case, an external load cell (model 1F-FS001-W200) is positioned between the end effector and the tool for glue deposition, whereas for the first cobot, the integrated force cell is used while for the second an external load cell is mounted on the wrist.
The extruder used is the one described in 2.1 able to withstand the 300°C required. It is either mounted on the robot (*mobile extruder*) or fixed inside the workspace (*fixed*

extruder). In both cases, crucial is the definition of the fixed position of, respectively, the upper and the tool in relationship with the other geometrical constraint (kinematics parameters, upper and tool dimensions) in order to carry out the gluing process without additional actuated axis to reposition the upper or the extruder.

7.1.2 Definition of the end-effector trajectories

The slicing software performs, as a first step, preprocessing to select only the portion of the upper where the glue will be laid, to reduce the computational burden for the generation of the end effector trajectory (cut process as discussed in section 4.1). This region is identified as the area of interaction between the upper and the sole in the final shoe. A graphical representation of the input and output of this step can be seen in Fig.7.1, representing the original model of the shoe upper and the selected portion where the glue has to be placed.

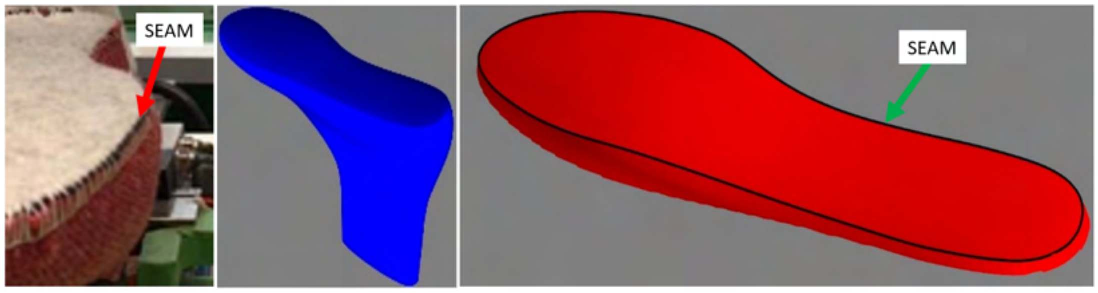


Figure 7.1: STL file of shoe upper and part of the upper where the glue has to be placed

The slicing algorithm uses the position of the seam as reference contour for the evaluation of the glue passes. This choice has been taken so that the path can be either reconstructed inside the software autonomously or it could be also input from the actual curve (using image processing or other measurement devices). Let's discuss here only the software reconstruction of the seam.
As it can be evinced from the upper shape (an example is reported in Fig.7.1, *left*), the seam corresponds to a variation of the normal vector field. This can be straightforwardly translated in founding the perimeter of those triangle whose normal vectors, n_i , distance from the vertical direction (n_{ideal}) of a threshold value.
The normal vector of a triangle is the cross product of the vector linking the first point to the second and the vector between the first and the third. The angular distance can be computed using:

$$\theta_i = \arccos(n_i \cdot n_{ideal})$$

provided that the vectors are normalized. The threshold value is estimated as the value that separates the two groups of points (the one of the top of the upper and the ones of the lateral surface). Hence those triangles that have normal vectors distant more than the threshold are excluded from the following steps. Fig.7.2 shows the θ trend that can

be used to get the discriminating value as well as an image showing the upper with the top and lateral normal vectors told apart.
The seams can be obtained imposing a threshold of 35°to singles out all the triangles within the contour. The actual point belonging to the contour can be found by first projecting the points on the X-Y plane and by exploiting alpha-shapes to get the hull. These points (belonging to the contour of the projection) serve as a base for the extrusion of a lateral surface along the Z axis. The intersection of this newly created surface and the partitioned upper yields the actual profile of the seam (Fig.7.2).

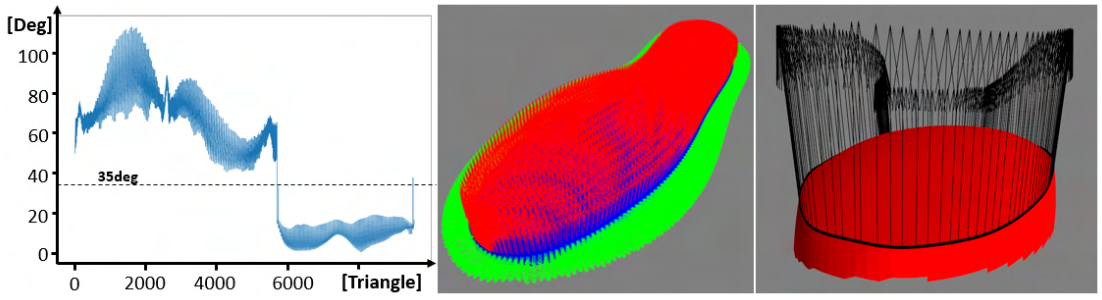


Figure 7.2: Seam evaluation algorithm.

The possibility to give birth to surfaces during the analysis of the shape can be exploited to get multiple concentric perimeters by scaling inward the contour. Infills can be generated using similar lateral surfaces but using different starting primitives. For instance, parallel lines can be used to fill the rear and front end with additional material. To do so, the projected contour obtained as for the previous cases needs to be intersected with user defined parallel planes (orientation, number and spacing). Lateral passes need to still rely on the contour, but this time the slicing surface changes. The cutting plane is obtained by lowering the seam (in the 3D space) by the requested quantity along the Z axis and by generating two child curves inward and outward. The meshed surface allows to compute the actual intersection path. Graphical results for both the lateral surface and infill are available in Fig.7.3.

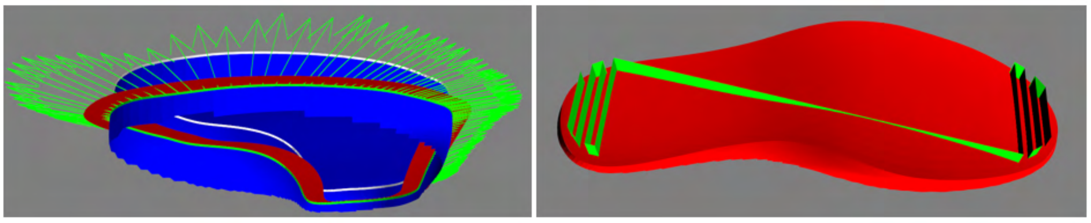


Figure 7.3: Lateral and infill surface generation.

The user can then customize the toolpath in post-processing by defining a spatial sampling along the curvilinear abscissa, or adjusting the orientation of the tool with respect to the surface normal, so as to investigate the best parameters to be adopted in the gluing process. This angular correction, within an acceptable range, could be

carried out automatically to make sure that none of the points lays outside the robot operational range.
One last crucial aspect to make sure before generating the machine instructions is to complete the definition of the local reference frame along the path (as discussed in 4.6). By default, the software imposes that the X unit vector of the local frame is pointing at the next point (correction is done to be sure to be working with orthogonal frames), the Y derives by imposing a right-handed coordinate frame.
The final toolpath is represented in Fig.7.4. The path for glue extrusion follows the seams, heel and toe.

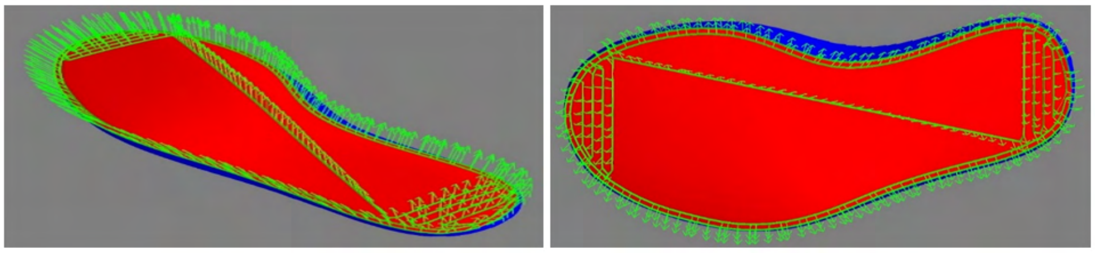


Figure 7.4: Toolpath generation from CAD model.

Fixed or Mobile tool

The robotic cell can be designed to have the extruder fixed in a given position of the workspace (in this case the upper is moved by the robot) or mounted on the EE (upper is unequivocally located in the cell).
In the case of mobile extruder (i.e. fastened to the wrist of the robot), the toolpath obtained from the slicer needs to referred to the robot's base coordinate frame, if not already, and convert the tool orientation with a suitable convention (RPY, quaternion or angle-axis). Information about the tool dimensions is stored in the controller, that it will use them to solve the kinematic problem.
In the case of fixed extruder, the path communicated to the robot is a collection of positions and orientations of the tool0 (i.e. center of the wrist) so that the actual profile extruded on the upper is the one expected. Hence, the path needs first to be described in the tool0 coordinate system knowing the geometry of the clamping mechanism and the upper. Given the fixed position of the tool, it is possible to define the corresponding configuration of the TCP (see section 4.6).
Graphical examples of both cases are reported in Fig.7.5; the mobile tool is simulated with a UR3 robot, while the fixed extruder with a TM5-700 for both top and side passes. A different definition of the pose of the robot for each point along the path will have generated different results. For instance, Rodrigues' formula leads, in the fixed tool cell, the base joint to rotate of about 180 degrees compared to the *tangential vector* depicted in Fig.7.5 for the side passes in which the J6 does the majority of the reorienting.

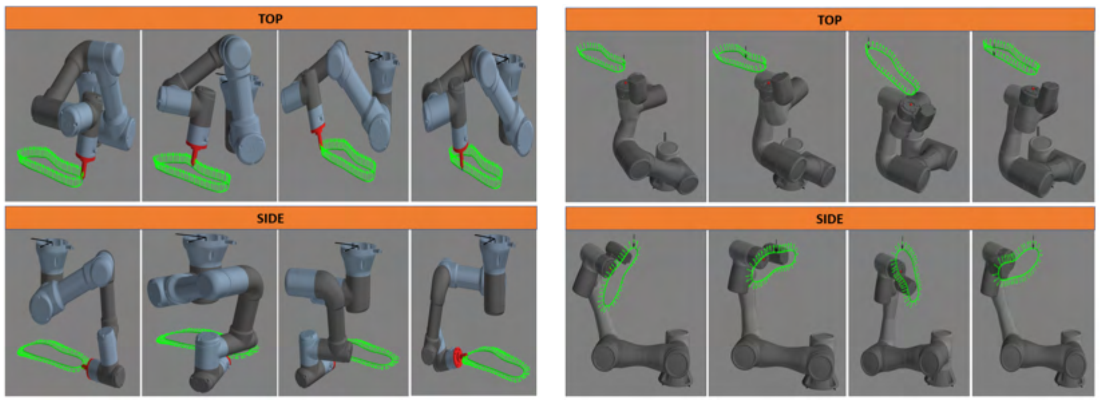


Figure 7.5: Finale toolpath in the case of mobile (*left*) or fixed (*right*) extruder.

7.1.3 Definition of the robot instructions

Here it's going to be addressed how the resulting path have been uploaded on the respective controller of each robot considered. Difference approaches were adopted to better comply to the built-in functionalities as discussed in chapter 3.

Mitsubishi The obtained references frames are used to define the orientation of the TCP (using RPY convection) that prevents a hazardous twisting of the cables directed to the extruder and load cell and of the filament.
All the possible control strategies have been simulated before proceeding to the practical tests; the one giving good result was the spline interpolation from a file (*.csv*) storing the points (*real time* was omitted to avoid requiring additional hardware in the robotic cell). It allows to rapidly redefine the trapezoidal speed profile along the abscissa.

Universal Robot In this case the orientation of the TCP is expressed using the angle-axis convection.
The slicer, running in Python, is customized with a function that generates a *.txt* file containing the code that needs to be uploaded on the controller. It does not only store the movements, but also the instructions to activate the force control and synchronize the extruder over TCP/IP.

Techman Robot This is the only case that the solution with fixed extruder has been tested.
The tool0 path is uploaded on the controller where it is stored into a local variable to be successively elaborated, transformed into arrays of floats of correct dimension and then saved as a global variable for further use. The points are then interpolated with straight lines with blends.

7.1.4 Mobile extruder: Experimental tests

The robot use is a Mitsubishi 6 Dofs robot. Experimental tests are carried out in the speed range from 50 [mm/s] to 200 [mm/s], and control parameters are firstly set up on a linear trajectory going from the shoe heel to the tip. After this set-up phase, the same control parameters are exploited for the complete toolpath following the shape of the shoe upper. The tuning of gain parameters is carried out by using a feeler instead of the glue extruder, with the load cells interposed between it and the robot end effector. Due to the low weight of the feeler (i.e. 0.05 [kg]), which is built with ABS with an FDM process, the contact force can be assimilated to the force measured by the load cell. These set up tests allow to measure the contact force and to assess the efficacy of the control for different linear speeds. In the case of Mitsubishi robot, the final parameters adopted are Gain = 20 [μm/N] and Damping coefficient = 0.1 [N/(mm/s)].
Fig.7.6 reports the 3D trajectory followed on the shoe upper and the corresponding contact force, for the speed case of 50 [mm/s]. During the experimental tests the robot is firstly controlled with the position control mode to reach a starting point close to the shoe's surface. At this point the force control mode is activated, while the tool keeps on following the predefined trajectory. The performances of the system for different

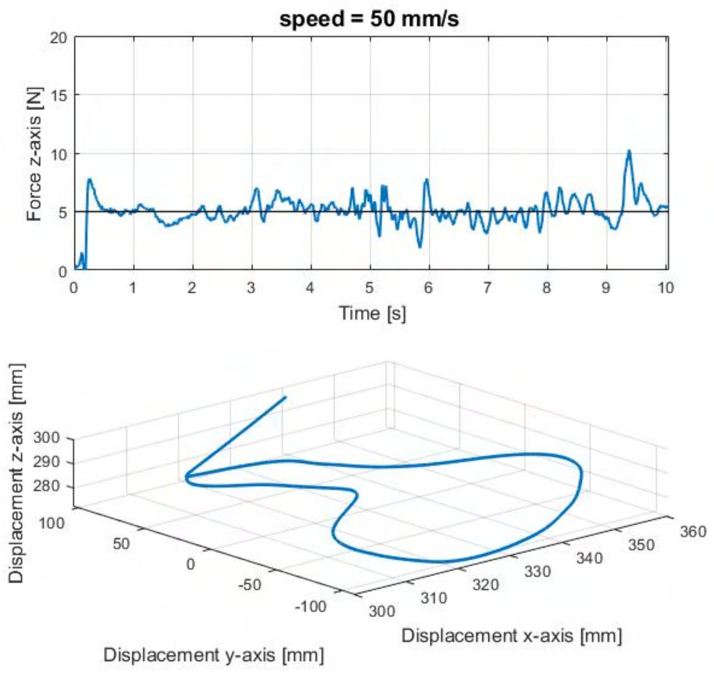


Figure 7.6: Path followed and contact force between the feeler and the shoe upper.

linear speeds of the tool are evaluated by considering the average and the standard deviation of the contact force, under the hypothesis that the contact force distribution can be represented as a Gaussian distribution. Fig.7.7 reports the values of average contact force and the corresponding standard deviations for all the tests carried out on the same path of Fig.7.6. When the statistical minimum ($F_m - 3\sigma$) is higher than zero, no contact loss is occurring. Even if the target mean force is the same for all the tests (i.e. 5 N) it is possible to observe that the value of the average contact force increases as a function of speeds, due to some overshoot occurring due to the force control loop. The standard deviation also increases with speed, and the statistical minimum is lower than zero already at 100 [mm/s].

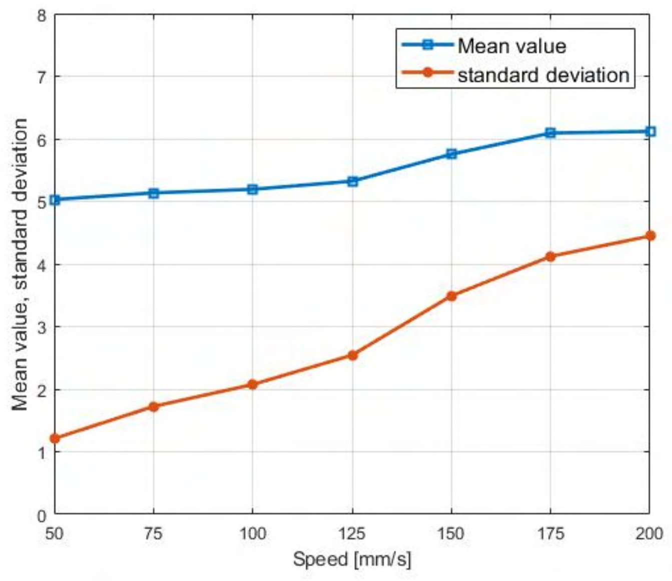


Figure 7.7: Contact force average and standard deviation as a function of the end effector linear speed.

Glue deposition tests

The optimal glue flow required is evaluated by preliminary experiments, and then adopted for the following deposition tests. The desired glue flow is achieved, for each of the different linear speeds of the end effector tested, by changing the speed of the extruder motor. Tests with two kinds of upper with different stiffness values are performed, defined as hard upper and soft upper. Experiments are carried out in both cases by performing two complete closed trajectories with different offset from the seams.
During the tests on soft upper a good glue deposition, with good adhesion, was observed up to the speed of 200 [mm/s], even if even if the contact force behavior showed some detachment of the extruding tool. The fine control of the value of contact pressure is not a strict requirement in glue deposition; some detachments can be tolerated, provided that they occur for a limited path length.
On the other hand, the tests with hard upper, carried out with the same parameters of those on the soft upper (default gains set by the manufacturer, UR3, and advised against their modification), showed a worse adhesion of the glue. Also, in this second case a good glue deposition is achieved up to 150 [mm/s], but a worsening in deposition

is observed at the speed of 200 [mm/s], with the glue not continuously deposited especially when the contact force increases. This is due to the fact that the force control performance gets worse due to the lower compliance of the upper.

7.1.5 Fixed extruder: Experimental tests

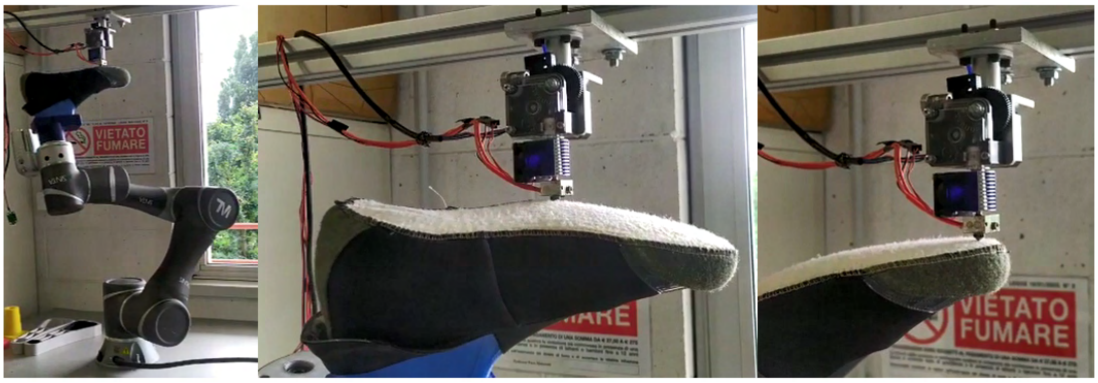


Figure 7.8: First lunch of the *fixed extruder* cell with the upper approaching the nozzle.

Trials on this type of cell configuration have yet to be performed. In fig.7.8, the first dry run is reported showing the upper meeting the fixed nozzle.

7.2 Automatic Subtraction: Skull prosthetic implants

This project develops in the field of automation in the medical-surgical sector. This research aims to automate the process that realizes prosthetic devices for cranioplasty after brain tumor removal, putting stress on the possibility to create the opercula during the surgery itself (i.e. run-time) and so alleviate the work that surgeons has to do during the operation.
Generally, the surgeon, before performing surgery and placing the prosthetic device, delineates a plan of the medical operation that gives an idea on where and how it has to be managed. A CT (Computed tomography) is done and then through a image processing software data are processed and a 3D model of the expected operculum is recreated. Later the surgery is performed, but a significant geometrical uncertainty can exist between the part of the skull removed and the one expected during the preliminary analysis. To avoid this problem the operculum is manufactured, ahead of times, considering a larger surface so that by retouching the prosthesis (and sometimes the skull itself if needed) it fits the hole. Results however is not very precise and the risk of making mistakes is high.
In order to avoid these issues, a project is born with the idea to manufacture at run-time the operculum relying on the actual incision of the skull. By processing a 3D scan of the skull, a digital model of the prosthetic device is generated. This file, in a specific format (i.e. STL), is used to make the machine code, that will finally be

uploaded on the chosen machine to create the prosthesis. Moreover, the implant has to be realized in a material that allows the patient to undergo radiotherapy without the need to remove the prosthesis from the cranium.
In this context, it was asked to develop the procedure that allows to machine (by means of a CNC milling machine) the final operculum starting from a digital file in the time required by surgeon to complete the operation.
The activity is reported in the following sections:

1. *Craniotomy and Cranioplasty*: aspects of the medical operation are researched to contextualize the manufacturing process;

2. *Operculum analysis*: it is executed in order to understand if a standardization of starting blocks is feasible;

3. *Process definition and toolpath generation*: the steps that the medical personnel has to follow to obtain the operculum are translated into the software counterpart;

4. *Testing*: obtained machine instructions (G-code), that allows to get the final prosthesis, are run to get first specimens.

7.2.1 Craniotomy and Cranioplasty

Craniotomy is defined as the surgical excision of a part of the bone from the skull that uncover the brain. This is performed every time that the surgeon has to access the brain itself or relieve intercranial pressure.*Cranioplasty* is defined as the surgical restoration of a bone defect in the skull.
Brain surgery to remove tumors involving craniotomy usually takes 4-6 hours.
Here the steps related to the bone excision and implant placement are analyzed.

The surgeon, before doing the operation, analyze the case to examine and does a series of evaluations based on available information. A plastic surgeon might intervene to design a suitable incision in order to carry out the surgery in the best possible way, both aesthetically and functionally. He or she draws the line (on the shaved scalp) where the cut has to be done with a scalpel. The incision is followed by a cervical sweep that is a separation of two anatomic layer, exposing the bone. The skull is demolished following the contour traced using the to-be-used prosthesis. This initial step can be seen in Fig.7.9.

The brain surgeon can now access the aimed area to carry out the surgical operation. The prosthesis used for the incision might require to be reshaped to fit the actual hole resulting after the incision with surgical tools. Available prosthetic devices are provided with mooring holes already preformed. Surgeons may decide to add others during the modeling phase using a blade mill. The prosthesis has to be as much as possible integral with the skull, if it does not happen this neutralizes the colonization process. To get this result mooring wires are used. In Fig.7.10 these steps are depicted. Once the operculum is positioned it can happen that does not fit perfectly with the bone or it

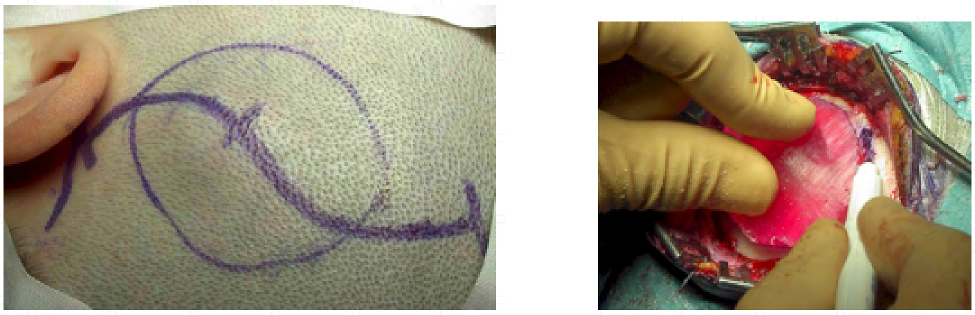


Figure 7.9: Tracks to follow in the skin incision and in the skull removal [115]

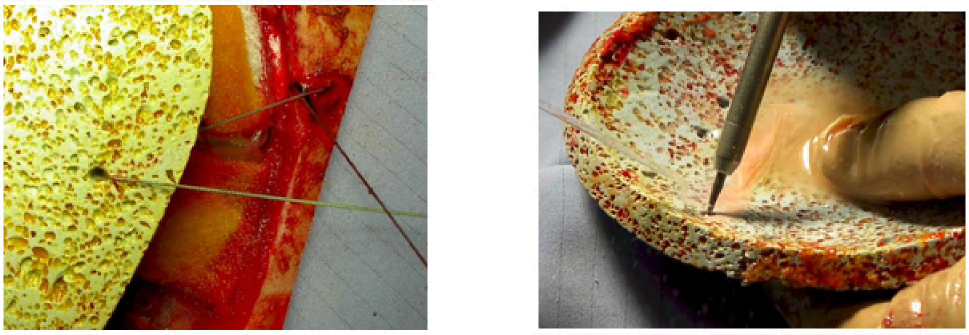


Figure 7.10: Mooring and realization of the holes on the prosthetic device [115]

stays lightly lifted. To avoid these problems finishing touch with the mill are done paying attention not to break the wires, suspensions and the prosthesis itself. At the end of the surgery, if necessary, depending on the case, a skin flap suture is carried out and then the bandage is applied.

All of these activities requires high precision (because a technical error, such as a hole in the skull wider than the filling operculum, is difficult to solve) and sterile rooms (environment, tools for the operation and material that will be inserted have to be perfectly sterilized, since a strictly contact with the skin increase highly the possibility of infection).

7.2.2 Operculum analysis

The prostheses geometry has to be studied in order to outline the most suitable manufacturing process and to reduce the machining time (that has to be completed in around 3 hours). A family of starting blocks (workpieces) could be manufactured with different sizes and curvatures to minimize the volume of material that needs to be cut out.
Therefore, 25 opercula (deriving from 5 skulls in equal measure), representative of the overall population of prosthesis according to the medical experts involved in the activity, are studied to generate such class.
The study has been subdivided in 5 phases. In the first phase, the files, referenced in the original skull coordinate system, are oriented and laid down to facilitate the anal-

ysis (Fig.7.11). In the second one, they are converted in a specific file format in order to be processed by a suitable software (from *.stl* files to *.ipt*). In the third phase, they are drafted along the X and Y axes to measure the curvatures and dimensions using sections (Fig.7.12).

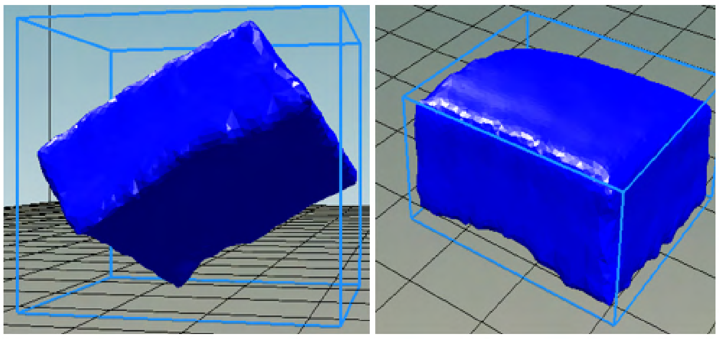


Figure 7.11: Reorientation

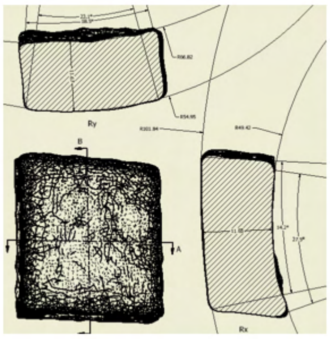


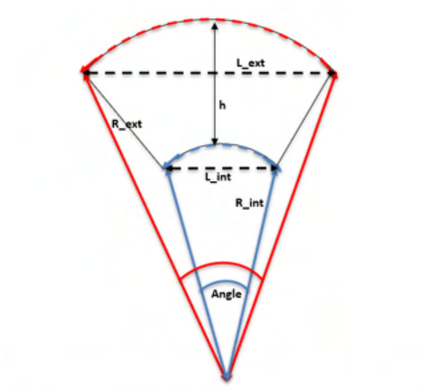
Figure 7.12: Drafting

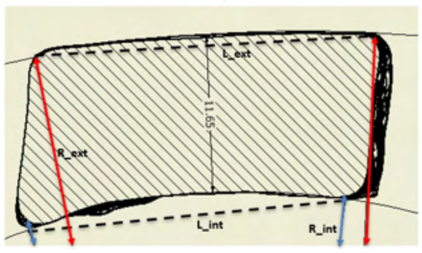
In the fourth, data is recorded and classified according to the determined curvatures. The analysis is done considering the radii, lengths and thickness along the X and Y direction for both the external and internal surfaces. Results are reported in table 7.1. Both internal and external radius go from a flat surface to around 18 [mm], where the most frequent value is around 50 [mm]. The thicknesses vary from 6 to 12 [mm] and the dimensions from a minimum of 17 [mm] to a maximum of 50 [mm].

Table 7.1: Radius, thickness and dimension statistical results

	mean	max	mode	SD
Rx_ext	82.4	196	50	38.78
Rx_int	85.08	500	47	93.532
Ry_ext	78.8	214	50	45.038
Ry_int	100	2848	43	554.9
hx	9.8	12	9	1.5
hy	10	12	12	1.5

	mean	max	mode	min
Lx_ext	29.68	45	30	19
Lx_int	30.76	47	31	20
Ly_ext	27.6	38	25	19
Ly_int	28.12	50	32	17





From the analyzed data, a possible classification is obtained by splitting the opercula world in two families: a group in which there are those samples that have the external and internal surface characterized by curvature radii, in both the X and Y direction, less or equal to their respective mean values (external: 80mm for both directions; internal: 80mm for X and 100mm for Y); a second group, enclosing the remaining samples having curvatures greater than the means.
In the fifth and last phase, the prostheses are placed inside the workpieces engineered in the previous step to validate the result. An example of this procedure is shown in Fig.7.13.

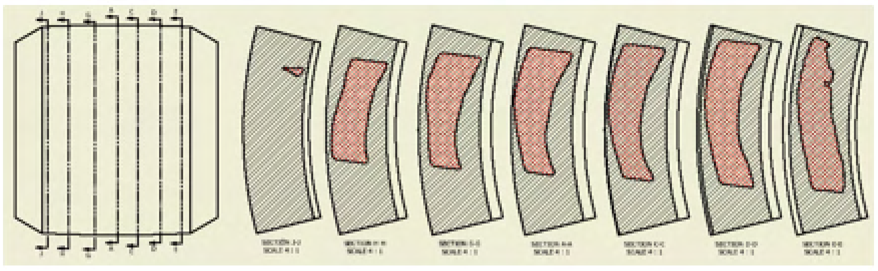


Figure 7.13: Example of samples contained in a possible starting block

Conclusions

Although feasible to start with pre-shaped workpieces, according to the analysis conducted, the final system will obtain the operculum, starting from a parallelepiped block with flat surfaces, by machining both the curvatures. This reduces the number of machining operations the block undergoes to (that might compromise the sterility of the operculum) before reaching the surgery room, although increasing the lead time during the surgical operation. The only classification that can be adopted involves the workpiece thickness to reduce roughing time.

7.2.3 Process definition

The manufacturing process has to be engineered to minimize the cooperation with medical personnel, not necessarily qualified or trained to use a CNC milling machine and relative CAM software. The automated process has to receive in input the .STL file of the part to be made and return the operculum requiring simple post processing activities. An overview of its main steps is here discussed.
The hole left by the craniotomy is scanned and the data is matched with the topography of patient skull taken before the surgery. A digital copy of the bone part is therefore created and saved in .STL file. This image processing software has been engineered by a partner company specialized in the field. The 3D model obtained is still referenced in the coordinate system of the skull scan, so it has to be oriented to meet the already discussed specifics of the activity. The manifold has to be then placed inside the

workpiece so that: there is enough room for the manufacturing process; scraps are reduced (the same workpiece can be used for multiple operations) and the clamping mechanism (of the block) is in the lower-center region. The path of the center of the tool needs to be computed knowing, from the opercula analysis, that inner and outer surfaces (used to to define the R_int and R_ext in the previous subsection) needs to be machined one side at a time. The resulting G-code is uploaded on a at least 4 DoFs machine (for this application a 5 axis CNC milling machine, *Pocket NC*, has been used) that carries out the generation of the physical counterpart. Once it has finished, the prosthesis will be ready and it could be removed, sterilized and used by the surgeon. The overall process is depicted in Fig.7.14.

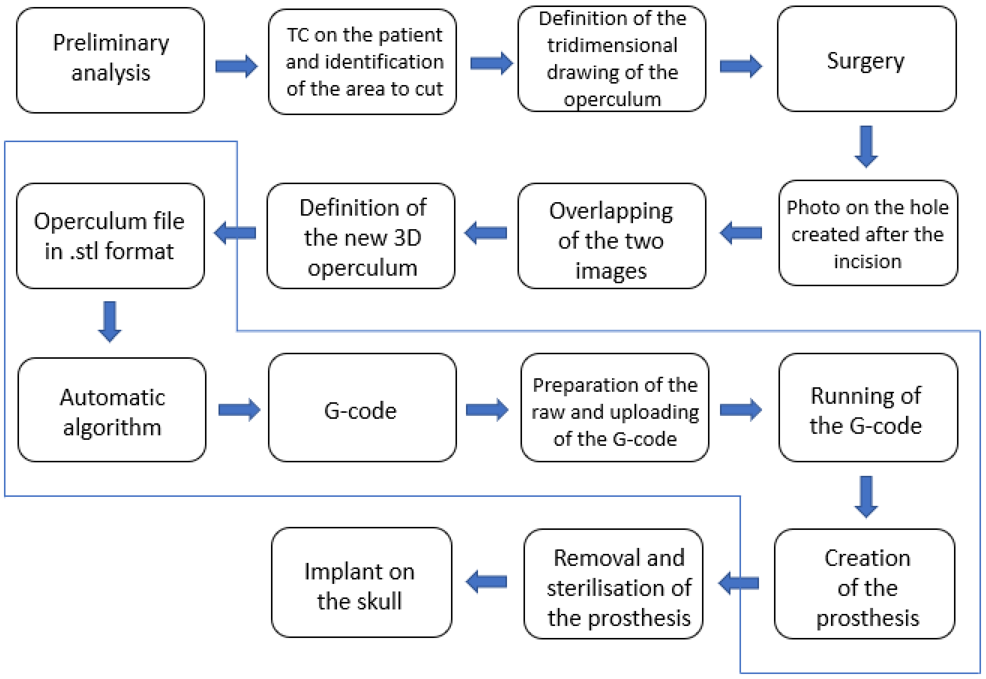

Preliminary analysis
TC on the patient and identification of the area to cut
Definition of the tridimensional drawing of the operculum
Surgery
Photo on the hole created after the incision
Overlapping of the two images
Definition of the new 3D operculum
Operculum file in .stl format
Automatic algorithm
G-code
Preparation of the raw and uploading of the G-code
Running of the G-code
Creation of the prosthesis
Removal and sterilisation of the prosthesis
Implant on the skull

Figure 7.14: Automatic procedure.

Let's contextualize this overview, by addressing specifically how the medical staff will interact with the system, how the material removal will be done and how the software will menage all of this.

Surgeon point of view: (or from the one of a qualified colleague at least). The procedure consists in taking a picture of the interested area following the on-video instructions. The software outputs on the screen the workpiece to retrieve from the ones available in stock and how to properly install it on the machine. Once completed, burs have to be removed with tools already available in the surgery room.

Manufacturing point of view: the material subtraction is carried out in three phases. The first one machines the inner side (that will also be referenced as *side A*), the second the outer side (or *side B*) and the third one the contour (or *side C*). The overall CNC machine operations are here summarized:

1. **Home position**: The spindle is brought to the machine reference position;
2. ***side A* machining**: While the tool is at home, the workpiece platform is oriented to show *side A* to the milling cutter. The TCP is positioned in the center of the region where the block will be chiseled, coincident with the origin of a local reference frame. Points in the G-code are referenced to this coordinate system;
3. **Tool home**: Bit exits the workpiece to rotate the platform;
4. ***side B* machining**: Steps carried out for the inner side are repeated for the outer being careful to leave an anchoring strip holding the operculum to the rest of the block;
5. ***side C* machining**: The contour is machines separating the prosthesis from the block;
6. **Home position**: The spindle is returns in the home position to dwell.

Software point of view: The software is based on algorithms developed for 3D printing; the only difference is the machine expects Z- in the subtraction direction instead of Z+ in the growing direction. Therefore, the toolpath generation is carried out in the software coordinate system and at the end rotated ($Rot(X, -\frac{\pi}{2})$) to meet the machine convention. This leads to consider the side load parallel to the software Z axis and the depth of cut to the Y axis. See Fig.7.15 for a graphical representation of such concepts. From here on out, if not differently stated, the software conventions are adopted.

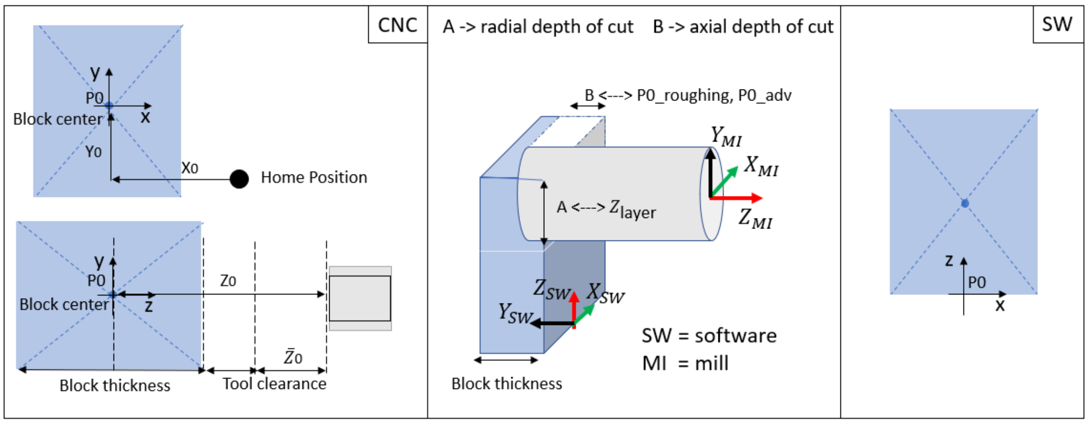


Figure 7.15: Reference frames for the machine (CNC) and the software SW.

The first steps revolve around the orientation of .STL manifold so that it fits the raw block geometry, it reduces the volume to take away (scraps formed from the workpiece).

A bounding box is generated to simplify the placing procedure; during this phase an anchoring strip is added around the box that will serve as holding material during the machining of both lateral sides (it will be removed during the *side C*) to avoid a premature separation from the workpiece. The final position inside the workpiece is obtained also considering a restricted region close to the block gripping mechanism (Fig.7.16 summarizes this concepts).

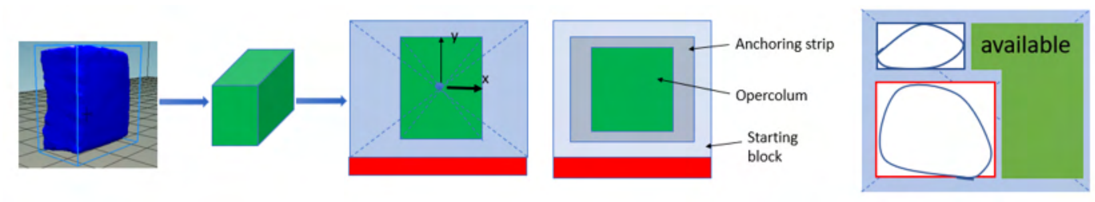


Figure 7.16: Machining region identification and placement inside the workpiece.

The subtraction of material starts out by cutting the augmented 3D model (considering an additional region at the top and the bottom of height *Z_gap_block*) with plane spaced by the side load (i.e. layer height). It returns a contour per each layer in the XY plane. Since we are interested in the subtraction of the workpiece and object, a box around the perimeter is added. Its dimensions are computed by adding a quantity, *X_gap_block*, to the maximum length of the prosthesis along the X dimension and considering the thickness of the workpiece (*Y_block*). The area between these two polygons has to be removed. The next step is to split the intersection polygon in two halves corresponding to the accessible surfaces from the *side A* or *side B*. For each of these two regions, roughing (*p0_roughing*) and fine (*p_adv*) passes have to be determined while leaving untouched the anchor (*p0_adv*).
The resulting segments are connected to form the final path per layer, considering a

- Miller diameter: d_m [mm]
- Cut movement speed: Feed1 [mm/min]
- Free movement speed: Feed0 [mm/min]
- Initial layer height: h_layer0
- Layer height: h_layer

- Safe distance tool-workpiece: tool_clearance
- Dimension of the strip per layer: p0_adv
- Roughing pass: p_roughing
- Thin pass: p_adv

- External margin along X: X_gap_block
- External margin along Z: Z_gap_block
- Starting block thickness X: X_block
- Starting block thickness Y: Y_block

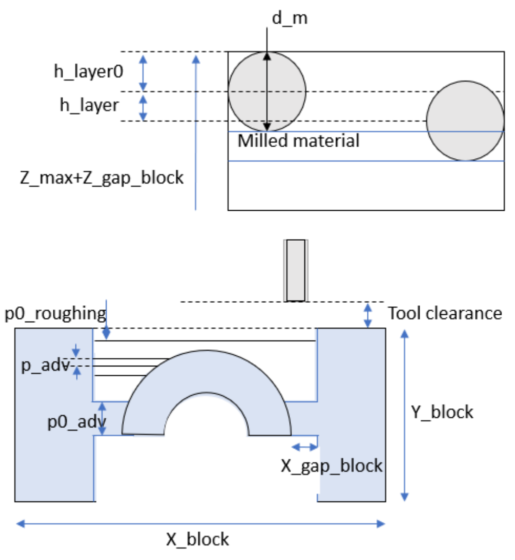


Figure 7.17: Parameters used to master the process.

gap (*tool_clearance*) from the workpiece when moving without material removal. All the *side A* half paths are united to form a unique curve and the same goes for the *side B*. Finally, the contour is evaluated to separate the prosthesis from the workpiece. The process parameters, defined in an excel file (modifiable only by qualified users), are accessible to the code to generate the toolpath. They are listed in Fig.7.17.

7.2.4 Toolpath generation algorithm

The algorithm can be subdivided into seven phases:

1. Reading of the 3D file (.stl) and software initialization;
2. Setting of the working parameters;
3. Positioning of the piece inside the work volume;
4. Slicing;
5. Distinction of the side A, B and Contour for each layer;
6. Generation of the trajectories;
7. Post-processing.

It has been developed in the *Matlab* environment; it's structured to have a *"Main"* function calling scripts (identified with the name "Main_<task_name>") that use functions defined in dedicated sub-directories. The figure 7.18 shows the general structure.

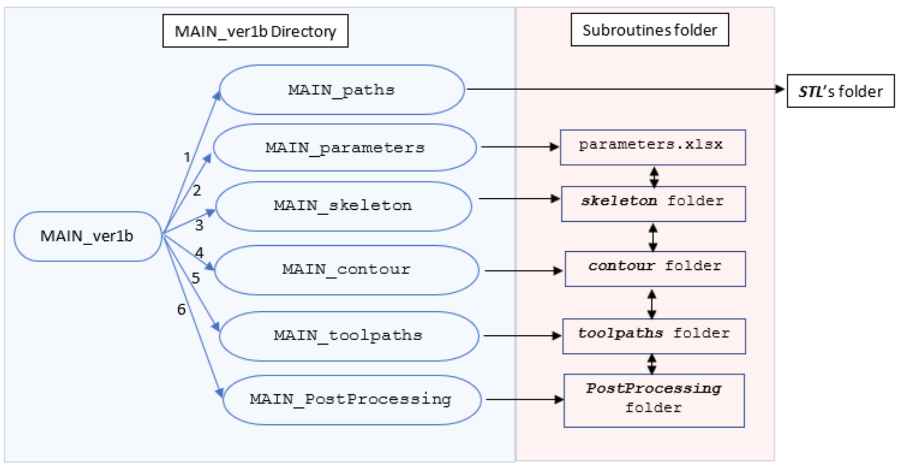


Figure 7.18: Code structure.

Reading of the 3D file (.stl) and software initialization This part is subdivided in the three different phases: reading of the file; non-manifold test; Mesh reconstruction. The general steps have been already presented in section 4.1. To this specific case, the

Table 7.2: CNC manufacturer advised parameters

	Speed (RPM)	Feed per tooth (mm)	Side Load	Depth of Cut
Hard Plastic	8500	0.0254	50%	80%
Soft Plastic	8500	0.0381	60%	70%

first two phases might be time consuming due to the file size of the meshes. This problem is solved removing redefined points. Possible errors in the file (non-manifold volumes), that will prevent from the reconstruction of the perimeters, are solved using the function *ConvexHull* in *Matlab*. This approximation is acceptable only because the modification caused are in compliance with the specifics of the activity.

Setting of working parameters This part is devoted to initiate all the parameters necessary for the definition of the toolpath. The main objective here is to define all the input that should have been set by the operator (i.e. surgeon), who, in this application, is not supposed to know how the chip removal works. See Fig.7.17 for the variables that can be set in the *.xlsx* file.
As concern the identification of the process parameters, the material to be machined is plastic, therefore, following the advisement of the machine manufacturer, for the initial test the parameters should be evaluated using the values found in table 7.2. From literature, it is also known that optimal performance and good surface results are obtained using single cutter tools. The table feed is computed as:

$$Feed\ [mm/min] = feed\_per\_tooth\ [mm] \times \#\_teeth \times Speed\ [RPM]$$

so, for instance, in case of "soft plastic" the feed is equal to 323.85 mm/min (single flute bit).
The mill cutter diameter is 3.175 [mm] (1/8 inch); therefore, the axial and radial depths of cut can be computed. For example, in the case of "hard plastic", *p_adv* is 2.54 [mm] while the *h_layer* is 1.5875 [mm].

Positioning of the piece inside the work volume This part of the algorithm is subdivided in four phases that are the following:

1. Orienting of the operculum;
2. Laying down of the operculum;
3. Scrap minimization in XY plane;
4. Scrap minimization in XZ plane.

Recalling that reference system of the opercula imported is positioned in the center of the rectangle circumscribing the projection in the XY plane and at the minimum Z, the original object is projected in the XZ plane and the contour is approximated with the convex hull. It is then sliced with planes to get a sampled version of it. If the middle point of the segments obtained are linearly interpolated (linear regression), the orientation of the object can be estimated. Finally, the line is used to compute the angle to rotate the object around the Y axis. This operation is repeated for the XY and YZ plane three time each to make sure to have reached the desired position. It is depicted in Fig.7.19.

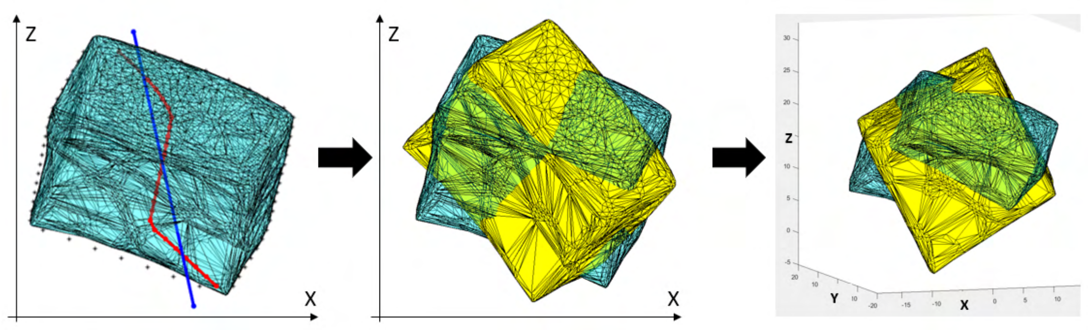


Figure 7.19: Orienting of the operculum: linear regression (blue line) in the XZ plane (*left*); relative rotation (*center*); final result of the first phase (*right*).

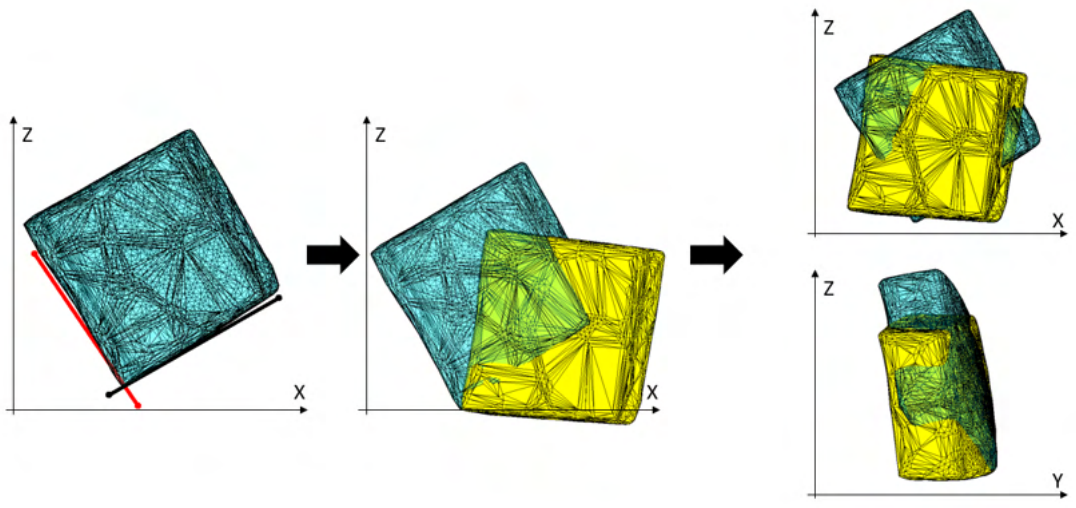


Figure 7.20: Laying down of the operculum: estimation of the local X and Z unit vectors (*left*); relative rotation (*center*); final result of the second phase (*right*).

As can be seen from Fig.7.19, the object at this point might not be necessarily laid down so this is taken care off prior to minimize the volume to remove. As did previously, the XZ projection is sliced, but this time to estimate orientated bounded box, at least the two lower sides. They are then used to rotate the object to align with the X or Z unit vector (the selection is done depending on the original orientation). The steps are then carried out for the YZ plane. Both are repeated for a maximum of 10 times or until no significant variation is detected from the previous case. It is shown in Fig.7.20.

Once laid, the operculum lastly undergoes to the same positioning procedure discussed in section 4.2 to make sure that the 3D model is properly fitted in the workpiece. The transition from the previous phase to the final position is reported in Fig.7.21.

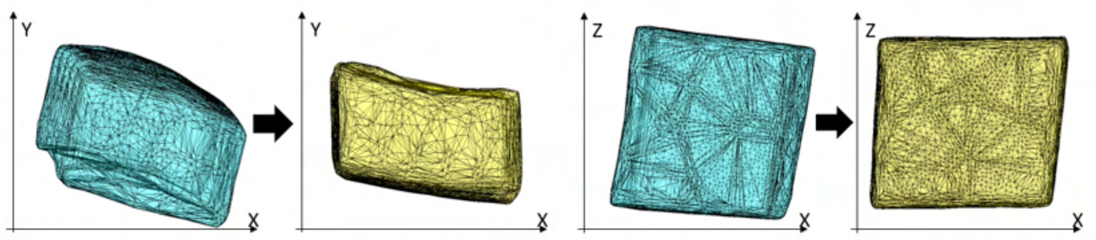


Figure 7.21: Scrap minimization: final configuration of the prosthesis.

Slicing Layers are defined considering an additional value, called *Z_gap_block* to the maximum (and the minimum) value of the Z of the operculum. This value is added to create a margin at the top and bottom of the operculum during the contouring phase and also to prevent the milling cutter to work the operculum itself. Moreover, it is included because, when the contour defined for the *sideC* is executed, the bit does not find a very thick block to work that would ruin or break it.
The first pass is done lowering the resulting maximum height by *h_layer*0 corresponding to half of the milling cutter diameter *d_m*; therefore the *Z_gap_block* should be greater than *d_m*. The remaining radial depths of cut (in this case the layer height) are equal to *h_layer*. The 3D object is sliced using the planes defined by these heights to evaluate the contours. The algorithm used is the one for conventional slicing (i.e plane-triangle intersection combined with loop closure). Results of the yielded perimeters are shown in Fig.7.22.

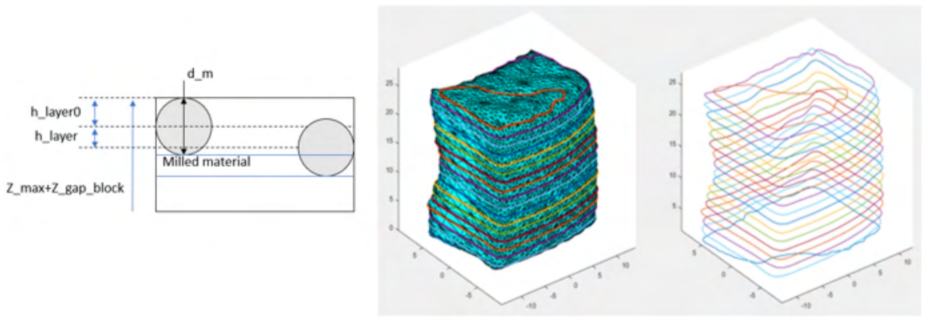


Figure 7.22: Closed profiles for one operculum.

Distinction of the side A, B and C for each layer In this phase it is used an algorithm to determine the normal to every segment of each contour. In a trivial way, given a segment, then it is known its unit vector that, if multiplied (cross product) with the one exiting the plane, returns the normal vector of the curve. At the connection of two neighboring segments, the normal can be approximated with the mean value of the two. If then the Y and X components are feed to a *atan2* function, the normal vector can be represented with its angular position instead. If the angle overlaps the first or second quadrant then the segment will belong to *side A* otherwise *side B*, as depicted in Fig.7.23. The perimeters can be split in the two sides. The *side A* is

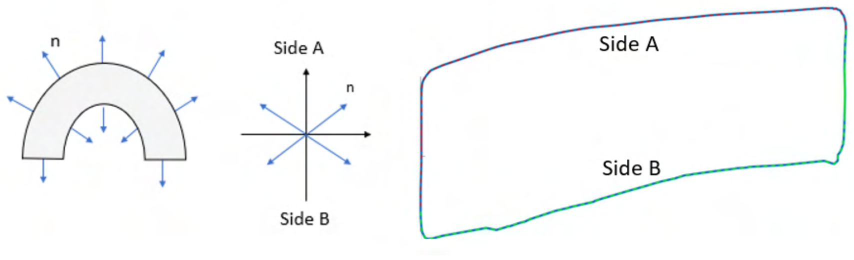


Figure 7.23: Quadrant subdivision.

rigidly moved in order to have the Y values negative, instead the *side B* is mapped (using transformation matrix) to have the Y negative. Subsequently, the lateral anchor (*X_gap_block*) is added, in this way the operculum will not detach from the workpiece prior to the machining of the *side C*. See Fig.7.24.

Side C on the other end requires side distinction, lower holding strip generation and

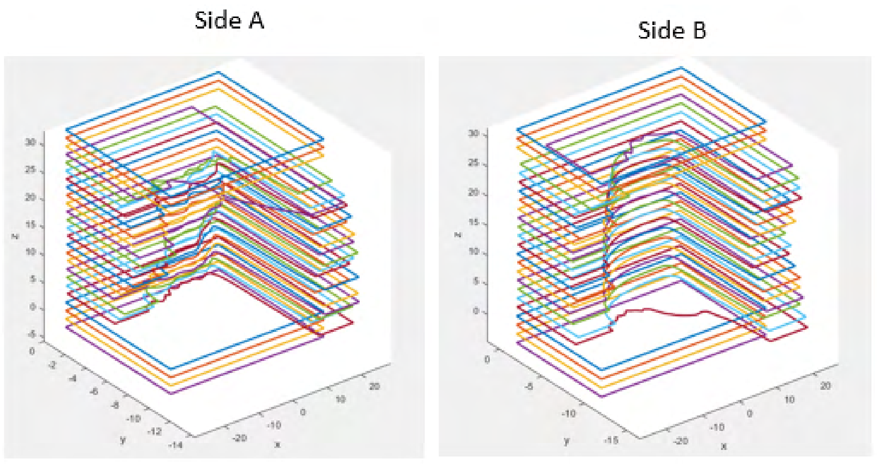


Figure 7.24: Side A and B.

definition of the contour. It is obtained looking at the projection in the XZ plane of the 3D model; the exact profile of the contour is obtained using alpha shapes boundary detection.

Concerning the lower margin, a contour that follows the local profile of the operculum is created so to have a length equal to the one of the piece, as shown in Fig.7.25. This adjustment will allow to the operculum not to fall when the final milling of the contour will be done and at the same time simplify the prosthesis removal by the surgeon from the workpiece.

Finally, the path, that the TCP will follow, is constructed using the projected contour, its normal unit vectors and the bit radius. See Fig.7.26 for a representation of the final result.

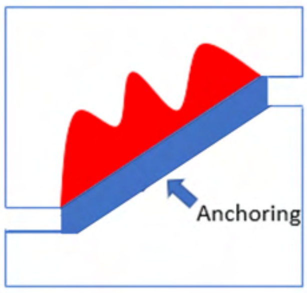


Figure 7.25: Lower anchor.

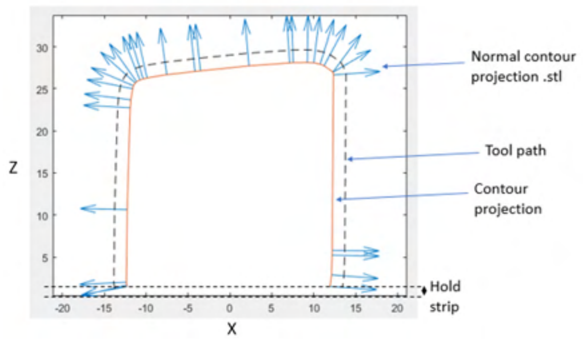


Figure 7.26: Definition of the side C.

Generation of the trajectories This penultimate part allows the generation of the actual toolpath and it is characterized by the two phases: determination of the characteristic points and their interconnections.
The first phase is important in order to guarantee zones of anchoring, tool clearance and roughing, instead in the second one it is relevant to avoid performing passes that could unintentionally mill the piece.
A series of parallel planes whose distances depends on the type of pass are generated. In all that zone where the passes do not intersect the operculum, roughing is performed (allowing a great time reduction); afterwards, where the toolpath begins to intersect the operculum, a smoother pass is implemented. The waypoints are obtained evaluating the distance of the points belonging to the contour (i.e. a polygon) from these planes. Indeed, if the sign of the product of such distance for one segment gives a positive result, it means that it does not pass through the plane and so it is not considered. Vice versa, if it is negative there is an intersection between the segment and the plane.
Let's discuss how the toolpath is generated among these waypoints. Conceptually the function starts from the first point that is seen and it positions itself at a certain distance (*tool_clearance*) from the workpiece. It then finds the closest point and joins them with a line.
The nearest point is looked for either on the same pass on the next one. For instance, starting from an external point of the *contour_sideA*, if no pass has been generated yet, the closest point to the one entering the workpiece lays on the same plane. Otherwise the closest one is searched along the axial direction. This creates an oscillating motion left to right and vice versa that can be seen in Fig.7.27. When the algorithm arrives to define the trajectories close to the operculum, it follows the same strategy, but this time it might occur that the point found implies the milling of part of the operculum. Therefore, the milling cutter is retracted, and it does an external transition that jumps

over the operculum zone to reenter the workpiece to reach the point found previously. Fig.7.28 reports an example of the final trajectory.

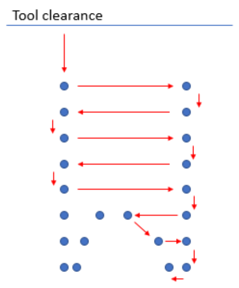


Figure 7.27: Toolpath generation logic.

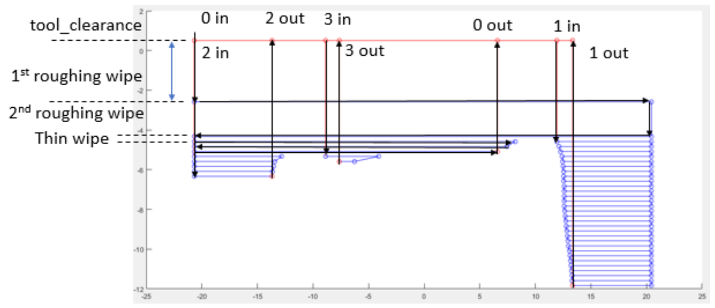


Figure 7.28: Example of trajectories obtained for one side.

Aiming to finish the operculum surface and to reduce the quantity of residual burr at the end of the job, a last pass joins the points of the contour following the surface of the operculum. Fig.7.29 shows the overall result for a single layer highlighting where the bit will enter to machine *side C*. It also reports the original manifold surrounded by the path necessary to cut it out from the workpiece.

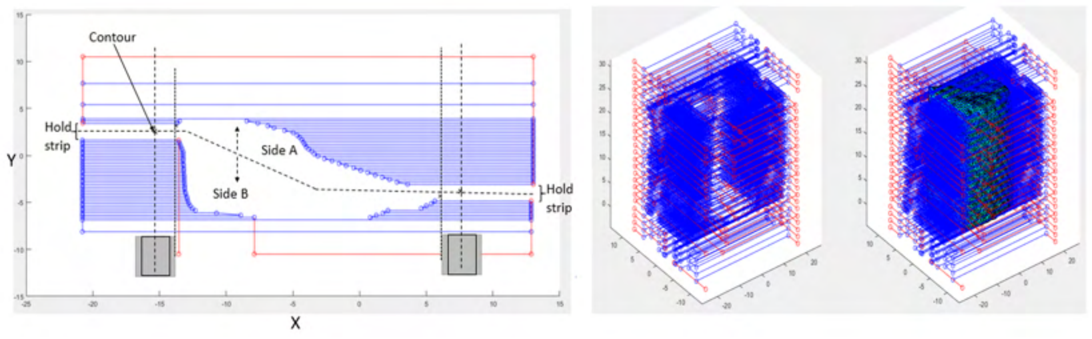


Figure 7.29: All the trajectories for one layer and reconstruction of the toolpath around the original 3D model.

Post-processing This last part is necessary to write the G-code to be fed to the CNC machine. It is made up of three phases listed as follows: referring trajectories to the mill reference system; check for eventual corrections; writing the relative G-code.
Axis Y and Z are swapped to map the software coordinates to the ones of the machine. Checks are performed to make sure the points are still compliant with the machine limits.
A series of computations allow to get an estimation of the machining time to make sure it matches the time frame of the surgery.
Next the machine code, that will be given in input to the mill, is written. The file starts with the setup of the machine and also the local coordinate systems that the machine

will have to use for *side A*. The trajectories are printed in the text as G-code functional blocks. This operation is executed also for the *side B* after having run a rotation of 180°. Last, the contour is appended to the file. It then ends with go to home position and with turn the machine off.

7.2.5 Testing

Experimental tests are conducted to both check the software behavior and to select the correct parameters and hardware. Goes without saying that the method proposed is the result of a process of testing and updating.
During these preliminary tests, the workpiece holder adopted required the creation of a given shape in the lower part of the workpiece. This, being done manually, was not so accurate to fit into the vise. In this way, some passes resulted not as expected.
Concerning the process parameters, *p_roughing* has reduced considerably the time without influencing in negative the final result; the value adopted is 5 [mm]. Tests have been conducted using both "hard" and "soft" plastic specifics. Best results have been achieved for the "soft plastic", in this case the residual material remained at the end of working was almost null and the burs have been easily removed manually (using rotary tools), leaving the operculum clean. The final prosthetic device can be found in Fig.7.30, right after the milling operation, and in Fig.7.31, after cleaning it with a rotary tool (brush with 80 grit size at 50% of the maximum speed of the device, max speed 35 kRPM).

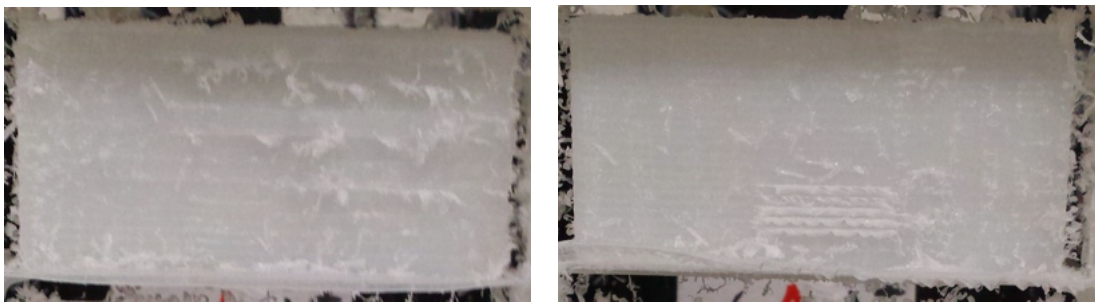


Figure 7.30: mono-cutting bid and "soft plastic": prosthesis removed from the CNC .

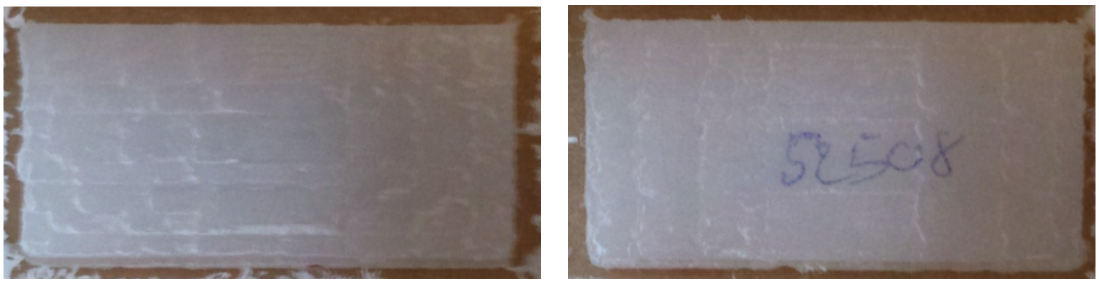


Figure 7.31: mono-cutting bid and "soft plastic": prosthesis after deburring.

Chapter 8

Conclusions and future work

This thesis inquired possibilities and limitations of an AM machine based on an industrial manipulator. Using FDM as a reference, five testing rigs have been developed centered on as many industrial robots. These test benches are constituted by three elements: a direct-drive filament extruder instructed via Ethernet, a reprogrammable machine and a path (and trajectory in some cases) planner. Additional devices have been integrated to carry out specific tasks.
The extruder serviced each one of the robots by only adapting the flange to the robot's wrist thanks to its standardized way to input commands.
The robots considered are a *Mitsubishi RV-2F-Q*, an *ABB IRB 2400*, an *Universal Robots UR3* and a *Techman Robot TM5-700*. From their use (online and offline) and the study of the manuals, three possible controlling architecture (*Data Link*, *Real Time*, *Stand-Alone*) have been outlined and implemented focusing on final position and speed that allow to execute the process as intended to.
The trajectory planner has to generate the toolpath accounting for the manufacturing strategy, robot's range of motion, collision of the parts inside the cell and translate the final process into machine instructions. All of these aspects are mutually affecting each other, hence conventional CAM software for FDM printers have to be re-tailored for this kind of cell. The result is the *RSS*, a CAM software in its infancy whose objective is to offer to the user all the possible tools to exploit the possibilities and contain the limits of a robotic based AM cell.
Practical aspects of the research are presented using actual examples taken from the industrial world, in fields that are not necessarily related to AM to point out that this activity is not merely linked to FDM printing, but also to the generic robotic domain.
In addition to the *RSS* as a mean to better exploit the setup, novel building approaches have been proposed to overcome issues such as inter-layer bonding, crack propagation, mechanical properties, manufacturing time and finish quality. They have been named: *Stitching*, *Layer Shaping* and *TreeD printing*. Only tests about the first one, have been conducted to validate its working principles and the possibility to put it side by side with the methods found in literature. All the three have been discussed detailing how the cell is expected to change (machines number and type, tools and devices).

Future work has to be done to complete the *RSS* and setup. The assessment of the proposed building strategies and their combination with the one already implemented is necessary to promote the development of the solutions that actually improve the yielded component. It is expected to:

- **Cell and parts characterization:** conduct basic test to validate *Layer shaping* and *TreeD printing*; test the *Stitching-2* method using a second robot to insert in the built already manufactured parts to reduce production time.
  Subject the yielded parts (using the different methods discussed) to testing in order to mechanically characterize them; this will guarantee the selection of the solutions that actually improve the final result.

- **Extrusion system:** complete the single-screw extruder with relative control will enhance the cell performance; the system will be able to process different material with limited restriction on the build time.
  Develop a cheap motherboard to manage only the necessary function without relaying on boards sold for other applications; put alongside Ethernet port, analog/digital I/O signals and other protocols compatible with industrial automation to promote integration with PLC based units.

- **Robotic system:** update the cell with multiple machines (when required) and tools such as grippers, measuring devices to monitor and adjust the process definition and parameters.
  Introduce information on the current speed of the TCP (in particular for those cases relying on the built-in motion planner) to update the material flow rate commanded to the extruder.
  Develop of a low-cost cell to appeal also to the hobbyist, that represent a large market share of user of this technology.

- **Trajectory planner:** in addition to the improvements found at 4.8, extend the CAM to run in parallel to the actual process to menage it and react to external event such as: error in the flow rate outputted at the nozzle, arising defects and operators entering the restricted area. Trajectory should be update according to these stimuli.
  Develop the basic routines to test the new building strategies identified in 6.4.4.

Appendices

Appendix A

Surface flattening

This method is also presented in [37]. Due to it simple implementation is used in the early stages to perform basic test with the robot. Some additional considerations are drawn to the generalization of the process as a tool to generate the toolpath.

From literature[116], it is known that if a surface undergoes a bending transformation or a change in the particular parametric form there are features that remain unchanged. These invariant quantities are the length of a curve drawn along the surface and the angle between a pair of curves drawn along the surface and meeting at a common point.
A cylinder (just the lateral surface) is a manifold that admits the definition of length, having indicated with g the metric tensor, x the local coordinates and c a curve, as:

$$L(c) = \int_a^b \sqrt{g_{ij} dx_i dx_j} = \int_a^b \|c'(t)\| dt$$

If a curve parallel to the base is considered, then the length becomes $L(\theta) = r[\theta]_a^b$ in polar coordinates.

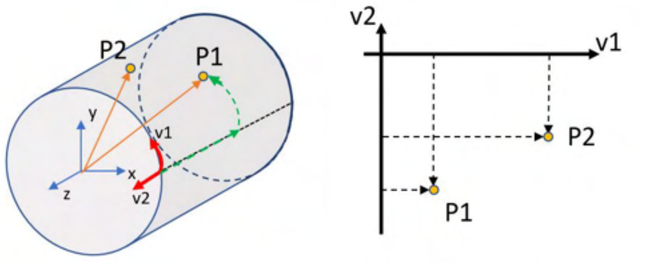


Figure A.1: Base change from $\mathbb{R}^3$ to $\mathbb{R}^2$

A new reference system (*v1*,*v2*) can be constructed as perceived by an observer moving on the surface such that *v1* is equal to $L(\theta)$ and *v2* is z (Fig.A.1). Thus, the surface can be described in the new base as:

$$P(x,y,z) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \\ z \end{bmatrix} \leftrightarrow P^*(r\theta, z) = \begin{bmatrix} \sqrt{x^2+y^2}atan2(y,x) \\ z \end{bmatrix}$$

A straight line in the new base is going to be mapped in a curve preserving the distance between two points, that it will also be the geodesic.
In Fig.A.2 is possible to see some basic toolpaths defined in the new base mapped in the $\mathbb{R}^3$. In order to keep the relative distances as designed in the workspace, the object in the service space need to be reshaped accordingly.
In case of the rib of the example in the section 5.2, then the resulting shape is rectangular trapezoid having the base at the bottom equal to $r \times \pi$, the top base to $(r+h) \times \pi$ where r is the radius of the cylinder and h is the variation of the radius. The stages from STL to path can be seen in Fig.A.3. The 3D extrusion of the trapezoid is sliced (in this case with a commercial slicer, *Slic3r*) to get the G-code that is processed (using Matlab) to reconstruct each layer (the new base is $r\theta, z, r$); the individuals segments are sampled using a suitable discretization (Chebyshev nodes) prior to bending.

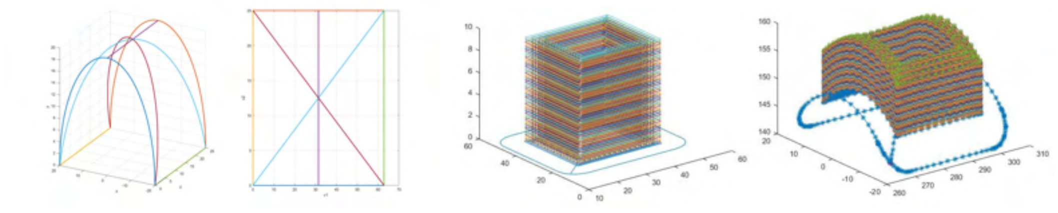


Figure A.2: Example of curved genrated in *v1*-*v2* and mapped in *xyz*

A.1 Generic surface

Let's consider a more complex surface, for instance, one (fig.A.4) behaving as a $x = \theta$; $y = w$; $z = 1 + \cos(\theta/2)$; the metric tensor and length, keeping y constant, are then:

$$g = J^T J = \begin{bmatrix} 1 + 0.25sin^2(\theta/2) & 0 \\ 0 & 1 \end{bmatrix} \quad ; \quad L = \int_a^b \sqrt{1 + 0.25sin^2(\theta/2)}d\theta$$

that depends on the position considered. Controlling the hatch distance in the V space reduces to solve the integral using numerical methods.

In order for this methodology to be generalized, the part to be manufactured should not be modeled using CAD software in the operational space, but in the auxiliary to avoid any deformation of the basic triangle.
Let's consider starting with a 3D model in the workspace and we want to slice it in the

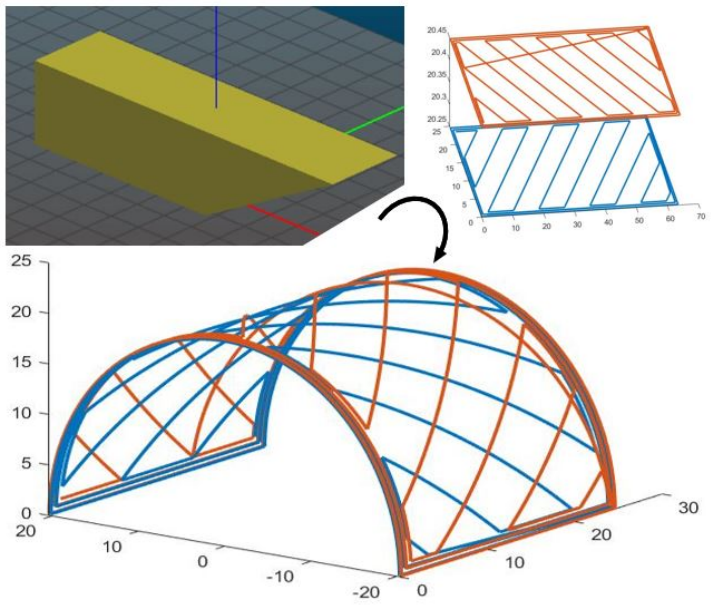


Figure A.3: Rib of the example in the section 5.2.

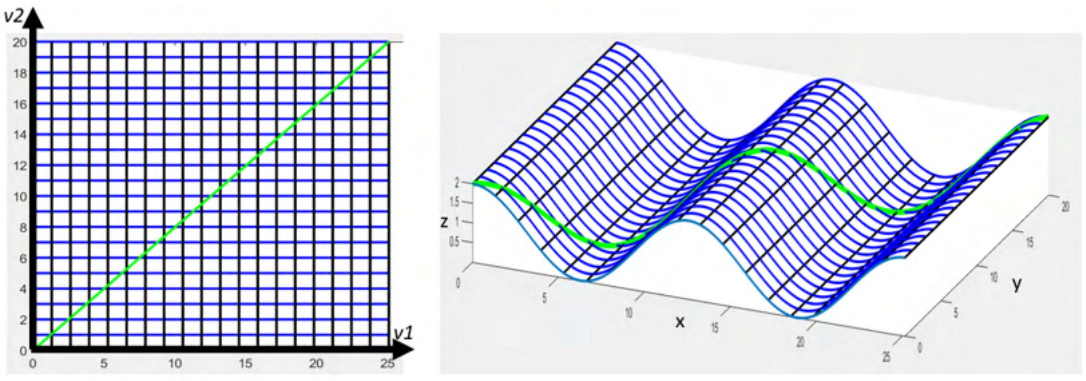


Figure A.4: Surface mapped from V space ($v1=\theta$;$v2=w$) to $x=\theta$; $y=\cos(0.5\theta)$; $z=w$ for $v1\in[0,8\pi]$ and $v2\in[0,20]$

auxiliary space V and then map the toolpath back in its original space. If we generalize the previous curve to also account for third dimension, we get:

$$\begin{bmatrix} x=\theta \\ y=w \\ z=h+\cos(\theta/2) \end{bmatrix} \leftrightarrow \begin{bmatrix} \theta=x \\ w=y \\ h=z-cos(x/2) \end{bmatrix}$$

If a single mesh triangle is considered in the XZ plane so that the vertices are (0;0), (π;1.5) and (0.5;8), it is possible to compute them in the V space: (0;0), (π; 2.5) and (0.5; 8.0311). In this plane it can be evinced that a line in the XZ plane is not necessarily mapped into a line in the V space. The same goes for the half-cylinder case presented earlier. It is worth to point out that the object in Fig.A.3 was possible only because the hull in the workspace was defined using iso-radius or iso-angle surfaces and

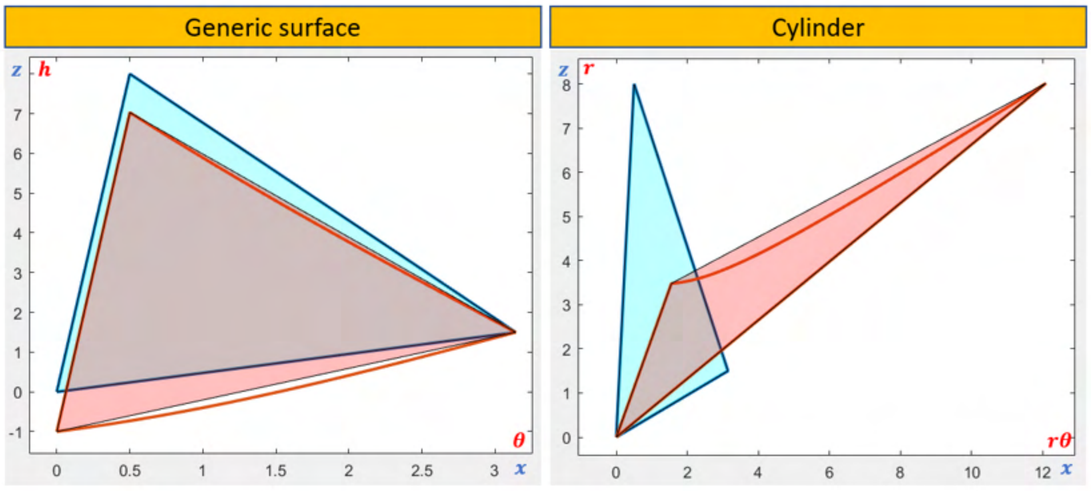


Figure A.5: Deformation of a mesh element due to $S \to V$ in the case of a generic surface and a cylinder. The blue triangle corresponds to the one in the XYZ space, while the red area corresponds to its transformation in the auxiliary space (considering just the vertices) against the one (red contour) obtained converting its sides.

each layer was computed keeping the radius constant.
If this approach is selected to define the toolpath, then each side of a single triangle has to be converted in the auxiliary space prior to the plane-triangle intersection and not only the vertices. In both cases (general and cylindrical surface), the maximum error (between the two triangles deriving from just vertices mapping or sides transformation) is of around 0.2-0.3 mm. This behavior can be seen in Fig.A.5.
The shift from working with 3 points to n is reflected also in the algorithm to be used to generate the intersection points. They can be found using (B.8) to single out only segments of the edges that are cut by the plane ($\prod d_i segment < 0$). Another mean to limit this error is to reduce the mesh size, thus the gap from the side used and the actual curve. This on the other end increases the dimension of the file as well as the number of elements that needs to be sliced.

Appendix B

Collision Theory

Collision Detection is the problems of determining if, when and where two objects touch each other. The determination of when and where they collide is also referenced as *collision determination.*
It is not fundamental only in robotics, but also in simulations, virtual prototyping, computer games and animation. These are example of fields in which the interaction of bodies within themselves and with the environment is crucial to guarantee a life like behavior.
This appendix is based on [117].

B.1 Collision Design issues

When a *CD* system has to be implemented several factors needs to be accounted for. The first ones are the geometrical representation of the world, type of analysis and environment simulation; they can have a high impact on the selection/development of algorithm and the effort required to run it. The *CD* chosen has to comply the required level of performance, robustness and implementation.
Let's deepen the first factors.

Representation of the environment One of the most widespread solution is the adoption of polygon mashes (easy to describe by means of vertices, edges and faces) to describe the simulation world. Object can be approximated by primitives such as boxes and cylinders and operation like union and difference; this approach is called Constructive Solid Geometry (CSG). CSG objects do not directly use meshes, but just the relationship among these simple geometries.
Using less detailed elements reduces the computational cost required by the algorithm.

Type of analysis (queries) Two typologies exist; one being the *intersection testing*, the other the *intersection finding.* The former is just interested in checking if at least one intersection point exists; the latter evaluates the contact points.

Environment Simulation Parameters The number of objects has a great impact on the simulation. Considering n bodies yields n(n-1)/2 tests, therefore even testing few elements can become expensive. Hence the procedure is usually handled in two phases, the broad and narrow one. The identification of small subset (broad phase) that might be colliding is followed by the pairwise test on the subgroup.
The test can be performed dealing one instant at the time (Static test) considering the bodies frozen; these tests are cheaper than Dynamic ones. One of their drawbacks is the time discretization used; if large values are used, the simulation might perceive the bodies just passing by, although a collision is happening inside the interval (*tunneling* phenomenon).
The swept volume SV (i.e. the region occupied by a body during the motion) can be used as an index of the collision:

1. $\nexists\ SV_1 \cap SV_2 \rightarrow$ no intersection will occur
2. $\exists\ SV_1 \cap SV_2 \rightarrow$ intersection can or can nor occur (it's a sufficient condition)

B.2 Broad phase: Bounding boxes

It has been reported how working with simplified version of the actual body can simplify the simulation during the broad phase to generate the likely-to-collide subset. This transformation is done by means of *Bounding Volumes* (BV) that are usually boxes (hence simple geometries) enclosing the original volume. BV substitute the actual body until an overlap is detected then the complex shape can be used to infer the actual intersection locus (narrow phase).
BVs are computed ahead of the simulation, only reorientation (and rigid motion) is performed at runtime. The most widespread BV is the AABB (Axis-aligned Bounding Box). It is a rectangle in 2D space and a parallelepiped in 3D space. It's compute at each time to guarantee the parallelism between the face normals and the coordinate axes. The collision test reduces to evaluate if a vertex lies in the other domain and vice versa.
The Oriented Bounding Box (OBB) differs from AABB because they also consider the orientation of the box. The intersection problem can be solved using the *separating axis* test. It states that two OBB object are separated if the sum of the projected radii on an axis L is less than the distance between their center. 15 axes are required to test OBBs: the 3 axes of the first object coordinate system, the 3 of the second and 9 axes perpendicular to an axis from each.

B.3 Narrow phase: Basic Primitive Tests

If the exact intersection has to be computed, then BV have to be put aside and use other tools.
One approach is based on finding the closest point. For trivial geometrical entities

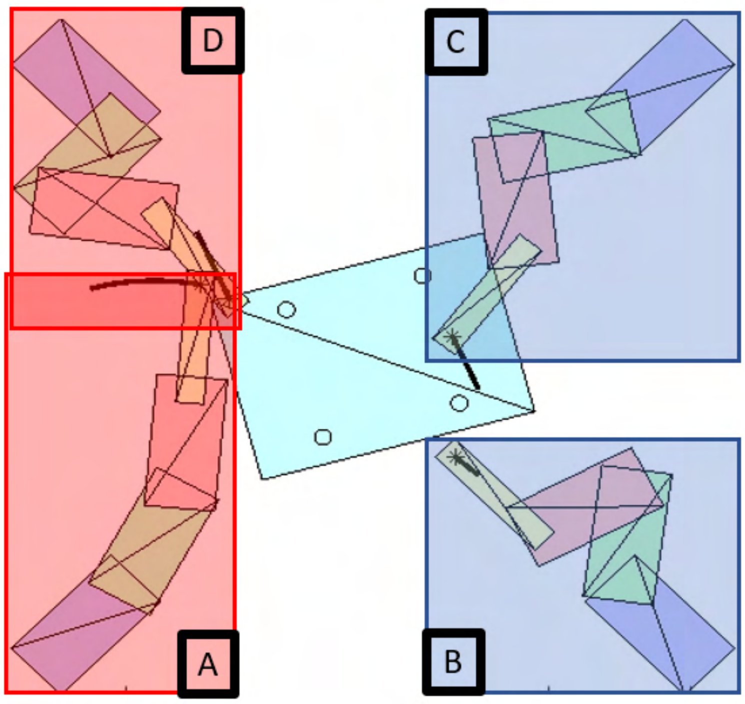


Figure B.1: Example of 4 robot sharing the workspace: each robot is approximated using AABBs, if they intersect (for instance box A and D), the links are approximated using OBBs.

(such as points, segments, lines and planes) the problem can be solved using analytical geometry, for complex shapes it reduces to a minimization problem.
A second approach is based on evaluating the intersection between elements. Many algorithms have been implemented to inquire the segment shared between two triangles (if a triangular mesh is used). The simplest one evaluates the intersection of each side with the other triangle and vice versa. The *Interval overlap method* first determines if the vertices of a triangle cross the other, if so, it calculates the intersection line between the two planes that is in turn intersected with the triangles; if the resulting two segment intersect, the triangles so do.

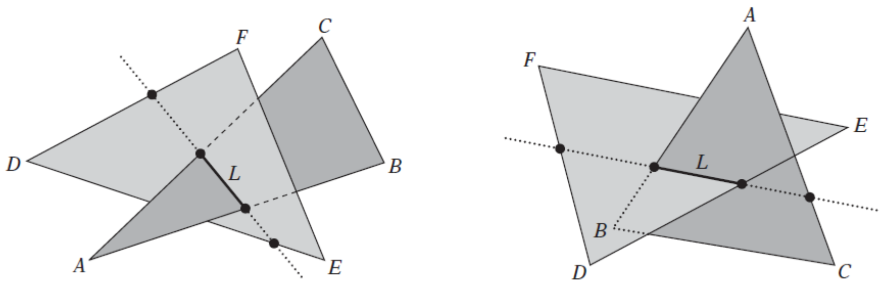


Figure B.2: Two triangles intersect when two edges of one triangle pierce the interior of the other or when one edge from each pierces the interior of the other [117]

B.4 Pairwise test reduction

Bounding Volume Hierarchy (BVH) is a tool to reduce the number of pairwise test required. Object are grouped in small sets and only if they belong together, they will be tested. This subdivision has to carried out spatially rather than functionally and the depth of the hierarchy should not be too deep or shallow.
Spatial partitioning consist on testing objects only if they overlap the same region of space partitioned.

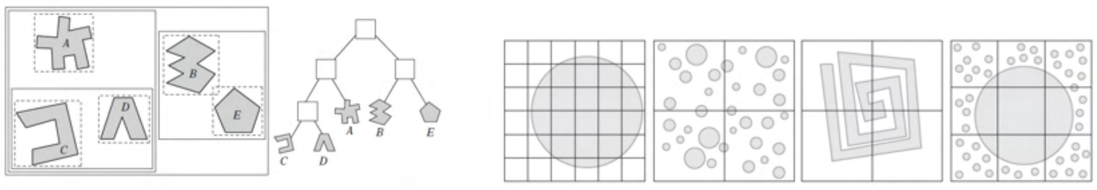


Figure B.3: BHV example(*left*); Spatial partitioning cell size issues(*right*) [117]

B.5 Available libraries

These Collision Detection algorithms gave birth to several libraries such as *I-COLLIDE* [118], *RAPID* [119], *V-Clip* [120] and *FCL* [121].
I-COLLIDE has been developed for convex object; it returns the distance between pairs of elements in the n-body simulation.
RAPID works with triangle either forming meshes or not. The collision check function only accepts two bodies; it then returns a list of overlapping triangles (empty being no intersection).
V-Clip is developed to cope with two objects at the time. To calculate the closest point, the following theorem is used:

Theorem: Let FA and FB be a pair of features from two disjoint convex polyhedra, A and B and let VR(FA) and VR(FB) denote their Voronoi regions. Let PA $\in$ FA and PB $\in$ FB be a pair of closest points between the features. Then, if PA $\in$VR(FB) and PB $\in$ VR(FA), FA and FB are a globally closest pair of features and PA and PB are a globally closest pair of points between A and B (albeit not necessarily unique).[117]

If the conditions are not met (therefore a collision could have occurred), it updates one of the features to a neighbor one (vertex $\rightarrow$ incident edges, face $\rightarrow$ boundary edges). Since only a limited number of such transition can be performed before the distance is decreased, converge is guaranteed as well as the determination of the closest pair of points.
FCL is the library used by ROS (Robot Operating System [122]) and MoveIt [123] to plan the trajectory. It performs several types of queries to evaluate collision and

proximity. Object are represented using a hierarchical structure (bounding volume); a n-body collision algorithm is used to get the state (return a Boolean) or evaluate the exact intersection.

Theory recall [97]:
line: Given a point (P) belonging to the line and the direction (n_{line}), the points lying on the curve can be found solving (B.1) varying the parameter (t).

$$\vec{P} = \vec{P}_0 + \vec{n}_{line} t \tag{B.1}$$

plane: Given a point (P_p) and the normal vector of the plane (n_{plane}), the points lying on the curve can be found solving (B.2) varying the parameter (t).

$$\vec{n}_{plane} \cdot (\vec{P} - \vec{P}_p) = 0 \tag{B.2}$$

line-line intersection: It can be found solving the system, (B.3):

$$\begin{cases} \vec{P} = \vec{P}_1 + \vec{n}_1 t_1 \\ \vec{P} = \vec{P}_2 + \vec{n}_2 t_2 \end{cases} \tag{B.3}$$

line-plane intersection: The intersecting point P^* can be calculated by placing (B.1) in (B.2) and solving for the parameter t and then substituting it in (B.1):

$$\vec{P}^* = \vec{P}_1 + \vec{n}_1 \frac{\vec{n}_p \cdot (\vec{P}_p - \vec{P}_1)}{\vec{n}_p \cdot \vec{n}_1} \tag{B.4}$$

plane-plane intersection: The intersection line is the solution of:

$$\vec{n}_{inter} = \vec{n}_1 \times \vec{n}_2 \tag{B.5}$$

$$\begin{cases} \vec{n}_1 \cdot \vec{P} = \vec{n}_1 \vec{P}_1 \\ \vec{n}_2 \cdot \vec{P} = \vec{n}_2 \vec{P}_2 \\ \vec{n}_{inter} \cdot \vec{P} = 0 \end{cases} \tag{B.6}$$

Where $\vec{P}$, solution of the system is a point belonging to the line.
point sorting on a line: It is possible to sort them by rearranging the distance (with sign) from one of the points (solving (B.7)).

$$d = \frac{P_2 - P_1}{n} \tag{B.7}$$

point-plane distance: The distance between a point $\vec{P}_0$ and a plane ($\vec{P}_1$, $\vec{n}$).

$$d = \frac{\begin{bmatrix} \vec{n} \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \vec{P}_0 \\ -\vec{n} \cdot \vec{P}_1 \end{bmatrix}}{||\vec{n}||_2} \tag{B.8}$$

The distance in (B.8) contains also a sign, to obtain the euclidean distance just consider the module.

Appendix C

PUBLICATIONS

Here, it is reported a list of the publications published/submitted for publication in international conferences and journals.

K. Rane, K.Castelli, M. Strano, "Rapid surface quality assessment of green 3D printed metal-binder parts", Journal of Manufacturing Processes, 2019.

K. Castelli, H. Giberti, "A preliminary 6 Dofs robot based setup for fused deposition modeling", Mechanisms and Machine Science, 2019.

L. Sbaglia, H. Giberti, K. Castelli, "A simplified approach to the calibration of extrusion based AM systems", Mechanisms and Machine Science, 2019.

K. Castelli, A.M.A. Zaki, H. Giberti, "Development of a practical toll for designing Multi-robot systems in pick-and-place applications.", Robotics ,8, 71, 2019.

K. Castelli, H. Giberti, "Additive manufacturing as an essential element in the teaching of robotics", Robotics, 8, 73, 2019.

K. Castelli, H. Giberti, "Simulation of a Robotic Arm for Multi-directional 3D Printing", II International Conference on Simulation for Additive Manufacturing - Sim-AM 2019, 2019.

A.M.A. Zaki, M. Carnevale, K. Castelli, H. Giberti, "A practical tool for the design of multi-robot systems in pick and place applications", The 3rd IFToMM ITALY Conference, 2020.

M. Carnevale, K. Castelli, A.M.A. Zaki, H. Giberti, "Automation of glue deposition on shoe uppers by means of industrial robots and force control", The 3rd IFToMM ITALY Conference, 2020.

Bibliography

[1] A. Dolenc, "An Overview Of Rapid Prototyping Technologies In Manufacturing",1994, Industrial automated institute of Helsinki university of technology

[2] Wohlers T, Gornet T, "History of additive manufacturing", 2014, Wohlers Report, http://wohlersassociates.com/history2014.pdf

[3] R. Ghodssi, P. Lin, "MEMS Materials and Processes Handbook", Springer, 2010

[4] Ian Gibson, David W Rosen, Brent Stucker et al. "Additive manufacturing technologies". Springer, 2015.

[5] Courtesy of Wikimedia user Materialgeeza; https:// commons.wikimedia.org/ wiki/File:Selective _laser_melting _system _schematic.jpg

[6] G. Zhao, G. Ma, W. Xiao and Y. Tian, "Feature-based five-axis path planning method for robotic additive manufacturing", pp. 1-13, 2018.

[7] S. V. Atre, T. J. Weaver and R. M. German, "Injection molding of metals and ceramics", pp. 1-2, 1998.

[8] Stratasys, "Design considerations for additive manufacturing tooling", pp. 6-8, 2018.

[9] Han W. et al. "Tool Path-Based Deposition Planning in Fused Deposition Processes". In: Journal of Manufacturing Science and Engineering 124.2 (2002), pp. 462-472.

[10] G. Zhao, G. Ma, J. Feng and W. Xiao, "Nonplanar slicing and path generation methods for robotic additive manufacturing", pp. 3-5, 2018.

[11] E. A. Nasr and A. K. Kamrani, Computer based design and manufacturing, Springer, 2007.

[12] C. Schranz, "Tweaker - Auto Rotation Module for FDM 3D printing", pp. 5-16, 2016.

[13] W. Oropallo and L. A. Pieg, "Ten challenges in 3D printing", pp. 4-13, 2015.

[14] Z. Zhao and L. LaperriÃ¨re, ”Adaptive Direct Slicing of the Solid Model for Rapid Prototyping”, p. 3, 1997.

[15] K. V. Wong and A. Hernandez, ”A review of additive manufacturing”, pp. 2-3, 2012.

[16] Kenich A, Burnand-Galpin M, Rolland E, et al. ”Lathe-Type 3D Printer”, available at: https://www.scribd.com/doc/147351838/Lathe-Type-3D-Printer. 2013.

[17] Munigala, Vikram et al. ”Development and Design of a Cylindrical 3d Printer.” (2015).

[18] M. Rabionet, A. Guerra, T. Puig Miquel, J. Ciurana, ”3D-printed Tubular Scaffolds for Vascular Tissue Engineering”, Procedia CIRP. 68. 352-357. 10.1016/j.procir.2017.12.094, 2018.

[19] J. Schwaiger, T.C.Lueth, F. Irlinger, ”G-code generation for a new printing process based on 3d plastic polymer droplet generation”, Proceedings of the ASME 2013 International Mechanical Engineering Congress and Exposition 2013

[20] R. Minetto, N. Volpato, J. Stolfi, R. M. M. H. Gregori and M. V. G. da Silva, ”An Optimal Algorithm for 3D Triangle Mesh Slicing”, Computer-Aided Design, 2017

[21] S.H. Choi, K.T. Kwok, (2002) ”A tolerant slicing algorithm for layered manufacturing”, Rapid Prototyping Journal, Vol. 8 Issue: 3, pp.161-179

[22] C. Zhou, ”A Direct Tool Path Planning Algorithm for Line Scanning Based Stereolithography”, Journal of Manufacturing Science and Engineering DECEMBER 2014, Vol. 136

[23] B. Huang and S. B. Singamneni, ”Curved layer adaptive slicing (CLAS) for fused deposition modelling”, pp. 5-7, 2014.

[24] Xu-Zheng Liu, Jun-Hai Yong, Guo-Qin Zheng, Jia-Guang Sun. ”An offset algorithm for polyline curves.”, Computers in Industry, Elsevier, 2007

[25] S. Dhanik, P. Xirouchakis, ”Contour parallel milling tool path generation for arbitrary pocket shape using a fast marching method”, The International Journal of Advanced Manufacturing Technology (2010)

[26] R. Molina-Carmona, A. Jimeno, M. Davia, ”Contour pocketing computation using mathematical morphology”, The International Journal of Advanced Manufacturing Technology volume 36, pages 334-342 (2008)

[27] S.C. Park, B.K. Choi, ”Tool-path planning for direction-parallel area milling”, Computer-Aided Design 32 (2000) 17-25

[28] W. Han, M. A. Jafari, S. C. Danforth, A. Safari, "Tool Path-Based Deposition Planning in Fused Deposition Processes", Journal of Manufacturing Science and Engineering, 2002, Vol. 124

[29] Y. Jin, Y. He, J. Fu, W. Gan, Z. Lin, "Optimization of tool-path generation for material extrusion-based additive manufacturing technology", Additive Manufacturing 1-4 (2014) 32-47

[30] Y. Jin, Y. He, G. Fu, A. Zhang, J. Du, "A non-retraction path planning approach for extrusion-based additive manufacturing", Robotics and Computer-Integrated Manufacturing 48 (2017) 132-144

[31] Jin et al. "Optimization of tool-path generation for material extrusion-based additive manufacturing technology". In: Additive Manufacturing 1 (2014), pp. 32-47.

[32] https://www.3dhubs.com/knowledge-base/supports-3d-printing-technology-overview/

[33] S. Suh, S. Kang, D. Chung, I. Stroud, "Theory and Design of CNC Systems", Springer Series in Advanced Manufacturing

[34] "LinuxCNC resources on G-code", http://linuxcnc.org/docs/html/gcode/g-code.html

[35] "Marlin resources on G-code", https://marlinfw.org/meta/gcode/

[36] S. Singamneni, A. Roychoudhury, O. Diegel and B. Huang, "Modeling and evaluation of curved layer fused deposition", pp. 28-34, 2011.

[37] G. Zhao, G. Ma, J. Feng, W. Xiao, "Nonplanar slicing and path generation methods for robotic additive manufacturing", The International Journal of Advanced Manufacturing Technology (2018) 96:3149-3159

[38] Y. Jin, J. Du, Y. He and G. Fu, "Modeling and process planning for curved layer fused deposition", pp. 273-285, 2016.

[39] H. Jee and M. Kim, "Spherically curved layer (SCL) model for metal 3-D printing of overhangs", pp. 1-5, 2017.

[40] D. Chakraborty, B. A. Reddy and A. R. Choudhury, "Extruder path generation for Curved Layer Fused Deposition Modeling", pp. 3-7, 2007.

[41] J. Baek, D. Anand and K. Redfield, "Finding Geodesics On Surfaces", pp. 1-5, 2007.

[42] J. Wu, N. Aage, R. Westermann and O. Sigmund, "Infill Optimization for Additive Infill Optimization for Additive Porous Structures", pp. 1-16, 2018.

[43] P. Singh and D. Dutta, "Multi-Direction Slicing for Layered Manufacturing", pp. 1-14, 2001.

[44] R. Sundaram and J. Choi, "A Slicing Procedure for 5-Axis LaserAided DMD Process", pp. 1-4, 2004.

[45] A., Gray, "The Local Gauss Map" in Modern Differential Geometry of Curves and Surfaces with Mathematica, 2nd ed. Boca Raton, FL: CRC Press, pp. 279-280, 1997.

[46] G. Elber and E. Cohen, "A Unified Approach to Accessibility in 5-axis Freeform Milling Environments", pp. 1-5, 1999.

[47] L.-W. Tsai, "Robot analysis: the mechanics of serial and parallel manipulators", A Wiley-Interscience publication

[48] J.-P. Merlet, "Parallel Robots", Springer

[49] G. Q. Zhang, A. Spaak, C. Martinez, D. T. Lasko, B. Zhang, and T. A. Fuhlbrigge, "obotic additive manufacturing process simulation-towards design and analysis with building parameter in consideration", IEEE Int. Conf. Autom. Sci. Eng., vol. 2016-Novem, pp. 609-613, 2016.

[50] C. Wu, C. Dai, G. Fang, Y. J. Liu, and C. C. L. Wang, "RoboFDM: A robotic system for support-free fabrication using FDM", Proc. - IEEE Int. Conf. Robot. Autom., pp. 1175-1180, 2017.

[51] J. LAARMAN, S. Jokic, P. Novikov, L. Fraguada, A. Markopoulou, "ANTI-GRAVITY ADDITIVE MANUFACTURING: Negotiating Design & Making", 2017, 10.2307/j.ctt1tp3c5w.27.

[52] F. Xie, L. Chen, Z. Li, and K. Tang, "Path smoothing and feed rate planning for robotic curved layer additive manufacturing", Robot. Comput. Integr. Manuf., vol. 65, no. June 2019, p. 101967, 2020.

[53] M. Chalvin, S. Campocasso, V. Hugel, and T. Baizeau, "Layer-by-layer generation of optimized joint trajectory for multi-axis robotized additive manufacturing of parts of revolution" Robot. Comput. Integr. Manuf., vol. 65, no. February, p. 101960, 2020.

[54] M. Yablonina, M. Prado, E. Baharlou, T. Schwinn, and C. Design, "Mobile Robotic Fabrication System", Fabricate, 2017.

[55] B. Felbrich et al., "A novel rapid additive manufacturing concept for architectural composite shell construction inspired by the shell formation in land snails", Bioinspiration and Biomimetics, vol. 13, no. 2, 2018.

[56] C. Gosselin, R. Duballet, P. Roux, N. GaudilliÃ¨re, J. Dirrenberger, and P. Morel, "Large-scale 3D printing of ultra-high performance concrete - a new processing route for architects and builders", Mater. Des., vol. 100, pp. 102-109, 2016.

[57] E. Barnett and C. Gosselin, "Large-scale 3D printing with a cable-suspended robot", Addit. Manuf., vol. 7, pp. 27-44, 2015.

[58] P.Duan, Y. Liu, J. Ding, M. Zhang, "Development of Vision-Based Control System for Mobile 3D Printer", Proceedings of the ASME 2019, 10.1115/IMECE2019-11577.

[59] M.E. Tiryaki, X.Zhang, Q.C. Pham, "Printing-while-moving: a new paradigm for large-scale robotic 3D Printing", 2019, 10.1109/IROS40897.2019.8967524.

[60] X.Zhang, M. Li, J. Lim, Y. Weng, Y.W.D. Tay, H. Pham, Q.C. Pham, "Large-scale 3D printing by a team of mobile robots", Automation in Construction. 95. 98-106, 2018.

[61] P. M. Bhatt, R. K. Malhan, A. V. Shembekar, Y. J. Yoon, and S. K. Gupta, "Expanding capabilities of additive manufacturing through use of robotics technologies: A survey", Additive Manufacturing, vol. 31. Elsevier B.V., Jan. 01, 2020

[62] S.A.M. Tofail, E.P. Koumoulos, A. Bandyopadhyay, S. Bose, L. O'Donoghue, C. Charitidis, "Additive manufacturing: scientific and technological challenges, market uptake and opportunities", Materials Today, vol. 21, 2018.

[63] "Additive manufacturing: Challenges, trends, and applications", Advances in Mechanical Engineering, Vol. 11(2) 1-27, 2019.

[64] B. Wang, "The Future of Manufacturing: A New Perspective", Engineering, 4, 722-728, 2018.

[65] W. Gao et al. "The status, challenges, and future of additive manufacturing in engineering", Computer-Aided Design 69, 65-89, 2015.

[66] "FDM Design Guideline.", https://facfox.com/docs/fdm-design-guideline.

[67] J. Jiang, X. Xu, and J. Stringer, "Support Structures for Additive Manufacturing: A Review", J. Manuf. Mater. Process., vol. 2, no. 4, p. 64, 2018.

[68] A. Haleem, M. Javaid, "Additive Manufacturing applications in Industry 4.0: A review", Journal of Industrial Integration and Management, 2019.

[69] U. M. Dilberoglu, B. Gharehpapagh, U. Yaman, M. Dolen, "The role of additive manufacturing in the era of Industry 4.0", 27th International Conference on Flexible Automation and Intelligent Manufacturing, 2017.

[70] B. N. Turner, R. Strong, S. A. Gold, "A review of melt extrusion additive manufacturing processes: I. process design and modeling." Rapid Prototyping Journal, 20(3):192-204, 2014.

[71] H. Valkenaers and al. "A novel approach to additive manufacturing: screw extrusion 3D-printing", Proceedings of the 10th International Conference on Multi-Material Micro Manufacture, 2013.

[72] https://manual.slic3r.org/advanced/flow-math

[73] M. Kujawa, "The influence of first layer parameters on adhesion between the 3D printer's glass bed and ABS", 2017

[74] E. M. Petrie, "Handbook of adhesives and sealants", The McGraw-Hill Companies, Inc.

[75] M. Hoque, H. Kabir, M. Jony, "Design and construction of a bowden extruder for a FDM 3D printer uses 1.75mm filament", International Journal of Technical Research & Science, 2018

[76] K. Ogata, "Modern Control Engineering", Pearson Education.

[77] C. B. Highley, C. B. Rodell, J. A. Burdick, "Direct 3d printing of shear-thinning hydrogels into self-healing hydrogels.", Advanced Materials, 27(34):5075-5079, 2015.

[78] J. G. Drobny, "Handbook of Thermoplastic Elastomers" (Second Edition), Plastics Design Library, 2014

[79] D. J. van Zuilichem, W. Stolp, L. P.B.M. Janssen, "Engineering aspects of single- and twin-screw extrusion-cooking of biopolymers", Journal of Food Engineering, Vol: 2, Issue: 3, Page: 157-175, 1983.

[80] G. Campbell, M. Spalding, "Single-Screw Extrusion: Introduction and Troubleshooting.", 2013.

[81] A. Lawal, D. Kalyon, "Mechanisms of Mixing in Single of Co-Rotating Twin Screw Extruders", Polymer Engineering & Science. 35. 1325-1338, 1995.

[82] S. Whyman, K. Arif, J. Potgieter, "Design and Development of a Small-Scale Pellet Extrusion System for 3D Printing Biopolymer Materials and Composites", International Journal of Advanced Manufacturing Technology, 96, 2018.

[83] "Three dimensional finite element simulation of polymer melting and flow in a single-screw extruder: optimization of screw channel geometry" - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/ figure/A - schematic-of -a-typical -single -screw-extruder _fig1_304090772 [accessed 15 Sep, 2020]

[84] H. Patil, R. Tiwari, M. Repka, "Hot-Melt Extrusion: from Theory to Application in Pharmaceutical Formulation". AAPS PharmSciTech. 17. 10.1208/s12249-015-0360-7 (2015).

[85] M. Annoni, H. Giberti, M. Strano, "Feasibility study of an extrusion-based direct metal additive manufacturing technique", Procedia Manuf 2016; 5:916-27.

[86] B.Siciliano, L. Sciavicco, L. Villani, G. Oriolo, "Robotics: Modelling, Planning and Control", Springer

[87] manual: "Robot System Safety and Installation (T, VT/ EPSON RC+ 7.0) Rev.11"

[88] manual: "SCARA ROBOT T series MANIPULATOR MANUAL Rev.9"

[89] manual: "EPSON RC+ 7.0 (Ver.7.4) SPEL+ Language Reference Rev.7"

[90] manual: "Mitsubishi Industrial Robot SAFETY MANUAL"

[91] manual: "RV-2F-Q Series: Standard Specification Manual (CR750-Q/CR751-Q controller)"

[92] manual: "CR750/CR751/CR760 Series Controller INSTRUCTION MANUAL: Detailed explanations of functions and operations"

[93] manual: "Product specification IRB 2400/10 IRB 2400/16"

[94] manual: "Technical reference manual RAPID Instructions, Functions and Data types RobotWare 6.06"

[95] manual: "Universal Robots e-Series User Manual UR3e Version 5.5"

[96] manual: "TM5 Guide Book I and II Hardware Version: 2.00 Software Version: 1.62"

[97] M. Boella, "Analisi matematica 1 e algebra lineare: Esercizi", Pearson Paravia Bruno Mondadori, 2008

[98] K.P.Hawkins, "Analytic inverse kinematics for the universal robots ur-5/ur-10 arms". Technical report, Georgia Institute of Technology (2013).

[99] L. Baldini and al., "Vademecum per disegnatori e tecnici", Ulrico Hoepli Milano

[100] Du Plessis, A., Broeckhoven, C., Yadroitsava, I., Yadroitsev, I., Hands, C., Bhate, D., "Beautiful and Functional: A Review of Biomimetic Design in Additive Manufacturing." Additive Manufacturing, vol. 27, 2019.

[101] A. D. Lantada, A. d. B. Romero, A. S. Isasi and D. G. Bellido, "Design and Performance Assessment of Innovative Eco-Efficient Support Structures for Additive Manufacturing by Photopolymerization", pp. 1-7, 2017.

[102] A. J. Breen, L. M. Guillette, S. D. Healy, "What Can Nest-Building Birds Teach Us?", Comparative Cognition & Behavior Reviews, January 2016

[103] https://www.youtube.com/watch?v=qbWM1QAVGzs

[104] https://www.youtube.com/watch?v=9gbvbKUfHdU

[105] J.D. Halley, M. Burd, P. Wells, "Excavation and architecture of Argentine ant nests", Insectes Sociaux, vol. 52, 2005.

[106] C. Polidori, L. Trombino, C. Fumagalli, F. Andrietti, "The nest of the mud-dauber wasp, Sceliphron spirifex (Hymenoptera, Sphecidae): Application of geological methods to structure and brood cell contents analysis". Italian Journal of Zoology, vol. 72, 153-159 (2005).

[107] https://www.youtube.com/watch?v=6nhcbwkupz8

[108] R. Jeanne, "The Adaptiveness of Social Wasp Nest Architecture", Quarterly Review of Biology, vol 50 (1975).

[109] S. Eshwar, h.dr Channaveerappa, "Architectural Mechanism Developed By Cliff Swallow for Nest Construction". Int. Journal of Engineering Research and Applications, vol. 4 925-930 (2014).

[110] https://www.youtube.com/watch?v=edKsH6P8VMk

[111] https://www.youtube.com/watch?v= tQ0IP8kSwcA& ab_channel = Cornel lLabofOr nithology

[112] https://daysontheclaise. blogspot.com/2016 /02/termite -mounds.html

[113] https://www.britannica.com/science/growth-biology

[114] only image on the right, http://www.copperman.co.uk /didgeridoo/ how_to_make_a_ wooden_didgeridoo/what_is_wood.php

[115] "Cranioplastica, Chirurgia, Appunti di procedura chirurgica. B. Zanotti", A. Cramaro. http://www.cranioplastica.it/chirurgia.htm.

[116] J. M. Lee, "Reimannian manifolds: an introduction to curvature".

[117] C. Ericson, "Real-Time Collision Detection", Elsevier Inc. (2005)

[118] J. Cohen, M. Lin, D. Manocha, M. Ponamgi, "Interactive and Exact Collision Detection for Multi-Body Environments", (1995)

[119] S. Gottschalk, M. C. Liny, D. Manocha, "OBBTree: A Hierarchical Structure for Rapid Interference Detection"

[120] B. Mirtich, "V-Clip: Fast and Robust Polyhedral Collision Detection"

[121] J. Pan, S. Chitta, D. Manocha, "FCL: A General Purpose Library for Collision and Proximity Queries"

[122] http://wiki.ros.org/

[123] https://moveit.ros.org/