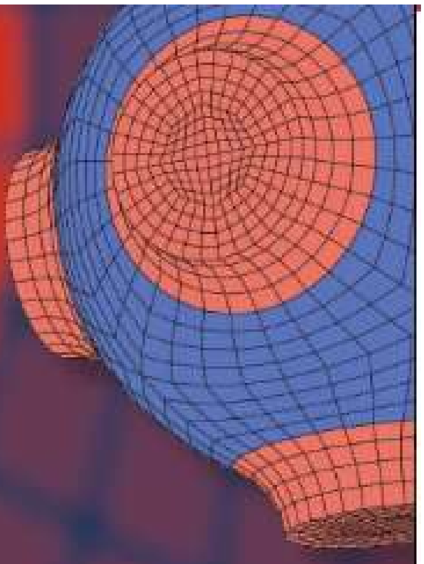
Advanced Structured Materials



António Torres Marques
Sílvia Esteves
João P. T. Pereira
Luis Miguel Oliveira *Editors*

Additive Manufacturing Hybrid Processes for Composites Systems

Springer

Path Generation, Control, and Monitoring

Carlos Faria [1]

Email cfaria@dei.uminho.pt

Daniela Martins [1]

Email daniela_martins93@hotmail.com

Marina A. Matos [1]

Email aniram@live.com.pt

Diana Pinho [1,2]

Email diana@ipb.pt

Bruna Ramos [1]

Email bruna.ramos@dps.uminho.pt

Estela Bicho [1]

Email estela.bicho@dei.uminho.pt

Lino Costa [1]

Email lac@dps.uminho.pt

Isabel Espirito Santo [1]

Email iapinho@dps.uminho.pt

Jaime Fonseca [1]

Email jaime@dei.uminho.pt

M. Teresa T. Monteiro [1]

Email tm@dps.uminho.pt

Ana I. Pereira [1, 2]

Email apereira@ipb.pt

Ana Maria A. C. Rocha [1]

Email arocha@dps.uminho.pt

A. Ismael F. Vaz [1]✉

Email aivaz@dps.uminho.pt

[1] ALGORITMI Research Centre, School of Engineering, University of Minho, Braga, Portugal

[2] Reseach Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Bragança, Portugal

Abstract

A critical issue in additive manufacturing (AM) is the control of the printer actuators such that the deposition of material (or a few different materials) takes place in an organized way. Typically, the actuators are connected with a low-level controller that can receive computer numerical control (CNC) instruction. A 3D printer controller is, usually, expected to receive a set of CNC instructions in a format called G-Code, where a set of control instructions is provided. These instructions include the necessary settings for the printer to work (e.g., a temperature setup) and printer head movement instructions (e.g., the *x*-, *y*-, and *z*-positions in reference axes). The set of the printer actuators positions, where some operations take place, is called the printer path. Path planning or generation corresponds to the computation of the printer head trajectory during a period of time where the object is to be built. A five-degree of freedom/5-axis 3D printer that considers a hybrid process based on additive manufacturing of composites with long or short fibers reinforced thermoplastic matrix is being addressed in this book. The 5-axis printer considers the three usual degrees of freedom plus two additional degrees of freedom, located at the printer table. While software for 3D printing is still possible to be used, full advantage of the printer potential demands for new path generation strategies. We start in Sect. 6.1 by introducing the reader to the optimal orientation of objects, where object orientation is optimal w.r.t. some objective functions that measure the printing performance. Since we are majorly interested in a 5-axis printer control, we present a printer emulator in Sect. 6.2, which allows us to monitor the printing process. Path generation is addressed in Sect. 6.3. We aim to provide flat and curved path planning to take advantage on the 5-axis printer, and in Sect. 6.4, we provide a strategy to print complex objects. The proposed approach for path planning can also be used for inspecting the printed objects by a non-destructive test, and we introduce this topic in Sect. 6.5.

Keywords

Path planning
Slicing
3D simulation
G-code emulator
Optimization
Optimal object orientation
Optimal printing sequence
Optimal inspection path

6.1. Optimal Orientation of Objects

AM has been used over the last decades with a high acceptance in aeronautics and automobile industries, in medical applications, and in the field of biomedical engineering [40]. Also known as rapid prototype (RP) or layer-by-layer manufacturing (LM), additive manufacturing is a process where a specific object is produced using layer-by-layer deposition of material [64]. Jin et al. [29] define it as a group of layer-based joining processes that build physical shapes and structures directly from virtual models. The first technique consists in converting the information of a CAD file into a stereolithography (STL) file [21]. STL is nothing else than an approximation (tessellation) of the CAD model in which the geometrical features of the 3D models are described by a mesh of triangles and corresponding surface normal vectors. Such a technique eases the way of dealing with the models and, therefore, STL became one of the most popular and widely accepted used file format in the LM industry [25].

LM processes emerged as an alternative to the traditional subtractive manufacturing (see [53] and references therein for other manufacturing processes). LM possesses some challenges related to model surface quality. The stair-stepping effect is one of the major problems inherent to LM [39]. Other challenge pointed out to LM are the low deposition quality, largely related to the filling strategy (the path deposition length and the strategy itself) and the type of used material, as well as the poor surface finish of printed objects. These challenges pose difficulties to the dissemination of LM techniques [30].

Typically, four planning stages must be considered in LM: initial orientation of the objects/parts being built, supports generation to ensure that overhanging features can be built without presenting major object deformations, slicing, and path planning [34]. This section focuses on the first three planning stages.

A proper selection of the initial object orientation is essential to reduce the supports generation’s need. However, some objects may be impossible to build without the use of supporting structures. Some authors claim that a proper orientation can also reduce the building time of the desired objects.

The slicing task refers to a procedure in which planes are intersected with the model in order to determine contours defining where the material will be deposited [34]. Over the last decades, different slicing strategies have been proposed for different LM techniques.

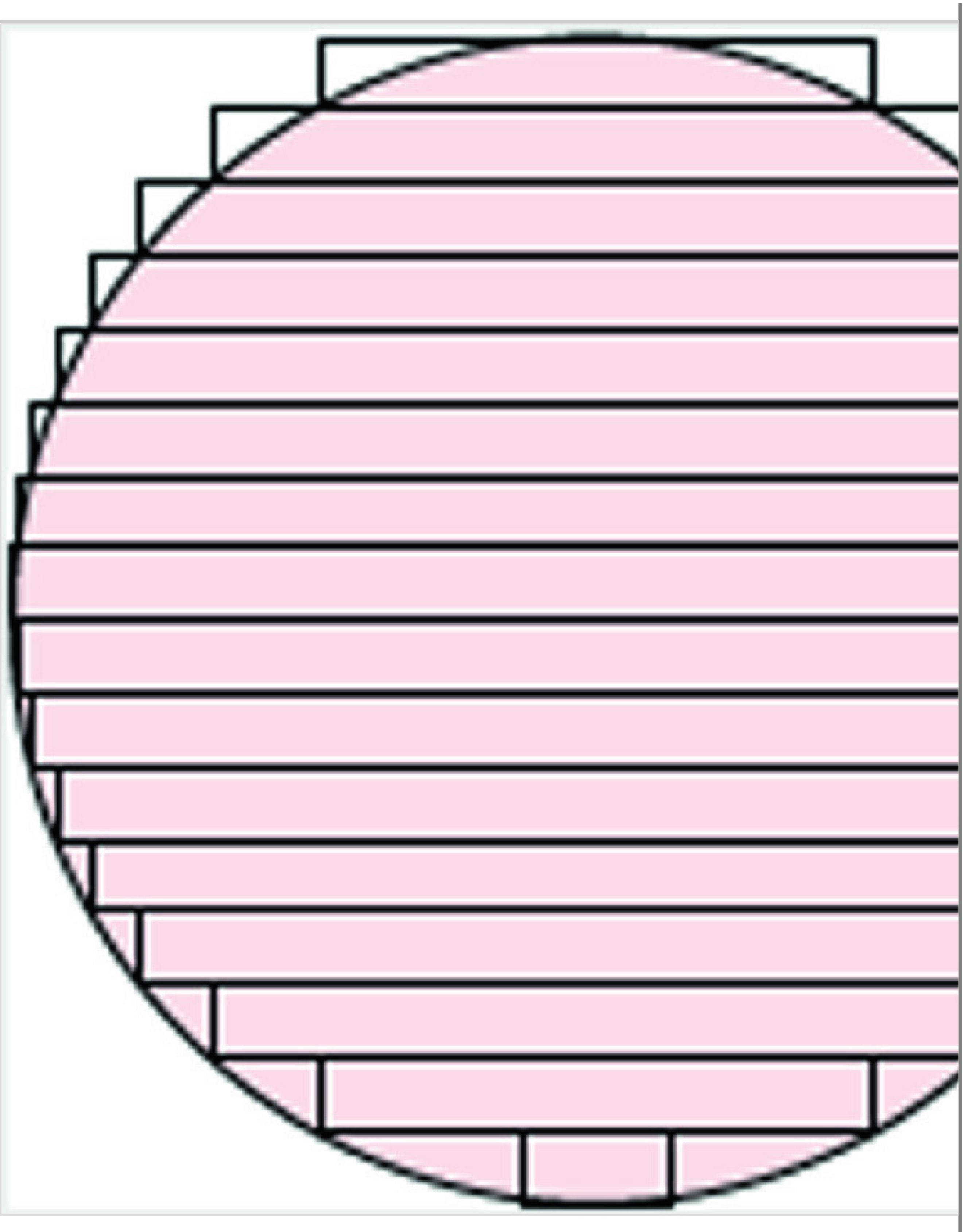
Recently, some bibliography has emerged where state-of-the-art optimization solvers are used to address the optimization of the final printed object orientation, based on minimizing the staircase effect, the need of supports, and the total building time.

6.1.1. Measuring Printing Quality

The slicing process in AM/LM possesses significant challenges. This process consists of cutting any 3D model into a set of slices with a certain thickness. Therefore, each slice is nothing else than a model layer represented in a two axes plane. The 3D model is then obtained by vertically incrementing each layer over a third axis with the step corresponding to the layer thickness. Slicing can be classified as direct when it takes place from a computer-aided design (CAD) software or indirect when the object is represented as an approximation (e.g., in the STL format). Over the last years, different slicing strategies have been proposed in the literature. The bottlenecks of the LM process can be reduced by using appropriate slicing processes. According to Oropallo et al. [46], there are two main issues regarding the slicing process. One is the staircase effect due to the stacking of each layer, and the other is what it is called the containment problem. These problems occur since different layers may fall inside or outside of the original objects contours as is schematically shown in Fig. 6.1 (in two dimensions for a better visualization), where the original object is represented by a disk, and layers are represented by rectangles. The staircase effect results from the representation of curved objects by layers and the containment problem consist in representing the object by layers that are approximating the object from inside or outside.

Fig. 6.1

Staircase effect and containment problem

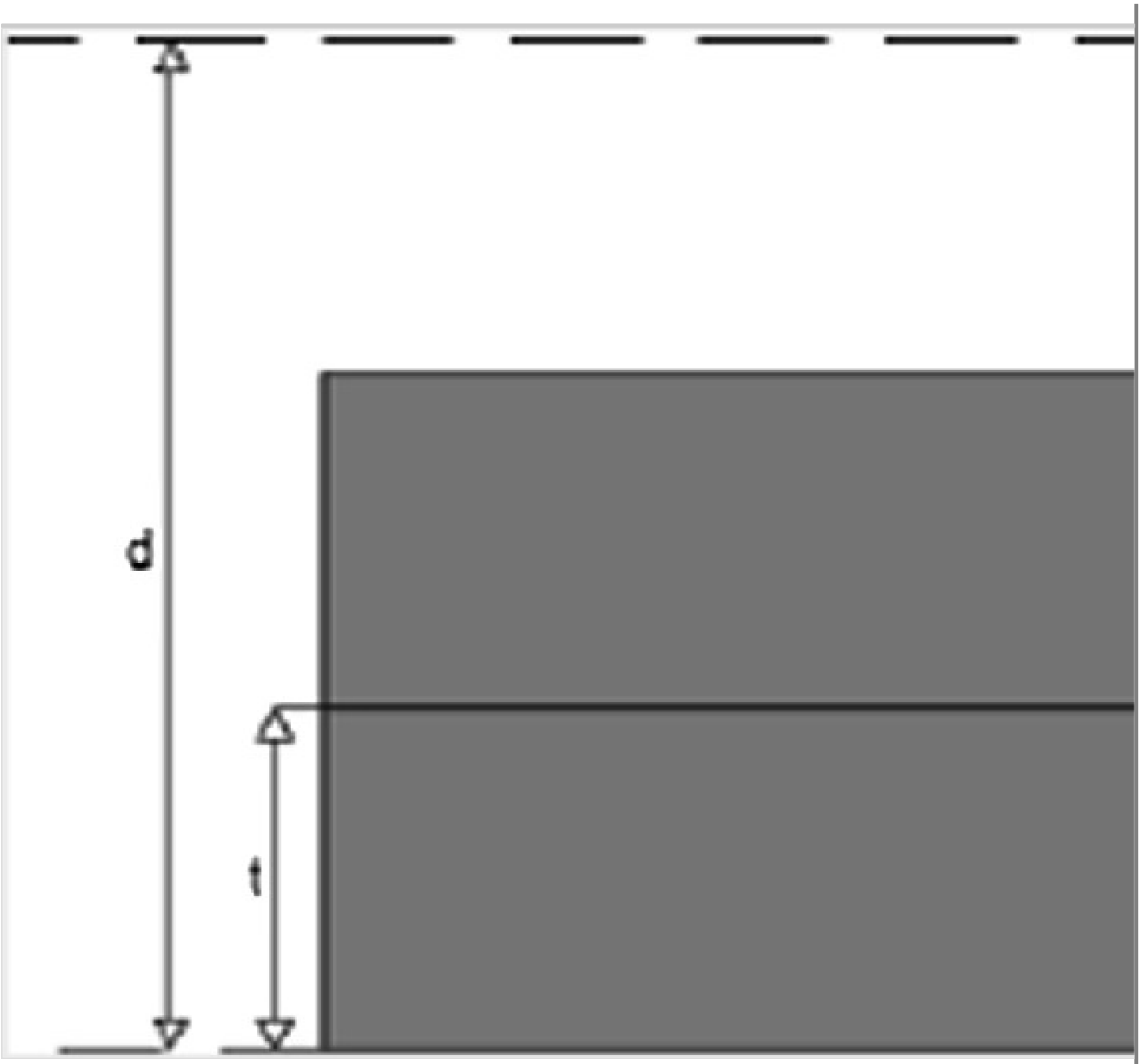


Despite the method used for slicing (direct or indirect), there are two main different strategies concerning the slicing process. Slicing can be uniform and adaptive; the former is used for the construction of layers with the same thickness and the later to construct layers with different thicknesses (adaptive). The adaptive layer thickness usually depends on the slope and curvature of the object: Thicker slices are used for thicker slopes and large curvatures, and thinner slices are used for thinner slopes or small curvatures. Adaptive slicing was firstly presented and addressed in Dolenc et al. [18], where it is presented as a way to restrict the staircase effect. This is achieved by selecting a layer's thickness given by the cusp height tolerance (meaning the measure between the slice vertex and the model surface).

Figure 6.2 depicts the cusp height to better understand how it can be used as a measurement of the quality of the built objects. A simple inspection of Fig. 6.2 gives rise to Eq. (6.1) below, which gives us the relation between the building angle (β), the cusp height (Hc), and the layer's thickness (t). The object staircase effect can be measured by summing up all the cusp heights formed between every slice and mesh triangle.

Fig. 6.2

Cusp height (Hc) representation



$$\cos(\beta) = \frac{Hc}{t} \Leftrightarrow \beta = \arccos\left(\frac{Hc}{t}\right) \quad 6.1$$

Lower cusp heights promote a better approximation of the object contours, leading to lower layers' thickness and, therefore, a balance between the cusp height and the number of layers must be considered since the building time often depends on the number of layers. In Jung and Ahluwalia [32], the cusp height is used to measure the quality between two consecutive layers. Wang et al. [69] develop a technique to reduce the manufacturing time of 3D printing using an adaptive slicing strategy to optimize slices thickness. Printed objects are evaluated by considering the cusp height as a measure of quality. The proposed technique consists in the division of the object in sub-parts, independently optimizing the slicing for each one. The results presented led to saves of 30–40% in the printing time. In Rianmora and Koomsap [56], an adaptive direct slicing approach is addressed. This approach consists in the application of an image processing technique to determine appropriate thickness for each sliced layer and to recommend slicing positions. The obtained results were compared with different techniques using different cusp height values. Results show that the adaptive direct slicing approach leads to a lower number of layers with direct impact on the building time. Other works considering the cusp height as a quality measure can be seen in [35, 37, 49, 61].

Object surface roughness can be measured by looking at the Ra value. The Ra value is computed by considering an experimentally obtained confidence interval for the roughness. The Ra can be obtained by using, e.g., Eq. (6.2) and was firstly addressed in [47, 50].

$$\mathrm{Ra} = (\mathrm{a\ to\ b}) \quad \frac{t}{\cos\beta} \tag{6.2}$$

where (*a* to *b*) is the confidence interval, *t* is the layer thickness, and *β* is the angle between the building direction vector and the normal vector. The Ra quality measure is also used in [51]. In this study, a multi-criteria genetic algorithm was used in order to determine the optimal object deposition orientations. The two objectives used are the surface roughness and the building time. Singhal et al. [61] develop an adaptive slicing strategy using the surface roughness of objects, and this procedure was considered as a starting point for the work in [5].

Taking into account the cusp definition presented in Eq. (6.1), one can compute the staircase effect (SE) considering the total cusp height by summing up all the individual cusp contributions, leading to the following equation:

$$\mathrm{SE} = \sum_j \frac{2\, n_j | A_j}{t^2 |d .} \qquad 6.3$$

where *t* is the (constant) layers height, *d* is a normalized (i.e., $\|d\| = 1$) slicing direction, n_j is a normalized mesh triangle *j* normal vector, and A_j is the mesh triangle *j* area.

Support generation and model orientation are two tasks of the LM process that can significantly influence the result of any built object. Often, both support generation and model orientation are dependent on each other, since only after model orientation it is possible to determine the overhanging parts of the model and thus the need or not of support generation. According to Kulkarni et al. [34], two types of supports can be considered: internal and external. While external supports are essential to support overhanging features, internal are used to support models parts with hollow surfaces. The need of supports must be minimized, since it leads to increasing costs of the manufacture objects due to the increase of building time and consumed material and to the decrease of surface quality in places where supports are built [34].

The need for supports can also be measured by considering the cusp height, but in this case taking it only when the facets are facing down, i.e., when *d* . n_j is negative. Therefore, the need for supports can be measured by the following equation representing the support area (*SA*):

$$\mathrm{SA} = \sum_j A_j |d . n_j| \delta, \qquad 6.4$$

where

$$\delta = \begin{cases} 1, & d . n_j < 0, \\ 0, & d . n_j > 0. \end{cases} \qquad 6.5$$

The manufacturing time of an object depends on its initial orientation as the number of slices to be considered varies with object orientation. Object orientation can improve the accuracy of the built object, reduce the number of generated supports, and consequently decrease the final building costs. Cheng et al. [12] present a multi-objective optimization problem to determine the optimal object building orientation. Essential requirements pointed out by these authors to obtain the best object building orientation are maximization of the number of perpendicular surfaces, maximization of the number of up-facing horizontal surfaces, maximization of the number of holes with their axes in the slicing direction, maximization of the area of the base surface, minimization of the number of slope surfaces, minimization of the total area of overhang surfaces, minimization of the total number of slices, and minimization of the height of required support structures. Richard and Crawford [57] consider the strength of the building objects as a measure of quality. Their objective function takes into account the object strength, the surface errors, the building time, and the volumetric supports.

An approximation to the building time (BT) may be obtained by computing the object height along the slicing direction, leading to the following equation:

$$\begin{aligned} BT = {} & \max(d.\ v_1, d.\ v_2, \ldots, d.\ v_n) \\ & - \min(d.\ v_1, d.\ v_2, \ldots, d.\ v_n) \end{aligned} \tag{6.6}$$

where v_i, $i = 1, \ldots, n$, are the mesh triangles vertices.

In Hussein et al. [26], a new strategy to minimize the negative effects of supports in the manufacturing procedure is presented. These authors introduce a new design and manufacturing support characterized by its efficiency. Such a support has the form of a lattice structure which results in a very low volume, leading to a significant amount of material savings and a reduction of the building time. Strano et al. [63] describe a new approach to minimize the need of support structures. They have developed a new algorithm that performs a two-step optimization procedure by firstly obtaining the best orientation that originates the minimum use of supports and then generating a cellular support structure using the computed orientation. This strategy leads to significant materials saving along with building time improvement.

6.1.2. A Global Optimization Approach

Jibin [28] introduced a new multi-objective optimization strategy to simultaneously minimize the staircase effect, the need for supports, and the building time. This author presented some numerical results for an object, in which a genetic algorithm for multi-objective optimization was then applied. However, as we will see in this section, we only need to consider a single objective when building symmetric objects.

This simplification will then allow the application of a state-of-the-art solver from derivative-free global optimization.

Objects in which we are interested to exhibit some regularity in the sense that the number and area of triangles leading to $d \,.\, n_j > 0$ are the same of the ones leading to $d \,.\, n_j < 0$, which then makes the SE and SA measures defined by Eqs. (6.3) and (6.4) to be non-conflicting when used as objective functions. Also, SE and SA should not account for all mesh triangles, since the cusp is not well defined when $|d \,.\, n_j| = 1$ and there is no need to build support at mesh triangles in the printing table base where $d \,.\, n_j = -1$.

The building time will not be considered in the present subsection, since the simplification of the true building time given by Eq. (6.6) is not appropriate for the type of objects of interest to us.

We will therefore use adapted SE and SA measures in the objective functions for numerical testing:

$$SE = \frac{t^2}{2}\sum_i \begin{cases} |d \;\; n_j|\, A_j, & \text{if } |d\,.\,n_j| \neq 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$SA = \sum_i \begin{cases} A_j\, |d\,.\;\; n_j|\, \delta, & \text{if } d\,.\,n_j \neq -1 \text{ and triangle } j \text{ is not at the printing table base} \\ 0 & \text{otherwise.} \end{cases}$$

We consider the bound constrained optimization problem

$$\min_{\theta\in[0,180]^2} f(\theta) \qquad 6.7$$

where $\theta = (\theta_x, \theta_y) \in [0, 180]^2$ are the object rotation angles (in degrees) along with the x- and y-axes. Recall that this is mathematically equivalent to compute a slicing normalized direction d.

A global minimum of problem (6.7) is to be computed, and thus, we have selected one of the state-of-the-art solvers for global derivative-free optimization subject to simple bounds on the variables (PSwarm [65, 66], available at www.norg.uminho.pt/aivaz/pswarm).

Numerical results reported in [53] consider three different objects: an "Humanoid" included due to its simplicity and the other two objects corresponding to applications of 3D printing in the aerospace industry. Each object has a specific degree of complexity indexed by the number of triangles (facets) used to compose the object. The need to use global optimization to achieve a satisfactory approximate solution for the optimization problem (6.7) was confirmed by the numerical results, due to the existence of many local minimizers and the extensive presence of non-differentiability, thus excluding the possibility to use gradient or Newton-type methods. The reported numerical results have shown the effectiveness and robustness of the proposed approach. Additionally, numerical findings have confirmed the non-conflicting nature of the two objective functions under the symmetry of the objects.

6.1.3. A Multi-objective Optimization Approach

Several approaches have been carried out to determine the orientation of a model based on single-objective optimization. Usually, the objective functions used for optimal build orientation were the building height, staircase effect, volumetric error, volume of support structures and part area in contact with support structures, surface quality, surface roughness, and build deposition time [10, 36, 42, 54, 59, 64].

Recently, multi-objective approaches have been developed to determine the optimal object building orientation, essentially by reducing the multi-objective problem to a single-objective one using classical scalarization methods such as the weighted sum method [8, 9, 28, 38].

A multi-objective optimization approach, using NSGA-II and MOPSO algorithms, considering as objective functions the surface roughness and the build time, for different models, was developed by Padhye and Deb in [48].

Gurrala and Regalla [24] applied the NSGA-II algorithm to optimize the strength of the model and its volumetric shrinkage as objective functions.

In this section, a multi-objective optimization approach to optimize the support area and the build time in order to get the best orientation of a Duct model [53] using the electromagnetism-like (EM) algorithm [60] combined with weighted Tchebycheff scalarization [62] method is presented. The EM algorithm is a population-based stochastic search method for global optimization that mimics the behavior of electrically charged particles. The method uses an attraction–repulsion mechanism to move a population of points toward optimality. The weighted Tchebycheff method was selected since it can be used to solve problems with non-convex Pareto fronts and can find non-extreme solutions (trade- offs) in the presence of multiple conflicting criteria. In this method, the L_∞ norm is minimized, i.e., the maximum distance to a reference point (or aspiration levels) is minimized. In this case, the reference point is defined as the ideal vector and the weights are uniformly varied to obtain different trade-offs. The ideal vector can be computed by determining the optimum of each objective. In this manner, after the search, a set of Pareto optimal solutions is presented as alternatives and the decision-maker can identify the compromises and choose according to his/her preferences.

The multi-objective optimization is formulated as

$$\min f(\theta_x,\theta_y) = \{f_1(\theta_x,\theta_y), f_2(\theta_x,\theta_y)\}$$

$$\text{s.t. } 0 \le \theta_x,\theta_y \le 180 \tag{6.8}$$

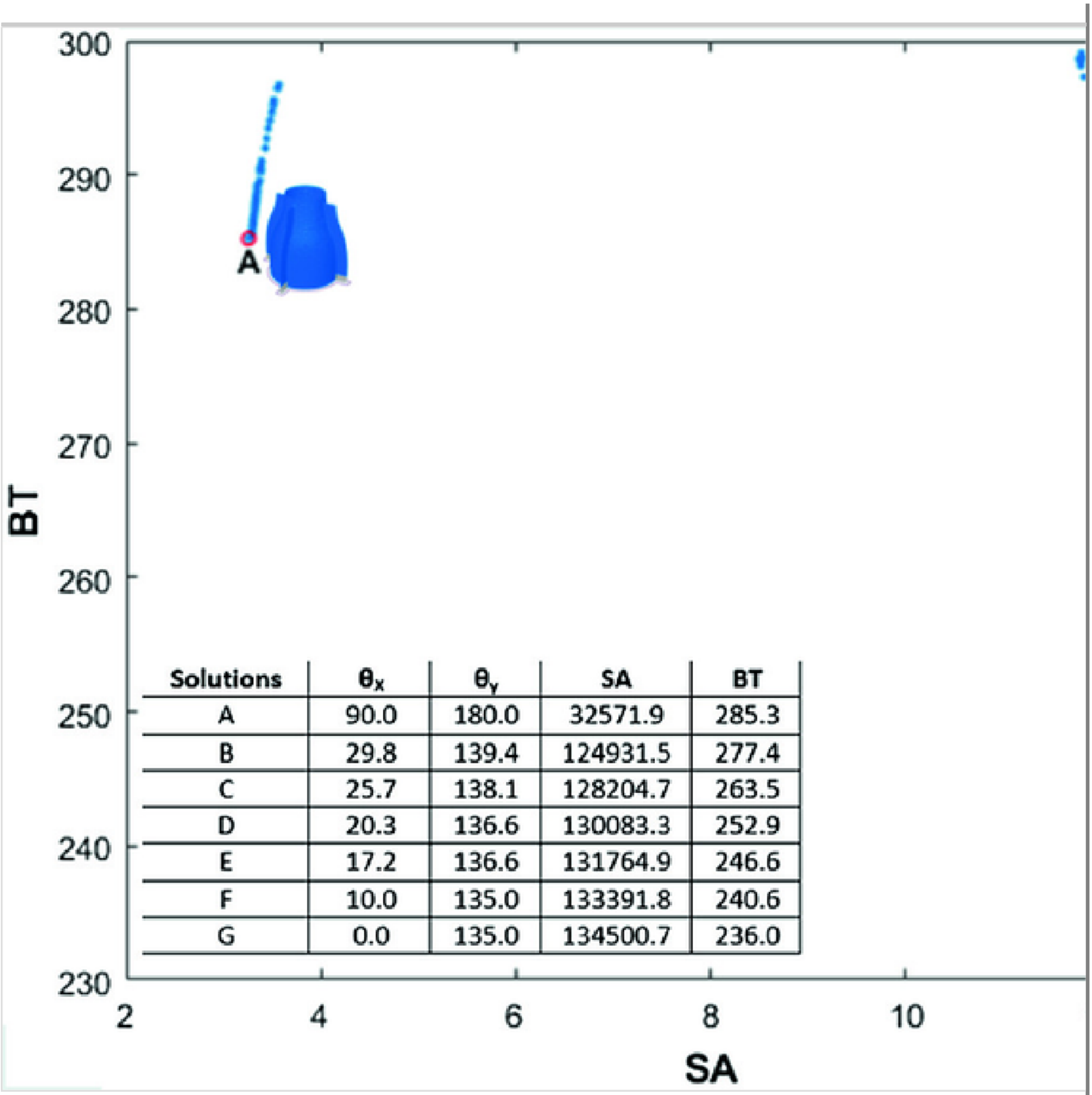
where the objective functions $f_1(\theta_x, \theta_y)$ and $f_2(\theta_x, \theta_y)$ are, respectively, the support area, SA in Eq. (6.4), and the part building time, BT in Eq. (6.6).

In order to compute the objective functions, a slice of 0.2 mm was applied. The objective functions were normalized using the ideal and nadir vectors. The weights were uniformly varied, i.e., $(w_1,w_2) \in \{(0,1),(0.1,0.9),\ldots,(1,0)\}$. A population size of 20 and a maximum number of function evaluations of 2000 were considered for the EM algorithm. For each combination of weights, 30 independent runs were performed.

Figure 6.3 plots the non-dominated solutions (in red) obtained in the objective space for the Duct model. The table presents the angles and objective function values for the seven representative non-dominated solutions of the Pareto front (solutions A to G), representing different trade-offs between the objectives. The Pareto front is non-convex for this problem. Solutions A and G are the optimal solutions in terms of SA and BT, respectively. It is observed that solution B is a little more advantageous in terms of BT in relation to solution A, but it is quite worse in terms of SA. From solutions B to G, a slight degradation in the SA objective and a significant improvement in terms of BT is observed. The 3D representations of solutions A to G can also be seen in Fig. 6.3. Solution A has the best value of *SA* and the worst value of BT and requires few supports although the part may take longer to be printed because it corresponds to its larger height. In the solutions B to G, the part lies down, resulting in a reduction in *BT*, but increasing the number of supports to be used.

Fig. 6.3

Pareto front and representative solutions for the duct model



Solutions	θ_x	θ_y	SA	BT
A	90.0	180.0	32571.9	285.3
B	29.8	139.4	124931.5	277.4
C	25.7	138.1	128204.7	263.5
D	20.3	136.6	130083.3	252.9
E	17.2	136.6	131764.9	246.6
F	10.0	135.0	133391.8	240.6
G	0.0	135.0	134500.7	236.0

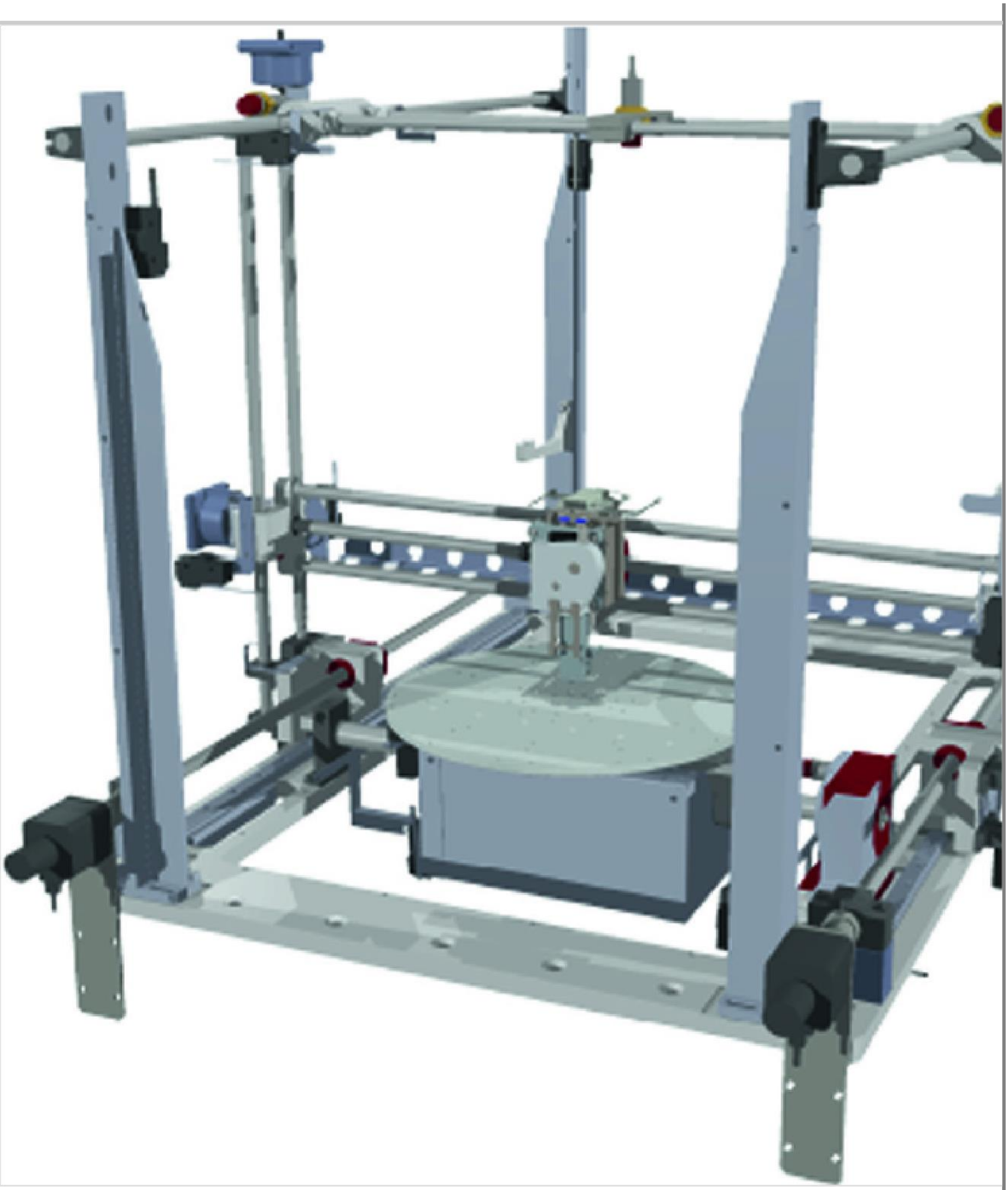
These results allow to perceive the relationship between the objectives for the model, being possible to identify the trade-offs between the objectives and select the most appropriate solution. Therefore, it is clear the advantage of using a multi-objective approach that considers different criteria to find the best orientation of building 3D CAD models, as can be seen in [44].

6.2. 5-Axis Printer and Emulator—Graphics Emulator Tool—FIBR3DEmul

Standard 3D printers have three degrees of freedom allowing the nozzle (or the printer bed) to move along the *x*-, *y*-, and *z*-axes. The type of printer we are considering has two additional degrees of freedom located at the printer bed, one allowing for the printer bed to rotate at the central point and another one allowing the printer bed to tilt (see Fig. 6.4 for a virtual representation of the printer, named as C3CPrinter). Available software for 3D printing (e.g., Slic3r© or CURA©) can also be used for this type of printer, but it takes no advantage on the extra degrees of freedom.

Fig. 6.4

C3DPrinter in the robotics simulator software ~~V-Rep~~CoppeliaSim



6.2.1. FDM Simulation

The fused deposition modeling (FDM) process for a standard 3-axis Cartesian printer is well established. This fact is supported by the amount and variety of tools to guide the user through all stages of the process: from object design, to slicing, to printer parametrization and actual filament deposition.

When the paradigm shifts to printers with more than 3 simultaneously actuated axes, the number of available solutions to generate tool paths or to test the machine operation is almost non-existent. This places an extra burden on developers of new platforms, which are only able to test the developed algorithms with the final machine assembly. Not only does the system development cycle is longer, but unforeseen machine operation faults might also lead to equipment damages.

To address these questions, the FIBR3DEmul is proposed. It is an emulation tool developed to replicate the behavior of a 5-axis FDM printer. The FIBR3DEmul solution consists of two separate applications, one to parse and interpret a custom G-Code protocol (ISO/DIN 66025 standard), and the other to virtually simulate the operation of the real machine with an embedded collision detection mechanism. Both applications are created to facilitate the development process of custom printers with up to 5 simultaneously actuated axes.

G-Code standards are typically formulated for 3-axis printers. To control the additional 2-axis that moves the printer bed, a new G-Code protocol was required. To interpret this new G-Code protocol, an application was developed in C#. It parses the new G-Code commands that control the 3 + 2 axis of the C3DPrinter and formulates structured messages that are sent to the virtual printer controller.

To virtually simulate the operation of the real mechanism, we developed a plug-into handle with the structured messages from the G-Code interpretation application and control the behavior of the virtual machine following the standard of the G-Code. This application should mimic the operation of the real machine and include collision detection mechanisms to detect and prevent possible problems with the G-Code script prior to executing the code in the real controller. Given the premises to the problem, the ~~V-~~

~~Rep~~CoppeliaSim (Coppelia Robotics GmbH, Zürich, Switzerland)[1] robotics simulator was selected to develop our solution.

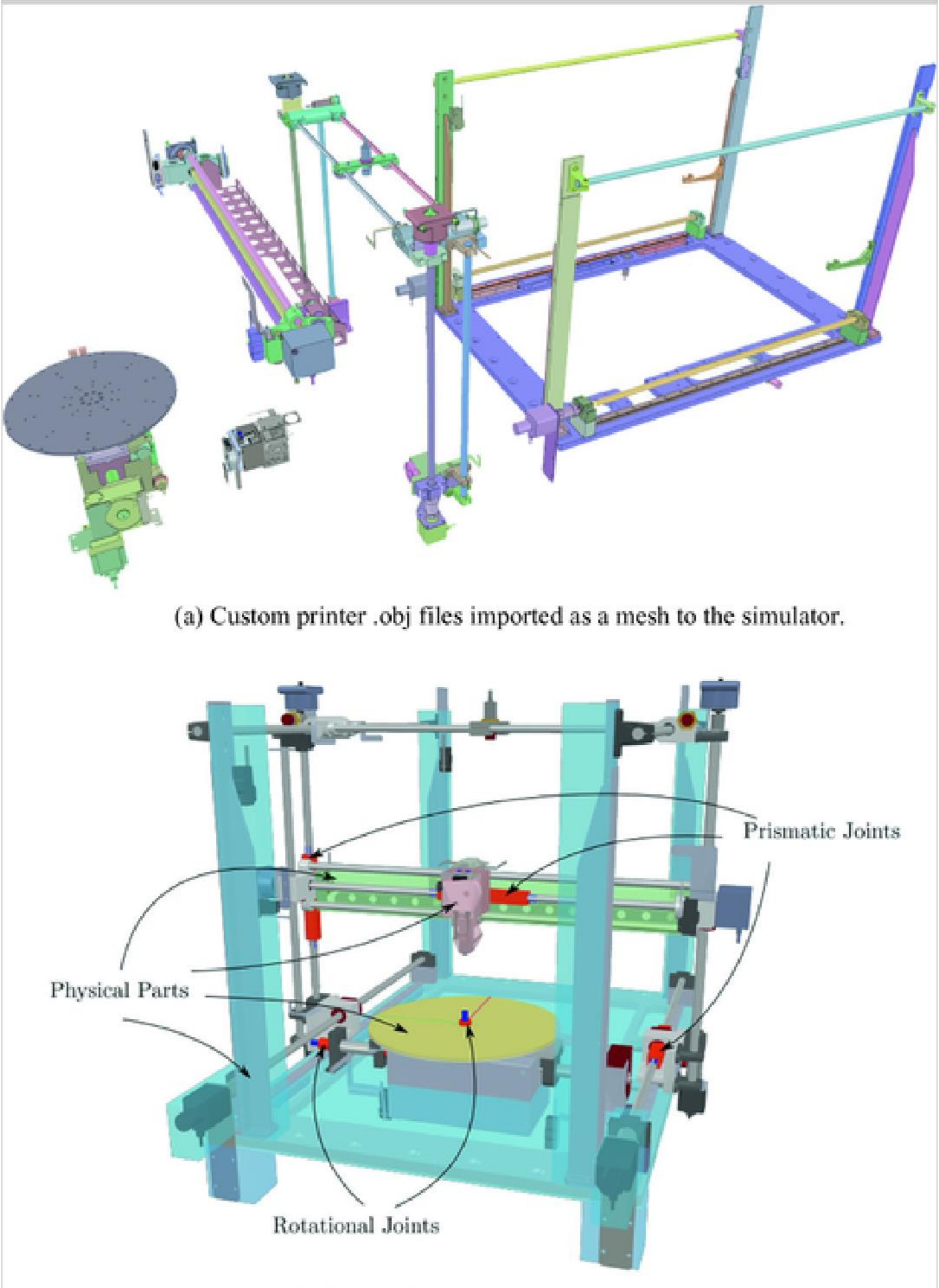
6.2.2. The Virtual C3DPrinter

~~V-Rep~~CoppeliaSim is one of the most popular and versatile robotics simulator software available. It counts with an extensive library of robots, sensors, models, etc., as well as a wide offer in terms of control algorithms for path/motion planning, collision detection, kinematics, dynamics, and more. More importantly, the simulator permits creating custom multi-actuated models similar to the prototype printer and designing a control strategy to closely mimic real printers.

The printer prototype designed in CAD software is exported as a set of separate .OBJ files. These files are imported into a ~~V-Rep~~CoppeliaSim scene as separated meshes with no physical shape (different colored objects in Fig. 6.5a). To create a functional actuated printer, the physical parts of the printer are added to match the graphical entities. First, physical bodies are added to match the graphical meshes, Fig. 6.5b. Contrary to the graphical counterpart, the physical bodies have dynamic properties: the mass, the center of mass, principal moments of inertia, and the inertial frame. These physical rigid bodies[2] are handled by the physics engine to dynamically simulate each component interaction, joint actuation, detect collisions, etc.

Fig. 6.5

Extruder model graphical and physical part



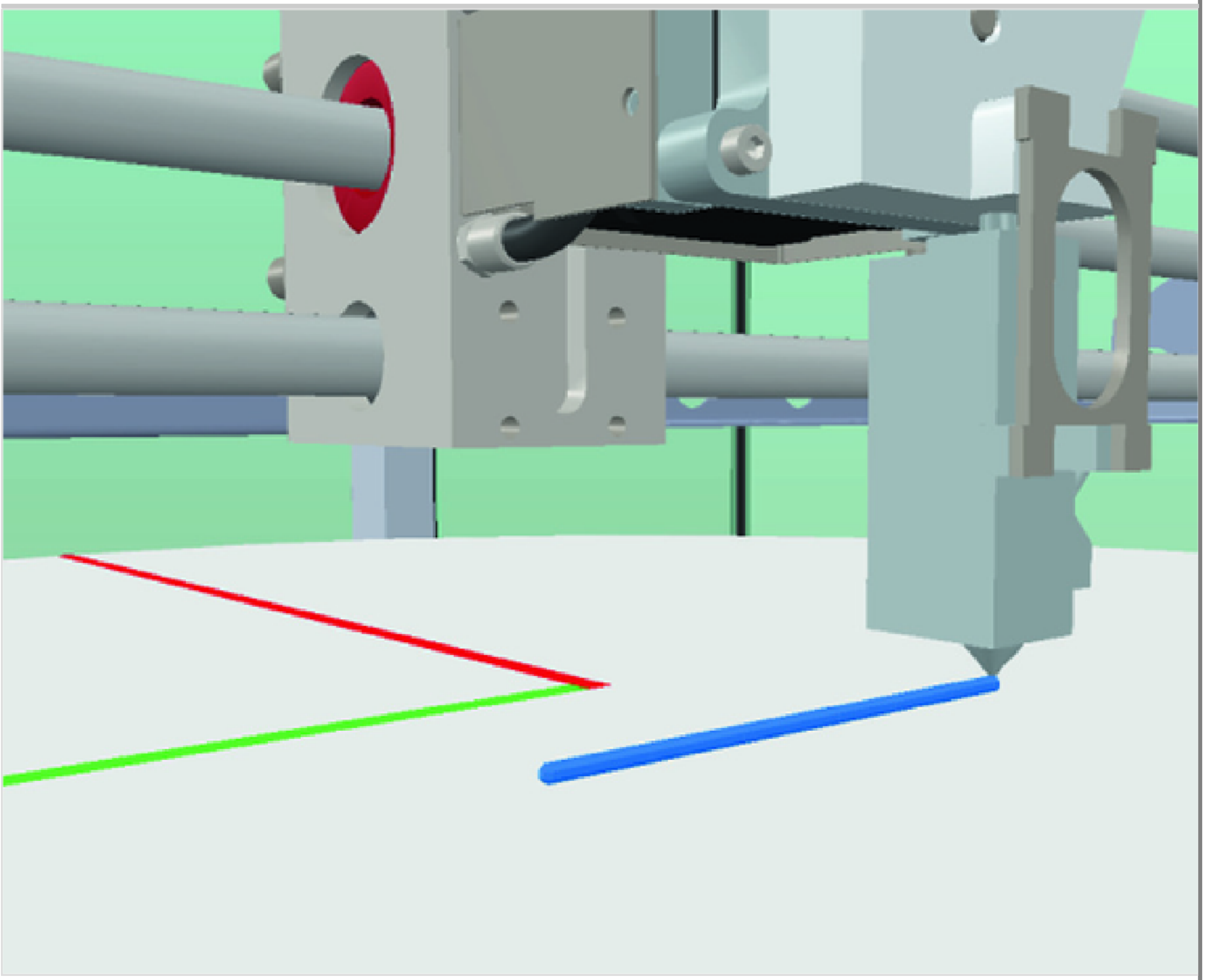
(a) Custom printer .obj files imported as a mesh to the simulator.

(b) Physical parts and joints.

Despite the versatility of the simulator, it does not currently contain a feature to emulate filament deposition. Thus, a new mechanic was implemented based on the drawing shapes feature. The filament is represented by a linear string of shapes drawn in regular intervals and following the extruder tool path, relative to the printer bed, Fig. 6.6. The user is given the possibility to adjust the filament color, size, profile shape, and resolution (i.e., consecutive shapes sparsity).

Fig. 6.6

Simulated filament extrusion



The collision detection between physical bodies is handled directly by the physics engine. The FIBR3DEmul is capable of detecting printer–printer or printer–workpiece collisions and notifying the user about the G-Code instructions that result in collision events.

6.2.3. Printer Control

~~V-Rep~~CoppeliaSim offers different programming approaches to implement a custom controller: embedded scripts, framework nodes, add-ons, remote API clients, and plug-ins. Each approach varies in portability, API completeness, synchronicity, code execution speed, communication lag, etc. To guarantee the best possible performance, no lag and access to the complete API, the C3DPrinter virtual controller was implemented in a ~~V-Rep~~CoppeliaSim plug-in.

As referred in Sect. 6.2.1, the FIBR3DEmul solution is split into the G-Code interpreter and the simulation application. When a G-Code command is parsed, its information is re-marshalled into a formatted JSON message. This message is then sent to the ~~V-Rep~~CoppeliaSim simulation to be executed by the virtual machine. The ~~V-Rep~~CoppeliaSim plug-in is divided into communication, motion control, extrusion, and collision handling.

The communication module connects to the interpreter application via TCP/IP and exchanges messages based on a bidirectional asynchronous communication model. It receives the parsed commands and sends back to the interpreter application the current command being executed and whether a collision event was detected. This module was implemented based on Boost-Asio network libraries.

When a new motion command is read (G0, G1, G2, G3, or G4), information about the properties of the movement as well as other parameters contained in the G-Code is explicitly provided to the plug-in, e.g., target joint positions, trajectory velocities, maximum acceleration, type of interpolation. Each command is internally handled as a 5-axis trajectory, a concept that divides into two other concepts: the geometric path and the velocity profile. The path describes the geometrical shape of the trajectory, which depends on the type of G-Code command, whereas the velocity profile codes the timing law, i.e., how each joint progresses along the path. The result is a vector containing the 5 joint positions for each simulation cycle from the start to the end of the received motion command.

As previously referred, the extrusion mechanic was implemented based on the consecutive drawing of graphical shapes in the simulator. If the current G-Code command being executed includes filament deposition, new shapes are drawn. To mimic the filament behavior in the real printer, new shapes are added according to the tool path of the extruder, which is moved by a 3-joint Cartesian system (*x*-, *y*-, and *z*-axes), while the previous “filament shapes” move as the two rotational joints shift the printer bed. The distance between the position of the extruder in the current and previous simulation cycle is measured, and together with the filament resolution parameter define the number and spacing of shapes drawn in the interval.

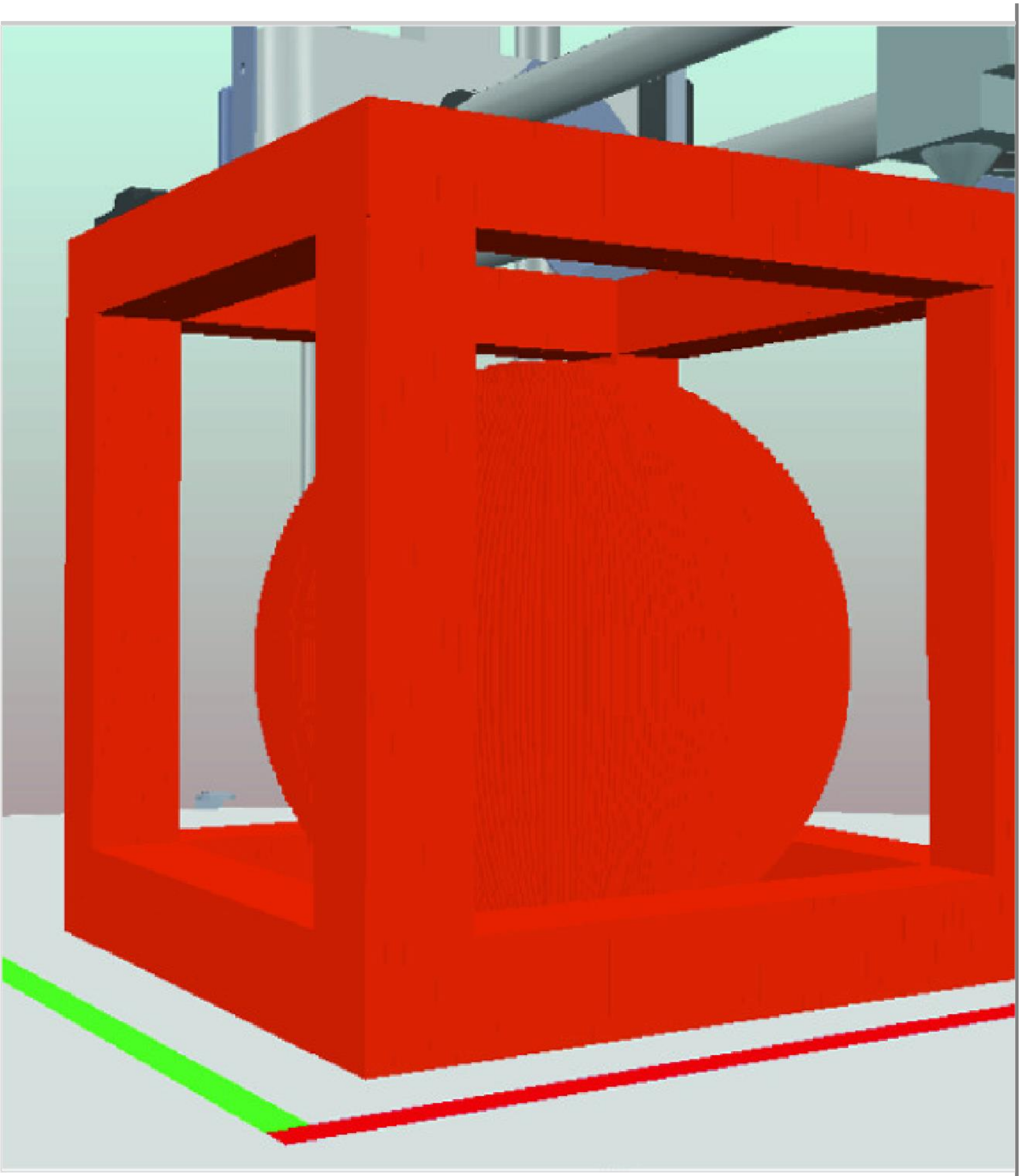
6.2.4. Results and Discussion

The FIBR3DEmul software is currently able to virtually simulate FDM printing processes with 3 and 5 simultaneously moving axis. The G-Code interpreter application is capable of reading and interpreting more than 30 G-Code commands and more than 10 coordinate identifiers. Information explicitly and implicitly comprehended in each line of the G-Code script are parsed and re-marshalled into an explicit JSON message that is forward to the virtual C3DPrinter controller. The interpreter application permits controlling the G-Code script execution to a line-by-line, a block of lines, or the full script. Moreover, as the simulation progresses, the application will notify the user about the current line being executed as well as of any collision events.

On the ~~V-Rep~~CoppeliaSim side, the virtual printer controller receives each message from the interpreter application and generates a vector of joint positions to replicate the motion parametrized in the G-Code command[3]. During the simulation process, the user may adjust parameters of the filament and control the speed of the simulation (down to a quarter or up to 64 times the real-time speed). The FIBR3DEmul was tested with several G-Code files, Fig. 6.7.

Fig. 6.7

Examples of 3D printer models with the FIBR3DEmul



The main output of the simulation is the capacity to predict the final shape of the model enclosed in the 3- or 5-axes G-Code script and check for possible printer–printer or printer–workpiece collisions.

The flexibility of the simulator tool permits in creating and testing custom printer solutions, reducing the development cycle by anticipating problems, and generating appropriate solutions without the risk of damaging equipment. See [73] for an in-depth description of the FIBR3DEmul software.

6.3. Curved Path Planning

This section focuses on curved path planning, where we assume that the object is already optimal oriented; there is no need of supports, and fixed height slicing is to be performed. In the AM manufacturing process, path is the trajectory followed by the machine nozzle, independently of the action being taken. A suitable planning of the nozzle path can bring benefits to the geometry being manufactured, either in what concerns to quality and building time, since quality and building time of the final product can be affected by the deposition rate, layer height, push-out time, suck-back time, and the diameter of the nozzle tip. This lead to the development of different strategies such as raster, zigzag, contour, spiral, fractal space curves, hybrid, medial axis transformation (MAT), direction parallel, and more recently curved layer. See [3] for a review about path planning and its importance in the deposition quality, efficiency, and in decrease of time travelled by the machine nozzle.

The importance given to the path planning stage on the AM process is well documented in the literature, where different strategies to determine the optimal path have been addressed. Regardless of the strategy used, the main goal becomes the quality improvement of the geometries being manufactured using the less possible time.

The emergence of the AM process allowed the manufacturing of more complex geometries/objects. Therefore, the variety of geometries and its wide range of forms became a challenge to the AM process. For example, interior holes are common, which increases the difficulty of the path planning process.

Recently, a new strategy named curved-layered fused deposition modeling (CLFDM) has emerged (see [2, 11, 22, 25, 31, 52]). This section is devoted to present a strategy for curved layer manufacturing considering the previously described 5-axis 3D printer. While the majority of previous works address curved layer path planning in standard 3D printers, the technique here described and introduced in [3] takes advantage of the 5-axis printer to provide a new path planning technique. The printer considers a nozzle that deposits composites material with long or short fibers of reinforced thermoplastic matrix. Therefore, we assume that deposition occurs at a constant speed, i.e., path planning does not need to consider additional parameters during the path planning stage.

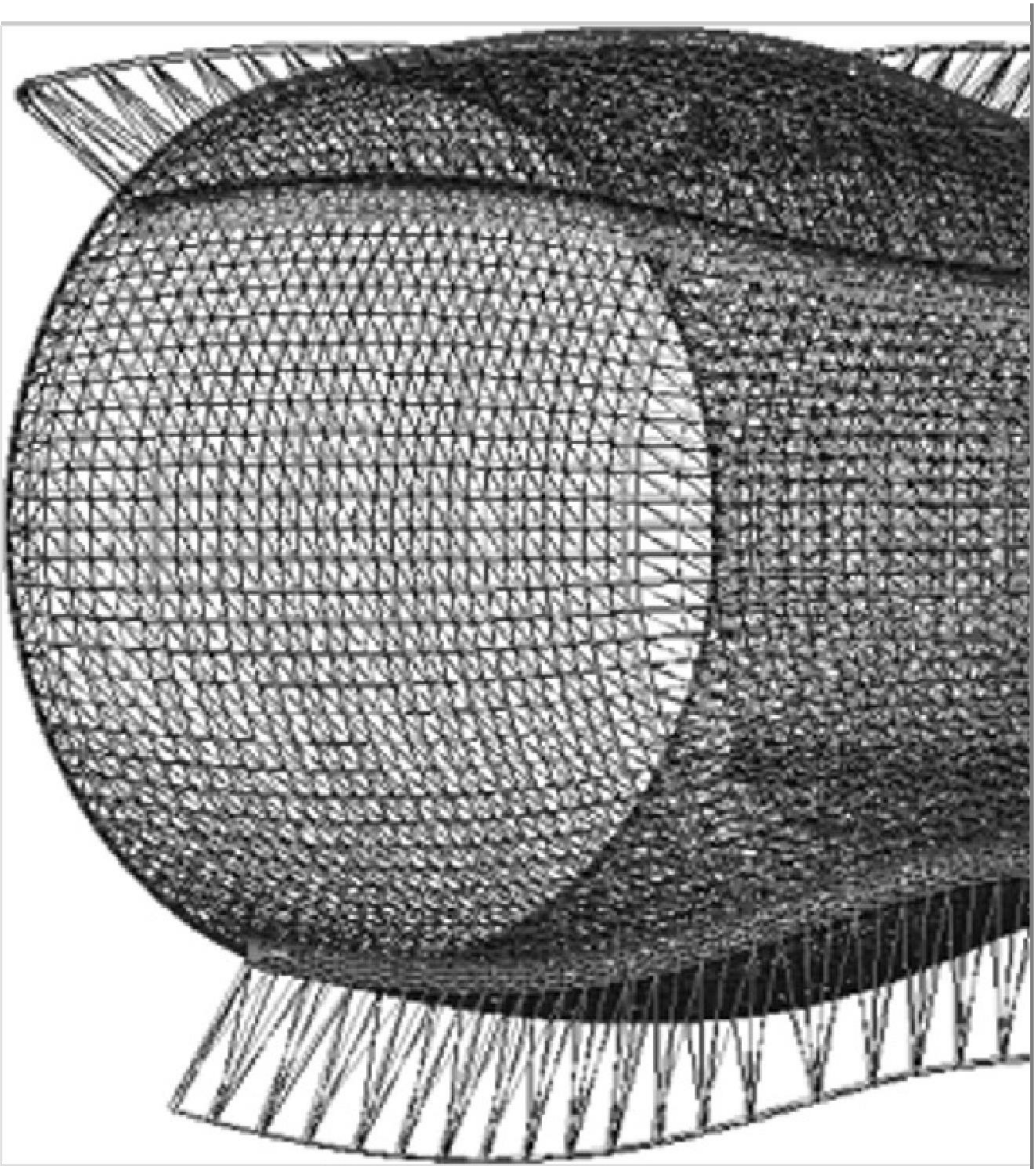
6.3.1. Curved Layer Manufacturing

Standard 3D printers consider slices of the object to be printed. Each slice corresponds to a given *z*-axis layer, and the object is printed by addressing each layer consecutively (with movements on the *x*-, *y*-axes). So, movement on the *z*-axis is typically restricted to change the printing layer. Previous works on curved layer manufacturing consider layers not to be perpendicular to the *z*-axis. Printing such resulting (non-coplanar) layers implies to control the *x*-, *y*-, and *z*-axes simultaneously. For 5-axis printers, curved layer manufacturing is not restricted to the *x*-, *y*-, and *z*-axes, since the advantage of the two additional degrees of freedom should be made to build more complex objects with higher quality (reducing the staircase effect, since we are allowed to print perpendicularly to the object normal direction).

While the proposed strategy to curved layer manufacturing in [3] may be applied to several types of objects, the major interest is to build objects with applications in the aerospace industry, namely objects of shell-type like the one presented in Fig. 6.8, taking advantage of the five degrees of freedom printer to compute by the deposition path. The computed deposition path should be able to produce the object in a continuous way and provide a maximum resistance part by adding curved layers. Additionally, the deposition is done with the nozzle perpendicularly to the object facets, so we can minimize the staircase effect.

Fig. 6.8

Example of a shell-type object



6.3.1.1. Path Planning Using Splines

While the herein exposed approach is devoted to curved path planning, the settings are somehow different from the ones already addressed in the literature. The 5-axis printer provides a flat printing bed, so curved path planning is related to build layers that are not perpendicular to the *z*-axis. We take advantage of interpolation by splines in order to define our deposition path.

A spline is an unidimensional piecewise interpolating function defined by segments between knots (points used to define the spline). Splines are characterized by their simplicity in the definition, easiness to compute and evaluation. These properties make them appealing for this application. While each segment used between two consecutive knots can theoretically be any continuous function, the one with greater interest are polynomials of degree one (linear) or three (cubic). Linear polynomials are simpler to compute and evaluate, but, in general, it makes the spline possibly non-differentiable at the knots. Cubic polynomials are used when smoothness and interpolation of first and second derivatives are requested, at the expense of solving a linear system of equations to determine the spline coefficients. A linear spline considers segments where first-order polynomials (linear functions) are used and a cubic spline considers segments formed by polynomials of degree three. For the herein application, splines with linear and cubic segments are used, depending on the accuracy requested and the shape of the polygons we are interpolating.

Let t_i, $i = 0,\dots,n, t_i < t_{i+1}$, and $f_i = f(t_i)$ be a set of knots (points) and corresponding function values. A spline to interpolate the function f at the given set of *n + 1* knots is composed of *n* segments, each one defined by two consecutive knots. A spline in its general form is given by:

$$s(t) = \begin{cases} s_d^1(t) & t \in [t_0, t_1] \\ s_d^2(t) & t \in [t_1, t_2] \\ \vdots \\ s_d^n(t) & t \in [t_{n-1}, t_n], \end{cases} \qquad 6.9$$

where $s_d^j(t), j = 1,\dots,n$ are the linear ($d = 1$) or cubic ($d = 3$) segments. The spline is well defined if we have $n \geq 2$ for a linear spline and $n \geq 3$ for a cubic spline.

Uniform slicing is considered along the *z*-coordinate, i.e., slicing takes place at the horizontal plane $z = z^\ell$, $\ell = 1,\dots,L$, where *L* is the number of slicing layers. After slicing along the *z*-coordinate, 2D closed polygons representing the object layers are obtained. Each $\left(x_i^\ell, y_i^\ell\right), i = 1, \ldots, n^\ell$ polygon is defined by a set of linear segments resulting from the intersection of a plane with the facets. Let, represent a set of points defining a polygon, for a given ℓ layer. Each polygon is then interpolated by two parametric splines. One interpolates the *x*-coordinate and other the *y*-coordinate, i.e., we have a 2D polygon represented as the parametric function

$$P^\ell(t) = \left(x^\ell(t), y^\ell(t)\right), \tag{6.10}$$

with $x^\ell(t) = s_x^\ell(t)$ and $y^\ell(t) = s_v^\ell(t), t \in [t_0, t_{n_\ell}^\ell]$. From the interpolation conditions we have $P^\ell(t_i^\ell) = (x^\ell(t_i^\ell), y^\ell(t_i^\ell)) = (x_i^\ell, y_i^\ell), i = 1, \ldots,$

$n^\ell, \ell = 1, \ldots, L.$

Slicing along the *z*-coordinate may lead to polygons with a huge number of segments, in special if we have curved objects with high curvature defined by a big number of facets. Since polygons are to be interpolated by splines, a significant number of points can be dropped, as long as the spline continues to provide acceptable accuracy for printing. Reducing the number of points (knots) is an obvious improvement in the splines computation and evaluation time. In [3], authors propose an adapted Douglas–Peucker algorithm [19] that aims a simplification of the closed polygon resulting from the slicing process.

The segments of splines defined by (6.9) are usually of one type: either linear or cubic $(d = 1 \text{ or } d = 3, \forall i = 1, \ldots, n)$. However, fixing the same polynomial degree for all segments may lead to a bad approximation of the polygon. Therefore, the decision about getting a linear or cubic segment is made based on the angle formed by two consecutive line segments (see [3] for details) resulting in a spline with mixed type of segments.

Layers to be made for fixed *z*-axis are computed by considering the splines for each inner polygons (from *t* = 0 to $t = t^{\ell}_{n^{\ell}}, \ell = 1, \ldots, L)$ and layers to be made along the *z*-axis are built by considering the path generated by fixing a *t* value for each layer *ℓ*. See [3] for an example with aerospace shell-type objects.

6.3.1.2. Bed Table Orientation

The slicing process and layer polygons approximation by splines provide a way to path planning along the *x*-, *y*-, and *z*-coordinates; i.e., we can provide the nozzle position in space at any given time step. For curved path planning, the nozzle orientation (or, equivalently, bed table orientation—rotation and tilt) is also important to control, since deposition can be made along the facet normal or its perpendicular direction, helping to minimize the staircase effect.

The intersection of a facet with the slicing plane (if any) provides a line segment, which is used to compute a vector perpendicular to the facet normal vector. Both the facet normal and its perpendicular vectors are of interest to the path planning strategy. If a path is following the facet direction, then the normal perpendicular vector can be used to compute the bed table tilt, while covering the facet can be done by using the facet normal direction.

6.4. Printing Complex Objects

Extra degrees of freedom available in the 5-axis printer allow printing of more complex objects and improvements in the surface quality and support structures reduction. A 5-axis system enables re-orientation of the object during the printing process; being extremely useful for 3D print since overhangs structures may be minimized.

Extensive research literature exists in AM related fields like computational design for AM [20, 27, 33], AM processes [23, 27], process modeling and optimization [7, 14, 53, 58, 72], material science [27], and energy and sustainability [68]. However, additive manufacturing for 3D printing of complex objects by its decomposition into parts only recently has been addressed.

Ding et al. [16] address a new strategy for multi-direction slicing of CAD models in STL format by considering an optimal volume decomposition–regrouping strategy applying a curvature-based volume decomposition method, which decomposes complex objects into sub-volumes using a depth-tree structure.

Wang et al. [70], in order to improve the surface quality in 3D printing, presents a pipeline of algorithms that compute an object decomposition by using the co-compatibility of the facet normal with the printing directions. A 3D Voronoi diagram is computed to consolidate the part's shape. This technique has the particularity that the (manual) assembly order or parts is collision-free, and parts order and direction for assembling were also obtained [71].

Massoni et al. [43] propose a method that automatically decomposes 3D complex models into parts with the goal of lowering overall production cost, and Luo et al. [41] propose a framework called chopper also based on the beam search algorithm.

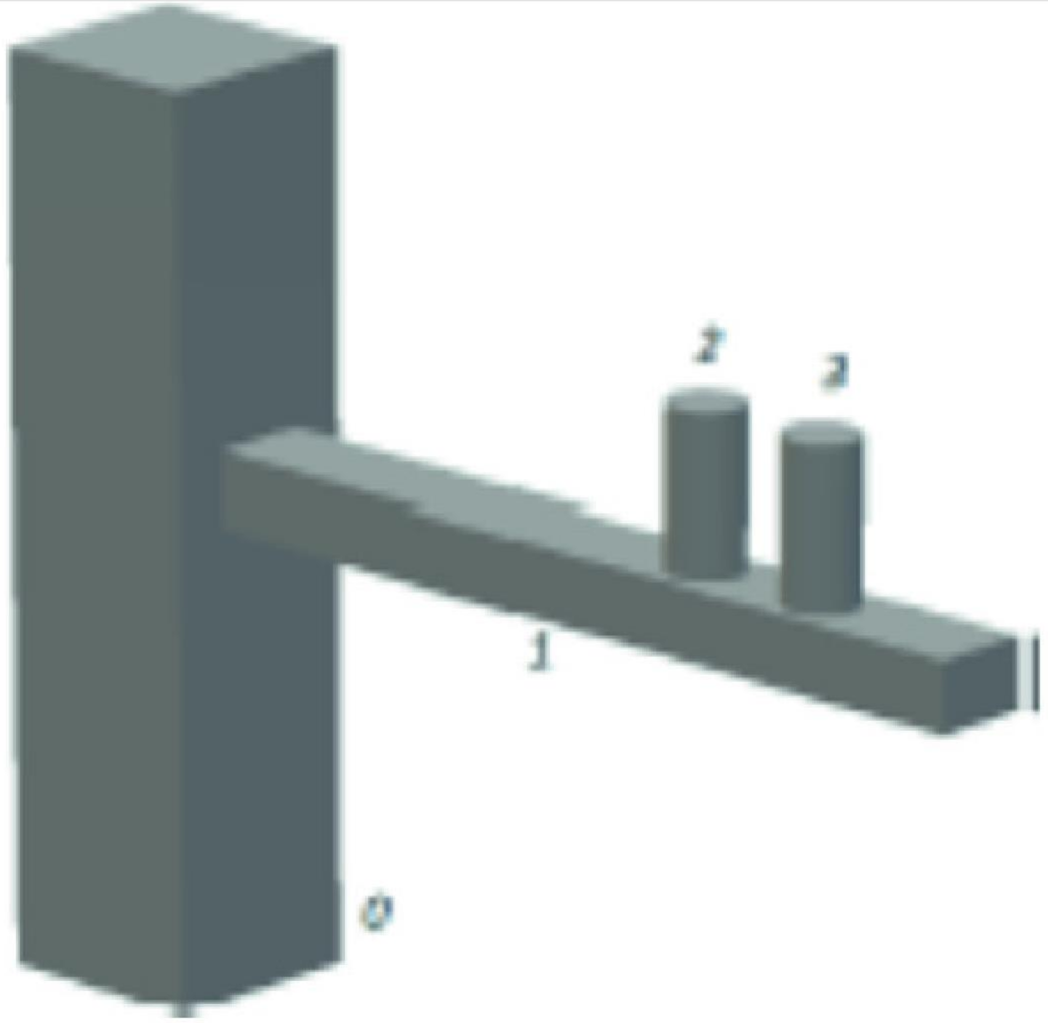
In this section, we describe an approach where complex object is decomposed into simpler parts allowing each part to be printed in an optimal way, reducing the number of supports needed and attaining high final object quality. This technique takes advantage on the 5-axis printer in order to propose an approach that builds complex objects without the user intervention to assembly the parts. The proposed strategy is illustrated with two case studies.

6.4.1. Complex Objects Printing Approach

The approach is illustrated with one example provided in Fig. 6.9. It is assumed that the object is composed of four parts, illustrated in Fig. 6.9a, which are provided in the STL file. From the STL file, we can also establish a printing order for the object and build the corresponding direct graph (Fig. 6.9b). Without loss of generality, we can assume that part 0 is connected to the printer bed (floor), while part 1 is connected with part 0, and parts 2 and 3 are connected with part 1. The object is, therefore, decomposed into $T = 4$ parts represented with four nodes in the graph.

Fig. 6.9

Complex object proposed by Ding et al. [16]



(a) Decomposition of complex object.

Clearly, the complex object considered in Fig. 6.9 cannot be printed in a standard 3D printer without building supports. The proposed strategy considers parts to be printed by taking advantage of previously printed parts and the printer possibility to print in different directions (by tilting and rotating the printer table bed). After computing each part optimal printing direction, the problem reduces to the computation of the optimal printing parts sequence. The resulting sequencing optimization problem must take into consideration of the possible collisions between the printer head and previously built parts. The mathematical formulation is followed by strategy used to solve the resulting optimization problem.

6.4.1.1. Mathematical Model

It is assumed that the set of local and global optima for the part rotation (or slicing direction) is available (see [53] about the optimization problem to be solved). Let K_i be the number of known optima for the rotation of part i, $i = 0, \ldots, T - 1$. We define the set of binary variables $r_{i,k}$, $k = 0, \ldots, K_i$ to be

$$r_{i,k} = \begin{cases} 0, & \text{if rotation } k \text{ of part } i \text{ is not to be considered} \\ 1, & \text{if rotation } k \text{ of part } i \text{ is to be considered.} \end{cases}$$

Clearly a part can only be printed once and natural constraints on variables $r_{i,k}$ are

$$\sum_{k=0}^{K_i} r_{i,k} = 1, \qquad i = 0, \ldots, T - 1 \tag{6.11}$$

To compute the optimal sequencing of parts, the $x_{i,t}$ binary variables are used that indicate if part $i = 0, \ldots, T - 1$ is to be built in the time slot t, where $t = 0, \ldots, T - 1$, i.e.,

$$x_{i,t} = \begin{cases} 0, & \text{if part } i \text{ not to be built at time slot } t \\ 1, & \text{if part } i \text{ to be built at time slot } t \end{cases}$$

Only *T* time slots are necessary, since the worst case corresponds to build all parts sequentially. Clearly, a part may only be built in a time slot, i.e.,

$$\sum_{t=0}^{T-1} x_{i,t} = 1, \quad i = 0,\ldots,T-1. \tag{6.12}$$

From the precedence of parts in the building process, we need to impose two sets of constraints that must be satisfied for every part *i* that precedes part *j*:

$$\sum_{t=0}^{l} x_{i,t} \geq \sum_{k=0}^{l} x_{j,k}, \quad l = 0,\ldots,T-1 \tag{6.13}$$

imposing that part *i* must be built at least in the same time slot, and

$$x_{i,l} \leq \sum_{k=0,k\neq l}^{T-1} x_{j,k}, l = 0,\ldots,T-1 \tag{6.14}$$

removing the possibility of building it at the time slot.

A nonlinear black-box constraint appears when considering that the part's building sequence (together with the part's rotation) provides a feasible building sequence, i.e., the building sequence does not provide any type of collision between the printer (head or table) and previously built parts. The constraint

$$\mathrm{NoCollision}(r,x) = \mathrm{true} \tag{6.15}$$

needs to be addressed so the model produces an optimal solution that leads to a building sequence that is, in fact, possible to be implemented.

A second nonlinear black-box constraint needs to be considered so an optimal solution does not force the need for supports. The constraint

$$\mathrm{NoSupport}(r,x) = \mathrm{true} \tag{6.16}$$

is also considered in the model. This second constraint can be relaxed if, for example, one is willing to accept a solution which leads to the need for supports.

The herein strategy assumes that parts on the printing table, i.e., on base, must be built in the first time slot, so $x_{i,0} = 1$ for all parts is attached to the printer table.

6.4.1.2. Optimization Problems

While the constraints in the previous section provide a mathematical model for a solution of the building sequence of parts, we aim to compute an optimal solution with respect to some performance measure of the printing process.

Let $SE(r_{i,k})$ be the staircase effect, $SA(r_{i,k})$ be the support area, and BT($r_{i,k}$) the building time of part i with rotation k. Based on these performance measures and on the shortest building sequence, we may formulate four objective functions to be used individually (using the one that best fits the application) or in a multi-objective approach.

The shortest building sequence can be obtained by considering the following minimization problem:

$$\min_{r,x} \sum_{t=0}^{T-1} t \cdot \left(\sum_{i=0}^{T-1} x_{i,t} \right), \tag{6.17}$$

subjected to constraints (6.11–6.16).

When considering $P(r_{i,k})$ to be $SE(r_{i,k})$, $SA\ (r_{i,k})$, or BT($r_{i,k}$), we can obtain the best building sequence w.r.t. P by considering the following minimization problems:

$$\min_{r,x} \sum_{i=0}^{T-1} \sum_{k=0}^{K_i} P\left(r_{i,k}\right), \qquad 6.18$$

subjected to constraints (6.11–6.16).

While problems (6.17) and (6.18) have linear objective functions, constraints (6.15) and (6.16) are nonlinear of a black-box-type, which make problems to be nonlinear with black-box-type constraints over binary variables. Therefore, a heuristic that can provide an optimal solution is described in the next section.

6.4.2. Heuristic to Obtain an Optimal Building Sequence

We solve the previously described optimization problem by using a heuristic. The heuristic constructs all solutions and selects the best one according to the objective function in use. The input of Algorithm 1 is a list of pairs with the combination of part and optimal rotations, i.e.,

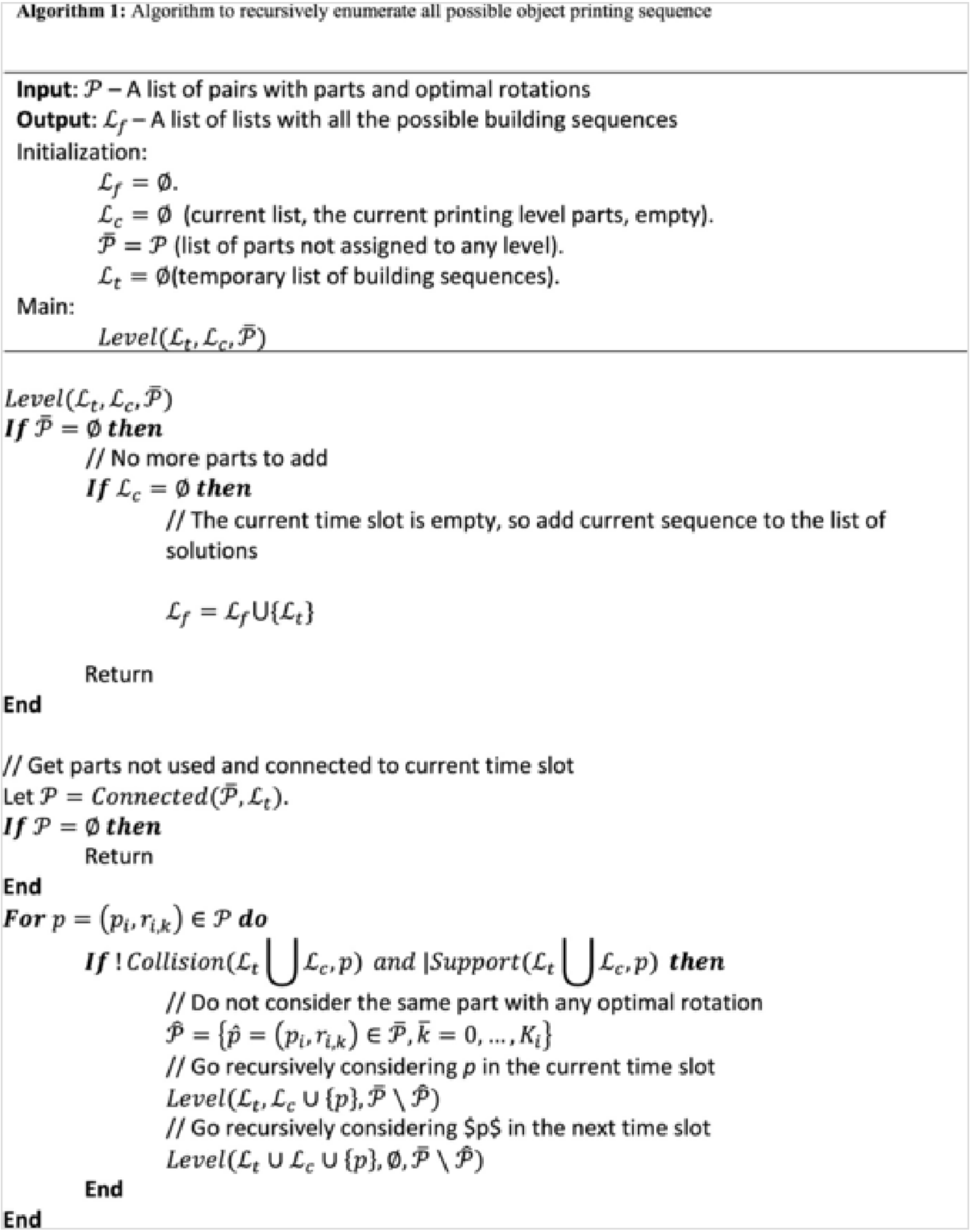
$$P = \{(p_i, r_{i,k})\}, \qquad i = 0, \ldots, T-1, \qquad k = 0, \ldots, K_i;$$

where p_i is the part number.

Algorithm 1 ends with printing parts sequences and corresponding rotations in the list L_f. L_f will be a list of L lists of building sequences.

The algorithm 1 will choose one part p, that belongs to P (initially set as P) and that have connection to the parts already added to the current list L_c, the current printing time slot of parts. Note that for initialization, $L_c = \emptyset$, so the parts connected to the printing table will be selected, i.e., supposing that the base of the object is numbered as p_0 the part $(p_0, r_{0,k})$, $k = 0, \ldots, K_0$ will be chosen.

Whenever a part is to be considered for the current or next time slot, a possible collision or need of support are checked against previously built parts.



Algorithm 1: Algorithm to recursively enumerate all possible object printing sequence

```
Input: $\mathcal{P}$ – A list of pairs with parts and optimal rotations
Output: $\mathcal{L}_f$ – A list of lists with all the possible building sequences
Initialization:
        $\mathcal{L}_f = \emptyset$.
        $\mathcal{L}_c = \emptyset$ (current list, the current printing level parts, empty).
        $\bar{\mathcal{P}} = \mathcal{P}$ (list of parts not assigned to any level).
        $\mathcal{L}_t = \emptyset$(temporary list of building sequences).
Main:
        $Level(\mathcal{L}_t, \mathcal{L}_c, \bar{\mathcal{P}})$

$Level(\mathcal{L}_t, \mathcal{L}_c, \bar{\mathcal{P}})$
$If\ \bar{\mathcal{P}} = \emptyset\ then$
        // No more parts to add
        $If\ \mathcal{L}_c = \emptyset\ then$
                // The current time slot is empty, so add current sequence to the list of
                solutions

                $\mathcal{L}_f = \mathcal{L}_f \cup \{\mathcal{L}_t\}$

        Return
End

// Get parts not used and connected to current time slot
Let $\mathcal{P} = Connected(\bar{\mathcal{P}}, \mathcal{L}_t)$.
$If\ \mathcal{P} = \emptyset\ then$
        Return
End
$For\ p = (p_i, r_{i,k}) \in \mathcal{P}\ do$
        $If\ !Collision(\mathcal{L}_t \bigcup \mathcal{L}_c, p)\ and\ |Support(\mathcal{L}_t \bigcup \mathcal{L}_c, p)\ then$
                // Do not consider the same part with any optimal rotation
                $\hat{\mathcal{P}} = \{\hat{p} = (p_i, r_{i,k}) \in \bar{\mathcal{P}}, \bar{k} = 0, \dots, K_i\}$
                // Go recursively considering $p$ in the current time slot
                $Level(\mathcal{L}_t, \mathcal{L}_c \cup \{p\}, \bar{\mathcal{P}} \setminus \hat{\mathcal{P}})$
                // Go recursively considering $p$ in the next time slot
                $Level(\mathcal{L}_t \cup \mathcal{L}_c \cup \{p\}, \emptyset, \bar{\mathcal{P}} \setminus \hat{\mathcal{P}})$
        End
End
```

The collision $(L_t \cup L_c, p)$ and support $(L_t \cup L_c, p)$ functions return true if there is a collision or the need of support when building part *p* after parts in $L_t \cup L_c$ were built, respectively.

6.4.3. Results

This section presents a case study with the complex object already addressed in Fig. 6.9. For the sake of simplicity, we are not considering all the local and global optima of the individual parts rotations (e.g., part 0 and part 1 has 6 optimal rotations corresponding to getting each face of the part down, while parts 2 and 3 have two global optima). For the illustration, we consider P = {(0, (0°, 0°)), (0, (0°, 90°)), (1, (0°, 0°)), (1, (0°, 90°)), (2, (0°, 0°)), (3, (0°, 0°))} and precedences given in Fig. 6.9b.

Algorithm 1 considers 16 building sequences:

$$L_1 = \left\{ \{(0,(0^\circ,0^\circ))\};\{(1,(0^\circ,0^\circ))\}; \{(2,(0^\circ,0^\circ)),(3,(0^\circ,0^\circ))\} \right\}$$

$$L_2 = \left\{ \{(0,(0^\circ,0^\circ))\};\{(1,(0^\circ,0^\circ))\};\{(2,(0^\circ,0^\circ))\}; \{(3,(0^\circ,0^\circ))\} \right\}$$

$$L_3 = \left\{ \{(0,(0^\circ,0^\circ))\};\{(1,(0^\circ,0^\circ))\}; \{(3,(0^\circ,0^\circ)),(2,(0^\circ,0^\circ))\} \right\}$$

$$L_4 = \left\{ \{(0,(0^\circ,0^\circ))\};\{(1,(0^\circ,0^\circ))\};\{(3,(0^\circ,0^\circ))\}; \{(2,(0^\circ,0^\circ))\} \right\}$$

$$L_5 = \left\{ \{(0,(0^\circ,0^\circ))\};\{(1,(0^\circ,90^\circ))\}; \{(2,(0^\circ,0^\circ)),(3,(0^\circ,0^\circ))\} \right\}$$

$$L_6 = \left\{ \{(0,(0^\circ,0^\circ))\};\{(1,(0^\circ,90^\circ))\};\{(2,(0^\circ,0^\circ))\}; \{(3,(0^\circ,0^\circ))\} \right\}$$

$$L_7 = \left\{ \{(0,(0^\circ,0^\circ))\};\{(1,(0^\circ,90^\circ))\}; \{(3,(0^\circ,0^\circ)),(2,(0^\circ,0^\circ))\} \right\}$$

$$L_8 = \left\{ \{(0,(0^\circ,0^\circ))\};\{(1,(0^\circ,90^\circ))\};\{(3,(0^\circ,0^\circ))\}; \{(2,(0^\circ,0^\circ))\} \right\}$$

$$L_9 = \left\{ \{(0,(0^\circ,90^\circ))\};\{(1,(0^\circ,0^\circ))\};\{(2,(0^\circ,0^\circ)),(3,(0^\circ,0^\circ))\} \right\}$$

$$L_{10} = \left\{ \{(0,(0^\circ,90^\circ))\};\{(1,(0^\circ,0^\circ))\};\{(2,(0^\circ,0^\circ))\};\{(3,(0^\circ,0^\circ))\} \right\}$$

$$L_{11} = \left\{ \{(0,(0^\circ,90^\circ))\};\{(1,(0^\circ,0^\circ))\};\{(3,(0^\circ,0^\circ)),(2,(0^\circ,0^\circ))\} \right\}$$

$$L_{12} = \left\{ \{(0,(0^\circ,90^\circ))\};\{(1,(0^\circ,0^\circ))\};\{(3,(0^\circ,0^\circ))\};\{(2,(0^\circ,0^\circ))\} \right\}$$

$$L_{13} = \left\{ \{(0,(0^\circ,90^\circ))\};\{(1,(0^\circ,90^\circ))\};\{(2,(0^\circ,0^\circ)),(3,(0^\circ,0^\circ))\} \right\}$$

$$L_{14} = \left\{ \{(0,(0^\circ,90^\circ))\};\{(1,(0^\circ,90^\circ))\};\{(2,(0^\circ,0^\circ))\};\{(3,(0^\circ,0^\circ))\} \right\}$$

$$L_{15} = \left\{ \{(0,(0^\circ,90^\circ))\};\{(1,(0^\circ,90^\circ))\};\{(3,(0^\circ,0^\circ)),(2,(0^\circ,0^\circ))\} \right\}$$

$$L_{16} = \left\{ \{(0,(0^\circ,90^\circ))\};\{(1,(0^\circ,90^\circ))\};\{(3,(0^\circ,0^\circ))\};\{(2,(0^\circ,0^\circ))\} \right\}$$

Assuming that part 3 cannot be built after being built part 2 (because the printer head will collide with part 2 when building part 3 due to not enough space between parts), and vice versa, the sequences in lists $L_2, L_4, L_6, L_8, L_{10}, L_{12}, L_{14}, \text{and } L_{16}$ are not feasible. The sequences in lists $L_1, L_3, L_9, \text{and}\, L_{11}$ are also not feasible due to the need of supports when building part 1.

Therefore, Algorithm 1 terminates with $L_F = \{L_5, L_7, L_{13}, L_{15}\}$, where $L_5 = L_7$ and $L_{13} = L_{15}$, since they correspond to the same building sequence.

The sequence in L_5 list corresponds to build part 0 without any rotation, apply a rotation of 90° to build part 1 and build simultaneously parts 2 and 3 without any rotation. Sequence in list L_{13} corresponds to rotate parts 0 and 1 to be built and build parts 2 and 3 without rotation. Sequences in lists L_5 and L_{13} provide the same objective function values for (6.17), SA and SE, being the sequence in list L_{13} the optimal w.r.t. the BT metric.

6.5. Non-destructive Inspection Path Planning

Despite the high level of fidelity and performance of 3D printers, it is still necessary to ensure that manufactured objects meet quality requirements, especially when used in areas requiring high standards of safety and reliability. Non-destructive tests (e.g., thermographic camera) may be used in industrial inspection machines to determine if the object was built according to industrial requirements. While inspection of 3D printed objects is a relatively common task, camera movement trajectories is not trivial and standard tools do not provide a single and efficient method for this purpose, especially in objects with complex structures. Techniques to compute adequate object inspection trajectories are, therefore, of most importance.

The approach described in Sect. 6.3 may be used to compute inspection trajectories for complex objects created on a 3D printer when using non-destructive tests. An inspection machine with five degrees of freedom is considered and described in Chap. 8 . The machine is able to perform the standard XYZ movements and has two degrees of freedom in the inspection head/camera. Given a set of inspection parameters (e.g., sampling distance to the object and number of samples), the main goal is to compute the inspection trajectory that minimizes the total inspection time, while avoiding collisions between the inspection head and the object under analysis. While some recent works on non-destructive tests are available (see [1, 15, 17, 45], and [13]), none of them takes advantage of the CAD model available in a STL file format. As described in [55], through the STL file is possible to provide a CNC inspection path planning, with the additional advantage of the inspection head to be perpendicular to the facet being inspected.

The CAD route to inspection is considered to be similar to the one for printing the object, i.e., we consider the object to be provided as a STL file, obtained from the CAD model of the object. The use of the same CAD information provides an additional advantage over traditional inspection strategies, since the user obtains the inspection information as a sub-product of the object CAD route for printing. After the slicing process, a projected polygon is created according to the normal vector of each point of the polygon. Through this created points, it is computed by the inspection path.

To produce the G-Code for inspection, one requests a number of parameters to be provided. These parameters are related to the type of inspection to be carried on. The object is sliced along the *z*-axis (or slicing direction) originating a set of (closed) polygons for each layer, being the distance between layers a requested parameter for inspection. The inspection distance to the object and the sampling distance are two parameters to be considered when generating the inspection places. The set of points followed by the inspection head defines the inspection path. The inspection path is composed of a path along with the current layer/slice with movements along the *z*-axis. The inspection distance, also known as lift-off, represents the distance to which the inspection head must be placed in order to avoid collisions with the object. Another parameter is the sampling distance, defining the distance between samples in the same layer. The lift-off, sampling, and slicing distances define the area to be captured by the inspection head. The computational complexity of the optimization algorithms used for computing the inspection path is highly dependable on these parameters, since they define the (possibly huge) set of inspection points. The inspection path complexity can also increase for high complex objects such as the ones composed by more than one inspection polygon per layer.

The facet normal is used as the inspection machine head direction, if an object perpendicular position of the head is to be obtained. The complete inspection path is formed by inspection paths for each layer obtained from slicing the object. The complexity in generating the path is highly dependable on the object complexity, majorly due to possible collisions of the inspection head with the object. In [55], authors generate the inspection points by using the splines obtained after slicing the object. These inspection points are then validated by checking for possible collisions with the inspection head. Additionally, traveling from one inspection point to another may not be possible due to collisions of the head with the object.

An inspection path corresponds to a solution of the traveling salesman problem, considering a graph whose nodes/cities are valid inspection points and arcs are valid links between nodes. The aim is to visit only once all the inspection points following valid links. Graphs are computed for each layer of the object, obtained from the slicing procedure. The initial node for the first graph/layer to be considered is also to be computed, i.e., the start city of the traveling salesman problem is not fixed, and returning to the start city is not mandatory, since we aim to proceed to the next layer/graph without visiting the start valid inspection point. The path must take into account the inspection machine characteristics in order to minimize the total inspection time. We assume arc costs to be proportional to the total travel time; i.e., Euclidean distance between arc points is used to represent the arc cost.

Dropping not valid inspection points and links still lead to a NP-hard problem to be solved for the (optimal) inspection path. Due to the problem complexity and diversity of objects to be inspected, authors in [55] proposed several algorithms that lead to an optimal or near-optimal solution inspection path.

The combinatorial approach (CombF) generates all possible inspection paths to find the inspection path with minimum cost. Given a set $N = \{i_{p1},\ldots,i_{pn}\}$ with n valid inspection points and a symmetric time cost matrix $T_{ij}, (i,j) \in \{1,\ldots,n\}$, and $T_{ii} = 0$, the main goal is to determine the permutation $\pi \in P_n = \{p : \{1,\ldots,n\} \rightarrow \{1,\ldots,n\}\}$ where the objective function $f : P_n \rightarrow R$ 6.19

$$f(\pi) = \sum_{i=1}^{n-1} T_{\pi(i),\pi(i+1)}$$

has its minimum value. The main drawback of this approach is related to the possible huge number of possible inspection paths. The complexity of such an algorithm is exponential in the number of nodes and arcs of the generated graph. However, if the number of nodes and arcs is modest, the enumeration of all inspection paths is still possible and an optimal solution is obtained.

A mathematical integer programming (MIP) model is possible to be used to obtain an optimal inspection path. To compute an optimal inspection path is equivalent to solve a classic traveling salesman problem, since the main goal is to visit all inspection points/nodes minimizing the total travel time. The implemented mathematical model is based on the Miller–Tucker–Zemlin [6, 67] formulation. For the first layer, the mathematical formulation assumes, without loss of generality, an arbitrary random starting inspection point and obtains a closed path. Recall that the inspection path does not need to return to the initial inspection point and should proceed to the next layer. In order to consider a path that does not return to the initial inspection point, we need to add a dummy inspection point d to the set N, where $N^t = N \cup \{d\}$. This point is connected to any other inspection point with zero cost, and, in the same way, any inspection point is connected with the dummy point with cost zero. From the optimal solution of the MIP, one is able to obtain the optimal inspection path by removing the links associated with the d inspection point [4]. For the remaining layers, the starting inspection point is the one closer to the previous layer end inspection point.

The combinatorial and MIP approaches have the theoretical guarantee to obtain an optimal inspection path. However, these two approaches need an exponential amount of time to obtain such an optimal solution. So, a greedy heuristic approach is also proposed in [55] that are able to obtain a near-optimal solution by using less computational resources. The nearest neighbor heuristic (NNH) is a simple heuristic, since it does not take into account a global view of the problem. The algorithm selects an initial inspection point to start the inspection path and successively considers the closest inspection point; until all inspection points are included in the inspection path. The procedure is repeated for all initial inspection points to guarantee a valid solution for the first layer.

The k-nearest neighbor heuristic approach (k-NNGH) is a generalization of the greedy heuristic approach, where we consider more than the current best arc to form several inspection paths. At each inspection point, we only consider the best k arcs to construct all possible inspection paths. Since each inspection point may only have a maximum of $n - 1$ arcs connected to it, we obtain the combinatorial approach when we take $k \geq n - 1$, and the greedy heuristic approach when $k = 1$. While the greedy heuristic and combinatorial approaches are particular cases of the approach presented here, and we choose to describe them separately since they lead to somehow different implementations.

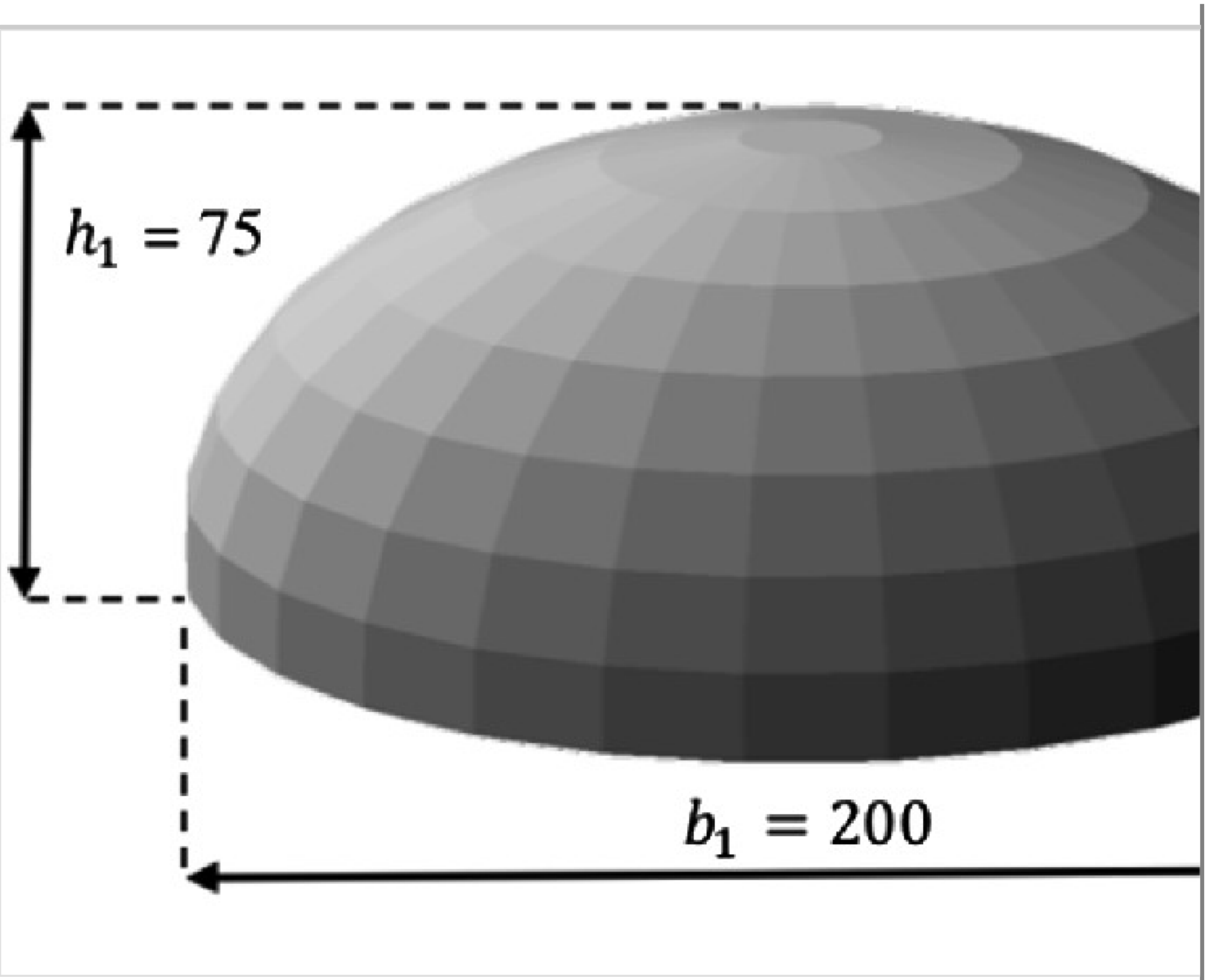
Inspection points are obtained from the parametric spline that represents each polygon, obtained from the slicing procedure. Each spline is, therefore, a parametric function that starts at $t = 0$ and ends at t_{end} where t_{end} corresponds to the perimeter of the polygon. A natural order for the inspection path is to follow the order which occurs in the inspection point computation, i.e., to use the same path as the one used for the printing process in [3]. A possible strategy is to consider a predefined number (α) of valid inspection points to define a sub-path taking into consideration the spline orientation. The *k*-nearest neighbor sub-path heuristic (*k*-NNSH) approach consists of building the inspection path by considering the possible combinations of sub-paths.

In order to demonstrate the results obtained through the different approaches, two different STL files are computed. The main objective of both case studies is to determine a valid inspection trajectory while avoiding collisions between the object and the inspection head/camera.

The first STL file represents a simple three-dimensional object with a base (b_1) with 200 mm and a height (h_1) of 75 mm (see Fig. 6.10).

Fig. 6.10

Best trajectory for the simple object

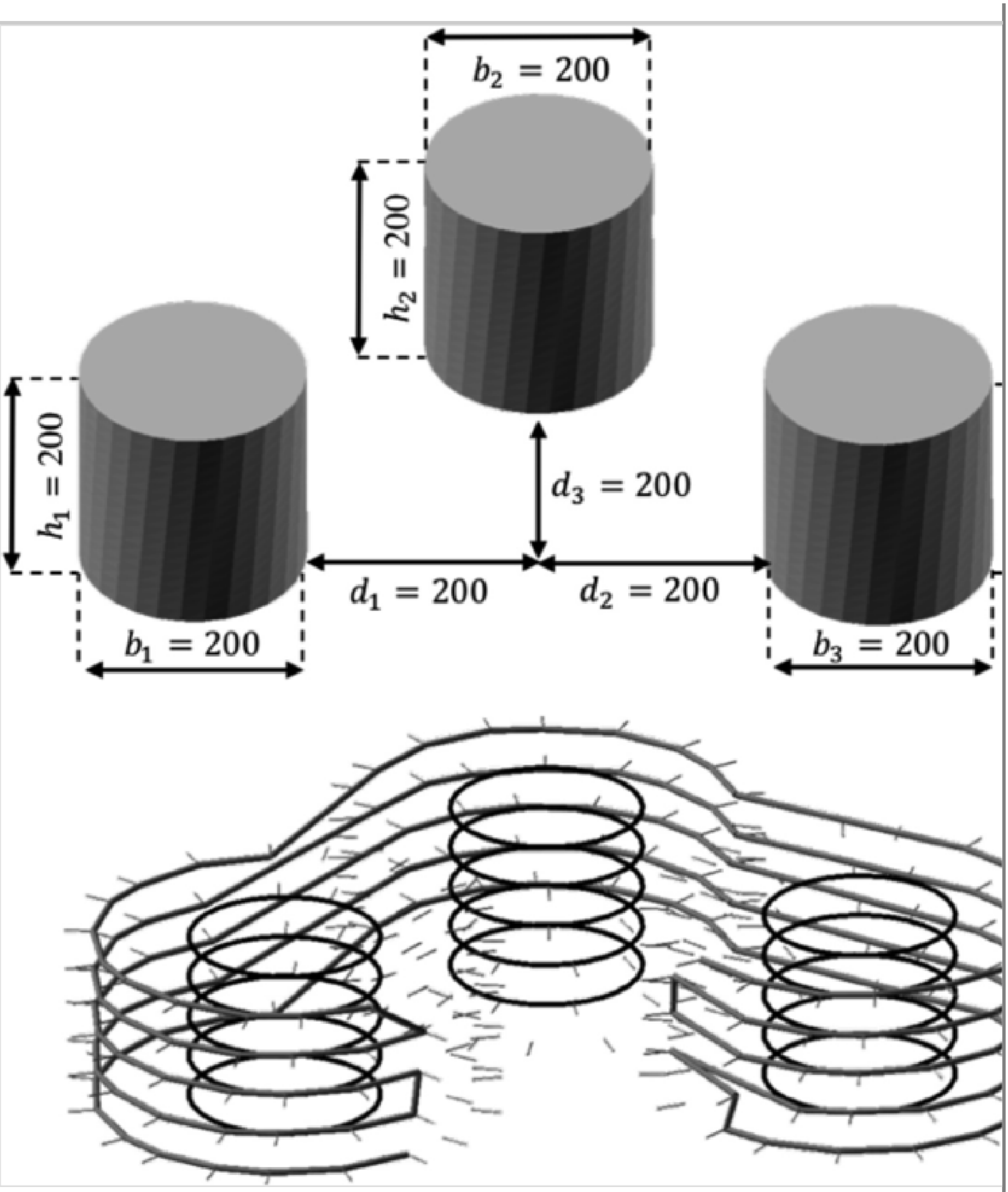


Despite being a simple object, the concave aspect of it makes the values of the normal differ along with the underlying layers. Indeed, for each inspection point, there is a different positioning of the inspection head along the trajectory. This first object is inspected at a distance of 50 mm (liftoff) and samples are taken every 20 mm. The slicing process is performed every 50 mm. All the approaches (*MIP*, *CombF*, *NNH*, *2-NNGH*, and *1-NNSH* with $\alpha = 3$) leads to the same solution with an inspection time of 2298.20 ms. The trajectories are generated in excellent computational times (less than 20 ms considering the time to generate the graph and to compute each approach).

The other STL file defines a complex three-dimensional object, since at least one layer is composed by more than one inspection polygon. Figure 6.11 shows three identical cylinders spaced from each other. Since the cylinders are not sufficiently spaced, it is not possible for the inspection camera to circumvent each polygon individually. For this reason, a collision area is created where no inspection point can be inspected. Although the number of links connecting the different polygons is substantially reduced by this collision area, due to the number of polygons present in each layer, the number of valid links is still relevant from a computational point of view, both in terms of resources and in terms of time. All the cylinders have a diameter of 200 mm (b_1, $b_{2,}$ and b_3) and a height of 200 mm (h_1, $h_{2,}$ and h_3). The cylinders are spaced by d_1, $d_{2,}$ and d_3 as shown in Fig. 6.11.

Fig. 6.11

Best trajectory for the complex object



For this case, a liftoff of 115 mm and a sampling distance of 40 mm were considered. The slicing was performed every 45 mm. The inspection trajectory obtained is the same for all approaches requiring 13,279.56 ms to inspect the complete object. Although the approaches converge on the same solution (Fig. 6.11), some of them use more computing resources than others. This case generates about 27 valid inspection points per layer that are combining and generating between 100 and 115 valid links per layer. Due to the relative complexity of the graph, it takes 484 ms to be generated, since it considers the exclusion of points that present collisions and also direct links between them. Indeed, these do not represent a valid trajectory between two valid inspection points. The *MIP*, *NNH*, and *k-NNSH* approaches with $k = 1$ and $\alpha = 4$ are the fastest to generate a valid inspection trajectory taking less than 30 ms. The combinatorial approach is the slowest taking 577,625 ms, and, with an intermediate time, there is the *k-NNGH* heuristic approach with $k = 2$ (1109 ms).

In these cases, the proposed approaches lead to the same solution for each object under analysis. This situation may not occur as reported in other case studies described by the authors in [55]. The referred approaches may be more suitable for one object than another due to the complexity of the object and to the parameter settings at the computation moment. The complexity of the algorithms is highly dependable on the number of valid inspection points and links that are computed. The definition of the parameters may strongly influence the cardinality of the aforementioned sets (inspection points and links). Hypothetically and theoretically define the complexity without a concrete object may not conduct to valid conclusions. In this way, there must be some user sensitivity so that the parameters are correctly configured according to the object inspection needs.

References

1. Akhloufi, M.A., Guyon, Y., Castanedo, C.-I., Bendada, A.: Three-dimensional thermography for non-destructive testing and evaluation. Quant. InfraRed Thermogr. J. **14**(1), 79–106 (2017)

2. Allen, R.J.A., Trask, R.S.: An experimental demonstration of effective curved layer fused filament fabrication utilising a parallel deposition robot. Addit. Manuf. **8**, 78–87 (2015)

3. Duarte, J., Espírito Santo, I., Monteiro, T., Vaz, A.I.F.: Curved

layer path planning on a 5-axis 3d printer (submitted, 2019

4. Applegate, D.L., Bixby, R.E., Chvatál, V., Cook, W.J.: The Problem, pp. 1–58. Princeton University Press (2006)

5. Bacchewar, P.B., Singhal, S.K., Pandey, P.M.: Statistical modelling and optimization of surface roughness in the selective laser sintering process. Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. **221**(1), 35–52 (2007)

6. Bektaş, T., Gouveia, L.: Requiem for the Miller–Tucker–Zemlin subtour elimination constraints? Eur. J. Oper. Res. **236**(3), 820–832 (Aug 2014)

7. Bikas, H., Stavropoulos, P., Chryssolouris, G.: Additive manufacturing methods and modelling approaches: a critical review. Int. J. Adv. Manuf. Technol. **83**(1), 389–405 (2016)

8. Brika, S.E., Fiona Zhao, Y., Brochu, M., Mezzetta, J.: Multi-objective build orientation optimization for powder bed fusion by laser. J. Manuf. Sci. Eng. **139**(11), 111011 (2017)

9. Canellidis, V., Giannatsis, J., Dedoussis, V.: Genetic-algorithm-based multi-objective optimization of the build orientation in stereolithography. Int. J. Adv. Manuf. Technol. **45**(7–8), 714–730 (2009)

10. Canellidis, V., Dedoussis, V., Mantzouratos, N., Sofianopoulou, S.: Pre-processing methodology for optimizing stereolithography apparatus build performance. Comput. Ind. **57**(5), 424–436 (2006)

11. Chakraborty, D., Aneesh Reddy, B., Roy Choudhury, A.: Extruder path generation for curved layer fused deposition modeling. Comput.-Aided Des. **40**(2), 235–243 (2008)

12. Cheng, W., Fuh, J.Y.H., Nee, A.Y.C., Wong, Y.S., Loh, H.T., Miyazawa, T.: Multi-objective optimization of part building orientation in stereolithography. Rapid Prototyp. J. **1**(4), 12–23 (1995)

13. Ciampa, F., Mahmoodi, P., Pinto, F., Meo, M.: Recent advances in active infrared thermography for non-destructive testing of aerospace components. Sensors **18**(2) (2018)

14. Das, P., Chandran, R., Samant, R., Anand, S.: Optimum part build orientation in additive manufacturing for minimizing part errors and support structures. Procedia Manuf. **1**, 343–354 (2015). In: 43rd North American Manufacturing Research Conference, NAMRC 43, 8–12 June 2015, UNC Charlotte, North Carolina, USA

15. Ding, D., Pan, Z., Cuiuri, D., Li, H., Larkin, N.: Adaptive path planning for wire-feed additive manufacturing using medial axis transformation. J. Clean. Prod. **133**, 942–952 (2016)

16. Ding, D., Pan, Z., Cuiuri, D., Li, H., Larkin, N., van Duin, S.: Automatic multi-direction slicing algorithms for wire based additive manufacturing. Robot. Comput.-Integr. Manuf. **37**, 139–150 (2016)

17. Djupkep Dizeu, F.B., Hesabi, S., Laurendeau, D., Bendada, A.: Three-dimensional non-destructive evaluation of cylindrical objects (pipe) using an infrared camera coupled to a 3d scanner. In: The Largest Open Access Portal of Nondestructive Testing (NDT) Proceedings (2016)

18. Dolenc, A., Makela, I.: Slicing procedures for layered manufacturing techniques. Comput. Aided Des. **26**(2), 119–126 (1994)

19. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of

the number of points required to represent a digitized line or its caricature. Cartogr. Int. J. Geogr. Inf. Geovisualization **10**(2), 112–122 (1973)

20. Gao, W., et al.: The status, challenges, and future of additive manufacturing in engineering. Comput. Aided Des. **69**, 65–89 (2015)

21. Gross, B.C., Erkal, J.L., Lockwood, S.Y., Chen, C., Spence, D.M.: Evaluation of 3d printing and its potential impact on biotechnology and the chemical sciences. Anal. Chem. **86**(7), 3240–3253 (2014)

22. Guan, H.W., Savalani, M.M., Gibson, I., Diegel, O.: Influence of fill gap on flexural strength of parts fabricated by curved layer fused deposition modeling. Procedia Technol. **20**, 243–248 (2015)

23. Guo, N., Leu, M.C.: Additive manufacturing: technology, applications and research needs. Front. Mech. Eng. **8**(3), 215–243 (2013)

24. Gurrala, P.K., Prakash Regalla, S.: Multi-objective optimisation of strength and volumetric shrinkage of FDM parts: a multi-objective optimization scheme is used to optimize the strength and volumetric shrinkage of FDM parts considering different process parameters. Virtual Phys. Prototyp. **9**(2), 127–138 (2014)

25. Huang, B., Singamneni, S.: Curved layer adaptive slicing (clas) for fused deposition modelling. Rapid Prototyp. J. **21**(4), 354–367 (2015)

26. Hussein, A., Hao, L., Yan, C., Everson, R., Young, P.: Advanced lattice support structures for metal additive manufacturing. J. Mater. Process. Technol. **213**(7), 1019–1026 (2013)

27. Jaiswal, P., Patel, J., Rai, R.: Build orientation optimization for additive manufacturing of functionally graded material objects. Int. J. Adv. Manuf. Technol. **96**(1), 223–235 (Apr 2018)

28. Jibin, Z.: Determination of optimal build orientation based on satisfactory degree theory for rpt. In: Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05), p. 6 (2005)

29. Jin, Y., Du, J., He, Y., Fu, G.: Modeling and process planning for curved layer fused deposition. Int. J. Adv. Manuf. Technol., pp. 1–13 (2016)

30. Jin, Y., Du, J., Ma, Z., Liu, A., He, Y.: An optimization approach for path planning of high-quality and uniform additive manufacturing. Int. J. Adv. Manuf. Technol., pp. 1–12 (2017)

31. Jin, Y., Jianke, D., He, Y., Guoqiang, F.: Modeling and process planning for curved layer fused deposition. Int. J. Adv. Manuf. Technol. **91**(1), 273–285 (2017)

32. Jung, J., Ahluwalia, R.S.: Nc tool path generation for 5-axis machining of free formed surfaces. J. Intell. Manuf. **16**(1), 115–127 (2005)

33. Ko, H., Moon, S.K., Hwang, J.: Design for additive manufacturing in customized products. Int. J. Precis. Eng. Manuf. **16**(11), 2369–2375 (2015 Oct)

34. Kulkarni, P., Marsan, A., Dutta, D.: A review of process planning techniques in layered manufacturing. Rapid Prototyp. J. **6**(1), 18–35 (2000)

35. Kumar, A., Banerjee, A.G., Paul, S., Choudhury, A.R.: Maximization of slice height with uniformity in surface

roughness in the direct slicing of freeform surfaces. Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. **217**(6), 765–777 (2003)

36. Lan, P.-T., Chou, S.-Y., Chen, L.-L., Gemmill, D.: Determining fabrication orientations for rapid prototyping with stereolithography apparatus. Comput. Aided Des. **29**(1), 53–62 (1997)

37. Lee, K.H., Choi, K.: Generating optimal slice data for layered manufacturing. Int. J. Adv. Manuf. Technol. **16**(4), 277–284 (2000)

38. Li, A., Zhang, Z., Wang, D., Yang, J.: Optimization method to fabrication orientation of parts in fused deposition modeling rapid prototyping. In: 2010 International Conference on Mechanic Automation and Control Engineering, pp. 416–419. IEEE, New York (2010)

39. Li, H., Wang, T., Sun, J., Yu, Z.: The adaptive slicing algorithm and its impact on the mechanical property and surface roughness of freeform extrusion parts. Virtual Phys. Prototyp. **11**(1), 27–39 (2016)

40. Lim, S., Buswell, R.A., Valentine, P.J., Piker, D., Austin, S.A., De Kestelier, X.: Modelling curved-layered printing paths for fabricating large-scale construction components. Addit. Manuf. **12**(Part B), 216–230 (2016)

41. Luo, L., Baran, I., Rusinkiewicz, S., Matusik, W.: Chopper: partitioning models into 3D-printable parts. ACM Trans. Graph. (Proc. SIGGRAPH Asia), **31**(6) (2012 Dec)

42. Masood, S.H., Rattanawong, W., Iovenitti, P.: A generic algorithm for a best part orientation system for complex parts in rapid prototyping. J. Mater. Process. Technol. **139**(1–3), 110–116 (2003)

43. Massoni, B.R., Campbell, M.I.: A decomposition method for efficient manufacturing of complex parts. Comput.-Aided Des. Appl. **14**(6), 705–719 (2017)

44. Matos, M.A., Rocha, A.M.A.C., Costa, L.A., Pereira, A.I.: A multi-objective approach to solve the build orientation problem in additive manufacturing. In: International Conference on Computational Science and Its Applications, pp. 1–15. Springer, Berlin (2019) (in press)

45. Mineo, C., Pierce, S.G., Wright, B., Nicholson, P.I., Cooper, I.: Robotic path planning for non-destructive testing of complex shaped surfaces. In: AIP Conference Proceedings, vol. 1650(1), pp. 1977–1987 (2015)

46. Oropallo, W., Piegl, L.A.: Ten challenges in 3d printing. Eng. Comput. **32**(1), 135–148 (2016)

47. Pandey, P.M., Reddy, N.V., Dhande, S.G.: Improvement of surface finish by staircase machining in fused deposition modeling. J. Mater. Process. Technol. **132**(1–3), 323–331 (2003)

48. Padhye, N., Deb, K.: Multi-objective optimisation and multi-criteria decision making in SLS using evolutionary approaches. Rapid Prototyp. J. **17**(6), 458–478 (2011)

49. Pande, S.S., Kumar, S.: A generative process planning system for parts produced by rapid prototyping. Int. J. Prod. Res. **46**(22), 6431–6460 (2008)

50. Pandey, P.M., Reddy, N.V., Dhande, S.G.: Real time adaptive slicing for fused deposition modelling. Int. J. Mach. Tools Manuf. **43**(1), 61–71 (2003)

51. Pandey, P.M., Thrimurthulu, K., Reddy, N.V.: Optimal part deposition orientation in FDM by using a multicriteria genetic

algorithm. Int. J. Prod. Res. **42**(19), 4069–4089 (2004)

52. Patel, Y., Kshattriya, A., Singamneni, S.B., Roy Choudhury, A.: Application of curved layer manufacturing for preservation of randomly located minute critical surface features in rapid prototyping. Rapid Prototyp. J. **21**(6), 725–734 (2015)

53. Pereira, S., Vaz, A.I.F., Vicente, L.N.: On the optimal object orientation in additive manufacturing. Int. J. Adv. Manuf. Technol. **98**, 1685–1694 (2018)

54. Phatak, A.M., Pande, S.S.: Optimum part orientation in rapid prototyping using genetic algorithm. J. Manuf. Syst. **31**(4), 395–402 (2012)

55. Ramos, B., Pinho, D., Vaz, A.I.F.: Optimal inspection trajectories for 3d printed objects (in preparation, 2019)

56. Rianmora, S., Koomsap, P.: Recommended slicing positions for adaptive direct slicing by image processing technique. Int. J. Adv. Manuf. Technol. **46**(9), 1021–1033 (2010)

57. Richard, D.T., Crawford, R.H.: Optimizing part quality with orientation, pp. 362–368. The University of Texas, Austin (1995)

58. Rocha, A.M.A.C., Pereira, A.I., Vaz, A.I.F.: Build orientation optimization problem in additive manufacturing. In: Gervasi, O., Murgante, B., Misra, S., Stankova, E., Torre, C.M., Rocha, A.M.A.C., Taniar, D., Apduhan, B.O., Tarantino, E., Ryu, Y. (eds.) Computational Science and Its Applications—ICCSA 2018, pp. 669–682, Cham. Springer, Berlin (2018)

59. Rocha, A.M.A.C., Pereira, A.I., Vaz, A.I.F.: Build orientation optimization problem in additive manufacturing. In: International Conference on Computational Science and Its Applications, pp. 669–682. Springer, Berlin (2018)

60. Rocha, A.M.A.C., Silva, A., Rocha, J.G.: A new competitive implementation of the electromagnetism-like algorithm for global optimization. In: International Conference on Computational Science and Its Applications, pp. 506–521. Springer, Berlin (2015)

61. Singhal, S.K., Jain, P.K., Pandey, P.M.: Adaptive slicing for SLS prototyping. Comput.-Aided Des. Appl. **5**(1–4), 412–423 (2008)

62. Steuer, R.E., Choo, E.-U.: An interactive weighted Tchebycheff procedure for multiple objective programming. Math. Program. **26**(3), 326–344 (1983)

63. Strano, G., Hao, L., Everson, R.M., Evans, K.E.: A new approach to the design and optimisation of support structures in additive manufacturing. Int. J. Adv. Manufact. Technol. **66**(9), 1247–1254 (2013)

64. Thrimurthulu, K., Pandey, P.M., Reddy, N.V.: Optimum part deposition orientation in fused deposition modeling. Int. J. Mach. Tools Manuf **44**(6), 585–594 (2004)

65. Vaz, A.I.F., Vicente, L.N.: A particle swarm pattern search method for bound constrained global optimization. J. Glob. Optim. **39**, 197–219 (2007)

66. Vaz, A.I.F., Vicente, L.N.: Pswarm: a hybrid solver for linearly constrained global derivative-free optimization. Optim. Methods Softw. **24**, 669–685 (2009)

67. Velednitsky, M.: Short combinatorial proof that the DFJ polytope is contained in the MTZ polytope for the Asymmetric Traveling Salesman Problem. Oper. Res. Lett. **45**(4), 323–324 (2017)

68. Verma, A., Rai, R.: Sustainability-induced dual-level optimization of additive manufacturing process. Int. J. Adv.

Manuf. Technol.ogy **88**(5), 1945–1959 (2017)

69. Wang, W., Chao, H., Tong, J., Yang, Z., Tong, X., Li, H., Liu, X., Liu, L.: Saliency-preserving slicing optimization for effective 3d printing. Comput. Graph. Forum **34**(6), 148–160 (2015)

70. Wang, W.M., Zanni, C., Kobbelt, L.: Improved surface quality in 3d printing by optimizing the printing direction. In: Proceedings of the 37th Annual Conference of the European Association for Computer Graphics, EG'16, pp. 59–70, Goslar Germany, Germany. Eurographics Association (2016)

71. Yu, E.A., Yeom, J., Tutum, C.C., Vouga, E., Miikkulainen, R.: Evolutionary decomposition for 3d printing. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'17, pp. 1272–1279, NY, USA, 2017. ACM

72. Zhang, Y., Bernard, A., Harik, R., Karunakaran, K.P.: Build orientation optimization for multi-part production in additive manufacturing. J. Intell. Manuf. **28**(6), 1393–1407 (Aug 2017)

73. Faria, C., Fonseca, J., Bicho, E.: FIBR3DEmul—an open-access simulation solution for 3D printing processes of FDM machines with 3+ actuated axes. Int. J. Adv. Manuf. Technol. 106, 3609–3623(2020)

[1] http://www.coppeliarobotics.com/index.html .

[2] Physical bodies are not displayed during the simulation.

[3] Example of C3DPrinter movement at different feed rates https://www.youtube.com/watch?v=G_3gCUfiRAA .