# Behind Algorithmic Geometric Patterns

## A Framework for Facade Design Exploration

**Inês Caetano**

INESC-ID/Instituto Superior Técnico, University of Lisbon, Portugal

**António Leitão**

INESC-ID/Instituto Superior Técnico, University of Lisbon, Portugal

**Abstract** Architecture has always explored the latest technological advances both in terms of building design and fabrication. Among the recently adopted computational design approaches, Algorithmic Design (AD) shows great potential for the conception, analysis, and production of architecture, due to automating repetitive and time-consuming design tasks, facilitating design changes, increasing design freedom, and facilitating the search for better-performing solutions. These advantages are particularly important for the design of building facades, providing the flexibility needed to deal with the design complexity of this architectural element. However, AD is an abstract formal method that requires programming skills, which explains its still shy adoption in the field. Despite the AD tools released to smooth its learning curve, few successfully combine creative tasks with the need to respond to multiple requirements and almost none simplify the algorithmic task, forcing architects to build the necessary functionalities from scratch. This research addresses these problems by structuring an architectural-oriented theory considering the variability and context-specificity of architectural design practice and responding to its different aesthetic, performance, and construction requirements. To make it useful for architects, the theory is implemented in an AD framework, whose application promises to decrease the time and effort needed to geometrically explore, analyze, and materialize new facade designs, while smoothing the transition between design stages and their different tools. In this paper, we focus on the mathematical implementation of three-dimensional unconventional facade elements, assessing the ability of the resulting formalisms to generate, transform, and materialize the produced solutions.

**Keywords.** Algorithmic Design; Facade Design; Geometric Patterns; Mathematical Framework.

## 1. Introduction

Architecture has always explored the latest technological advances both in terms of building design and fabrication. Nowadays, digital tools and computation-based design approaches play a relevant role in the conception, analysis, and production of architecture. These include Algorithmic Design (AD), a formal method that creates designs through algorithms [1] and whose flexibility brings several advantages to the design practice, such as automating repetitive, time-consuming design tasks; facilitating design changes; increasing design freedom; and facilitating the search for better-performing solutions.

Among AD's potential applications, the design of building facades stands out due to (1) the aesthetic and environmental relevance of this architectural element [2,3]; (2) the complexity of its design [4], which involves dealing with multiple context-specific design constraints [5]; (3) the growing need to reduce the buildings' environmental impact [6,7]; and (4) its tectonic potential in providing comfortable and better-performing interior spaces. Nevertheless, and despite its advantages, AD is not yet widespread in the field, mainly due to its high level of abstraction and its need for programming skills [8]. To smooth its learning curve, several AD tools have been released in the last decades, but few successfully combine the architects' creative practice with the need to simultaneously respond to multiple requirements. Moreover, most do not sufficiently simplify the algorithmic task, forcing architects to build the necessary functionalities from scratch.

To make AD more accessible to architects it is therefore important to systematize the algorithmic generation of design solutions in an architectural-oriented theory considering the wide variety of possible design scenarios, while responding to different aesthetic, performance, and construction requirements. We address this problem by extending our previous research on mathematics-based strategies encompassing the variability, diversity of requirements, and context-specificity of facade design problems [9–11]. More precisely, we add a set of strategies addressing a wider range of geometry-related principles, as well as their fabrication through different manufacturing means and materials. The aim is to support architects with some programming experience in the algorithmic development and realization of facade design solutions, decreasing the implementation time of new designs, while providing the flexibility needed to handle each design stage in a continuous workflow.

As in previous research [9–11], the mathematics-based strategies addressing the geometric exploration and fabrication of a wider range of facade elements are implemented in a framework of predefined algorithms that can be easily combined in the development of new facade design solutions. In this paper, we elaborate not only on their implementation in the AD framework but also on their application to generate, transform, and materialize nonstandard three-dimensional facade elements. To evaluate the flexibility and versatility of the extended AD framework, we apply it in the step-by-step development of a set of case studies of different volumetric compositions, while addressing their subsequent materialization using different manufacturing strategies.

## 2. Methods

Inspired by modular programming and design patterns strategies [12–16], this investigation aims at systematizing facade design processes based on AD by:

1. reducing the time and effort spent on AD tasks, freeing the architect from having to write all algorithms from scratch and avoiding potential programming errors.
2. guiding the selection and combination of the best algorithms for a given scenario, making it easier to deal with the diversity and context-specificity of most design requirements.
3. smoothing the transition between design stages, solving the interoperability issues between their different specialized tools and thus minimizing the accumulation of errors.
4. facilitating the conversion of the resulting AD models into physical ones by automatically adapting their structure according to the fabrication means.

To that end, we adopt a mathematics-based perspective to structure a theory handling the complexity, variability, and diversity of facade design problems, implementing it in an AD framework containing different categories of algorithms. The preference for a text-based algorithmic implementation over a visual-based one lies on its greater expressiveness and scalability, which is critical to deal with the complexity and scale of architectural design problems. In this research, we extend previous theories [9–11] by placing particular emphasis on the geometric exploration and concretization of three-dimensional unconventional shapes, resulting in a six-stage methodology encompassing the following tasks:

1. identification of the most relevant facade design problems.
2. solution of the collected problems using the mathematical formalism.
3. integration of the latter into the overall theory.
4. elaboration of algorithmic strategies addressing their materialization.
5. implementation of both (3) and (4) into the AD framework.
6. practical application.

The results of tasks 1 to 3 are briefly described in section 3, and those of tasks 4 to 6 are presented in section 4. Section 5 discusses the previous findings, concluding the proposal has enough flexibility to adapt to the ever-changing nature of architectural practice and its diversity of design scenarios and requirements.

## 3. Mastering unpredictability

Architectural design problems are unique by nature as they are the natural product of multiple design requirements and constraints that can be global or context-specific, straightforward or abstract, and fixed or evolving. When combined with the variability of the design brief and the architects' creative nature, their complexity becomes further accentuated, making it difficult to use the same strategy in different scenarios. Moreover, given the projects' tight deadlines and the lack of flexibility of most design tools, the need to quickly explore a wide range of possible solutions is often unfeasible, hindering the creative potential of the architect. As a result, only a limited set of solutions is often considered, leaving many design scenarios unexplored.

This becomes especially evident in the design of building facades, due to the need to consider their multiple functions and requirements [2,3]. Given the potential of AD approaches to improve design processes, we propose their use in current practices. To address their technical complexity and abstractness, we propose a mathematics-based theory and framework to support the algorithmic development of building facades, containing several AD strategies organized into a multidimensional classification.

We expect to overcome some of the limitations found in facade design processes by guiding architects towards the most suitable algorithms for (1) generating the idealized solution; (2) analyzing and optimizing it regarding one or more performance criteria; (3) making it feasible in terms of cost and resources; and (4) facilitating its manufacturing. Nevertheless, given the diversity and context-specificity of most design problems, we do not expect this matching process to yield a complete algorithmic solution and, thus, we assume that architects are still responsible for (1) dividing the design into smaller parts, (2) identifying and establishing dependencies between them, (3) combining the different algorithms dealing with each part, (4) implementing additional algorithms that might be needed to handle the specific circumstances of the design brief, and (5) executing the algorithms and evaluating the results.

In the next section we elaborate on the proposed formal methods together with their implementation and application in a set of case studies.

## 4. Mathematics-based implementation

As a starting point, consider the framework presented in [9], whose mathematical principles are organized by type and role in facade design processes in the following categories: *Geometry*, *Distribution*, *Pattern*, *Optimization*, and *Rationalization*.

The proposed formalism regards building facades as two-dimensional parametric surfaces described as $S(u, v)$, whose shape can be defined through algorithms from the *Geometry* category, such as $Straight$ and $Cylindrical$, which create planar and cylindrical parametric surfaces, respectively. Along this surface we can distribute one or more geometric elements according to different configurations available in the *Distribution* category, such as the squared and hexagonal grids produced by the algorithms $grid_{squares}$ and $grid_{hexagons}$. Combining these two types of algorithms with those of the *Pattern* category originates several geometric patterns, whose levels of complexity and variability depend on the algorithms selected: these can either target the creation of different shapes (the *Shape* subcategory), e.g., the algorithms $shape_{star}$ and $shape_{pyramid}$, or their geometric manipulation (the *Transformation* subcategory), e.g., the algorithms $T_{scale}$ and $T_{rotate}$. This means that the more algorithms are selected from the first subcategory, the more variety of shapes the design will have, whereas the more are selected from the second subcategory, the more diverse their geometric variation will be. To control the shapes' geometric characteristics to meet one or more performance requirements, we can use the algorithms available in the *Optimization* category, such as the algorithms $opt_{structure}$ and $opt_{daylight}$, and to make the shapes feasible for fabrication, we resort to the *Rationalization* category, which provides algorithms like $tallying$ and $rationalize$ to either count or reduce the number of different elements composing the final solution.

This research extends this classification by (1) adding more algorithms to both *Shape* and *Transformation* subcategories addressing the generation and manipulation of three-dimensional façade elements, accordingly; (2) adapting some of the existing ones in the *Transformation* subcategory to be able to deal with the manipulation of three-dimensional shapes; and (3) including an additional category, *Fabrication*, addressing the materialization of the resulting solutions.

## 4.1. Pattern

This category contains algorithms to create different facade patterns resulting from the repetition of one or more geometric elements that can be kept unchanged or change according to one or more geometric transformations. In previous research [9,17,18], we focused on the mathematical representation of one- and two-dimensional patterns, addressing their potential to generate a wide range of geometric patterns responding to different aesthetic and performance requirements. In the current research, we focus on the formal methods driving the creation, manipulation, and materialization of three-dimensional patterns.

As described in [9], all *Shape* algorithms receive a set of points ($pts$) and a set of additional parameters depending on the geometric characteristics of the shape to create: while a rectangular element receives the length and width of its edges ($e_u, e_v$) and an horizontal placement angle ($\alpha_u$) (see Equation 1), a cuboid element receives the previous parameters plus a height ($e_z$) and a vertical placement angle ($e_v$) (see Equation 2). When combined with both *Geometry* and *Distribution* algorithms, these algorithms will extract different amounts of information depending on the received parameters: while the former requires the surface points and their normal vector to place each rectangular shape, the latter requires these points to center the cuboid element and both their normal and tangent vectors to correctly orientate it (Fig. 1).

$$shape_{rectangle}(pts, e_u, e_v, \alpha_u) \qquad (1)$$

$shape_{cuboid}(pts, e_x, e_y, e_z, \alpha_u, \alpha_v)$ (2)

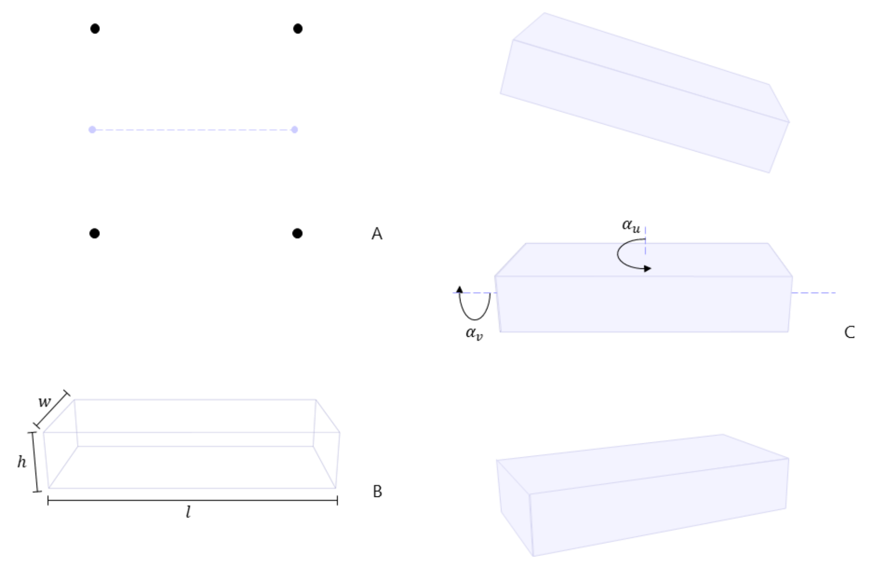


**Fig 1.** Cuboid algorithm parameters: set of points centering its axis (A); its length, width, and height values (B) and its placement angles regarding the points' tangent and normal vectors (C).

Regarding more complex shapes resulting from constructive solid geometry operations, such as union, subtraction, intersection, or more complex modeling operations such as morphing, lofting, bending, among others, the framework provides algorithms to create, for instance, irregular 3D tile shapes and panels [19], complex truss-like structures [20], and nonstandard brick elements [21].

One such example is the algorithm producing a type of *cobogó* bricks (Equation 3), which receives, in addition to the set of points ($pts$) defining its outer frame shape (Fig. 2A) and two placement angles ($\alpha_u$ and $\alpha_v$) dictating its spatial orientation (Fig. 2B-C), a thickness and a width ($thick$ and $width$) controlling the corresponding frame dimensions (Fig. 2D). In this case, the resulting shape is achieved either through the union of several parallelepiped elements or the subtraction of a smaller three-dimensional element from a larger one of the same shape. Then, to create its inner elements, the algorithm receives their radius size ($r_{bars}$) and a rule guiding their spatial orientation ($rule$). The result is a geometric composition made of smaller regular elements whose spatial distribution can follow different rules (Fig. 3):

$$shape_{cobogo}(pts, \alpha_u, \alpha_v, thick, width, r_{bars}, rule) \quad (3)$$

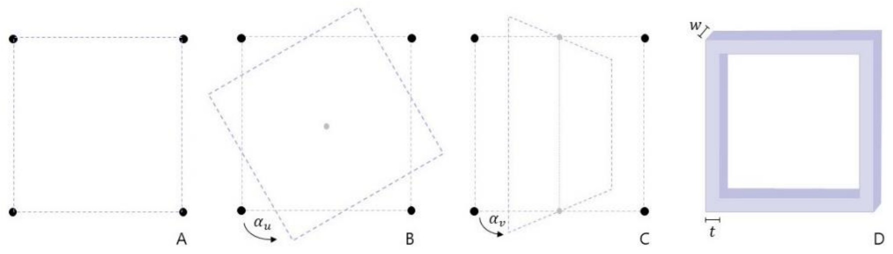


**Fig 2.** *Cobogó* algorithm parameters: set of points shaping the brick's frame (A); placement angles regarding the $u$ (B) and $v$ directions (C); and frame's width and thickness values (D).

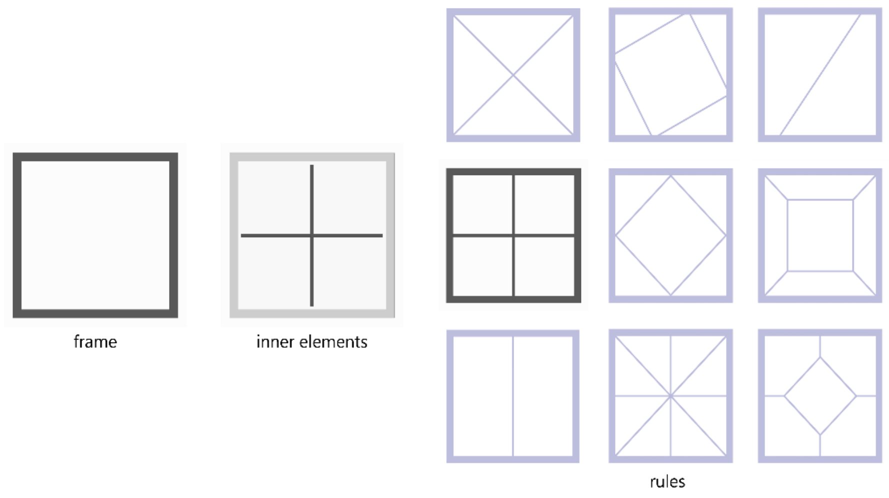


**Fig 3.** *Cobogó* algorithm geometric evolution: creation of the outer frame and its inner elements according to different rules.

Given the framework's flexible nature, we can now manipulate the latter's parameters in different and independent ways and apply, for instance, multiple geometric rules that result in different *cobogó* elements. To that end, we provide it with:

1. the set of positions where to create each *cobogó* shape ($ptss$), which we obtain by combining the algorithms $straight$ and $grid_{squares}$ from the *Geometry* and *Distribution* categories.
2. the dimensions characterizing those shapes ($thick, width, r_{bars}$), to which we assigned a set of fixed values.
3. the geometric rules to apply ($rules$), which correspond to a set of functions.

To deal with the different number of dimensions of the received arguments, we benefit from *broadcasting* [9], i.e., the mapping of a function across multidimensional data structures, allowing us to apply the *Shape* algorithm $shape_{cobogo}$ to an array of elements with a higher number of dimensions than expected, as it happens with its parameter $pts$, which is expecting a one-dimensional array but receives a two-dimensional one ($ptss$) resulting from the combination $grid_{squares}(straight(w,h))$; or with a higher number of dimensions than the other received arguments, e.g., while $ptss$ is a two-dimensional array, $rules$ is a one-dimensional one, and the remaining arguments are independent numeric values. Using broadcasting, each one-dimensional array ($pts$) of the two-dimensional one ($ptss$) is independently assigned to the algorithm $shape_{cobogo}$, as also is each independent function of the one-dimensional array $rules$. This allows us, for instance, to make the latter selection random by simply combining it with the *Transformation* algorithm $T_{random}$, which returns randomly chosen values (Equation 4). Fig. 4 illustrates the results with two examples resulting from randomly alternating between two and five possible rules.

$$shape_{cobogo}.\begin{pmatrix} grid_{squares}(straight(w,h)), \\ \alpha_u, \alpha_v, thick, width, r_{bars}, \\ T_{random}(rules) \end{pmatrix} \tag{4}$$

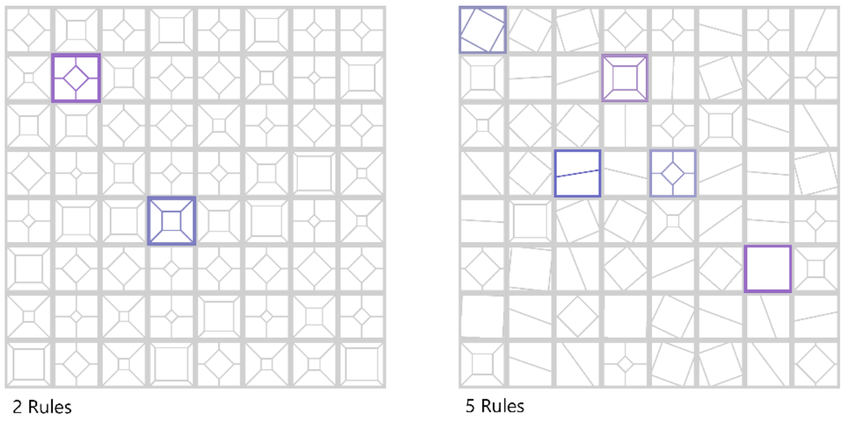


**Fig 4.** Two patterns resulting from the same algorithmic composition but different sets of rules: on the left, the random selection is between 2 rules and, on the right, it is between 5 rules.

As another example, we can make the equally sized *cobogó* elements irregular by simply transforming the set of surface positions shaping them, making their uniform distribution randomly vary in both $u$ and $v$ directions. To that end, we select the *Transformation* algorithm $T_{translate}$, which translates a set of surface points ($ptss$) according to a given translation factor ($k$), combining it with the $T_{random}$ in the following composition:

$$T_{translate}.\left(grid_{squares}(straight(w,h)), T_{random}\right) \tag{5}$$

By adding the *cobogó* algorithm to the previous composition (Equation 5), we obtain bricks of varying shapes and sizes as those illustrated in Fig. 5, which result from randomly selecting between four possible rules ($rules$) with either fixed (left) or random (right) opacity levels (see Equation 6).

$$shape_{cobogo}\cdot\begin{pmatrix} T_{translate}\cdot\left(grid_{squares}\left(straight(w,h)\right),T_{random}\right), \\ \alpha_u,\alpha_v,thick,width,r_{bars}, \\ T_{random}(rules) \end{pmatrix} \tag{6}$$

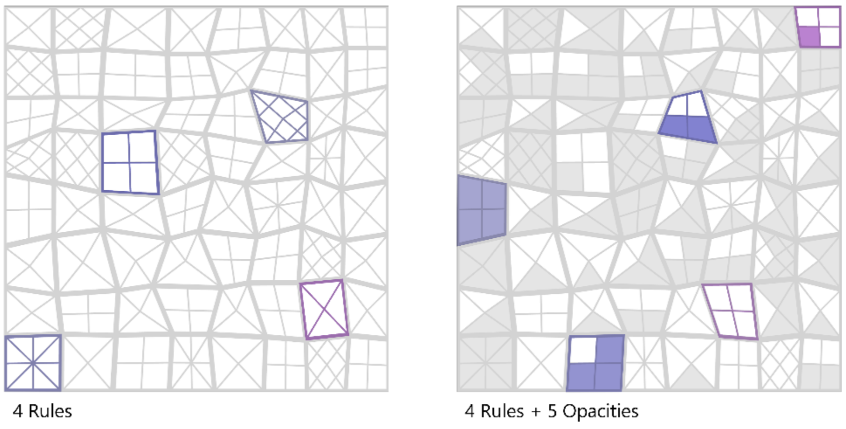


**Fig 5.** Two patterns resulting from the same algorithm but different opacity levels: the algorithm randomly selects between four rules with no opacity level, on the left, and five possible opacity levels, on the right.

As another example, to create a pictorial visual effect resulting from horizontally rotating standard bricks in different ways, we combine the previous algorithm $shape_{cuboid}$ with the *Transformation* one $T_{rotate}$, making the latter control the former's horizontal placement angle in the following composition:

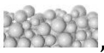
$$shape_{brick}(pts,lenght,width,height,\alpha_u\times T_{rotate},\alpha_v) \tag{7}$$

To control the rotation angle so as to create the desired visual effect, we select the *Transformation* algorithm $T_{pictorial}$, which receives (1) the surface points where to create the pattern ($ptss$); (2) a matrix containing the transformation algorithm(s) to apply ($M_{transfs}$,); and (3) another matrix with the intended pictorial effect ($M_{pattern}$). Based on the latter information ($M_{pattern}$), it then maps the algorithms of $M_{transfs}$ along the positions ($ptss$), affecting the shapes assigned to them. As, in this case,

$M_{transfs} = [T_{rotate}]$, only this algorithm is applied to the bricks, while using the rotation angles set in $M_{pattern}$.

To translate the desired pictorial effect into rotation angles, we select the algorithm $pixelmap_{image}$, which, based on an image ($image$) and its domain of application ($ptss$), collects the RGB values of the former's pixels, while storing them in a matrix with the size of the given domain. When provided to the algorithm $T_{pictorial}$, the collected RGB values are automatically converted into factors that consider, for instance, the type of transformation(s) to perform and the intensity of the pictorial effect to create. Fig. 6A illustrates the result of the following composition:

$$T_{pictorial}\left(\begin{array}{c} itera_{rhombus}\big(straight(w,h)\big), \\ [T_{rotate}], \\ pixelmap_{image}\left(\quad , straight(w,h)\right) \end{array}\right) \tag{8}$$



Having this composition, we can simply replace $T_{rotate}$ with other transformation algorithms, such as $T_{colours}$ or $T_{translate}$, and obtain the same visual effect through differently colored or protruded bricks, accordingly (Fig. 6B-C).

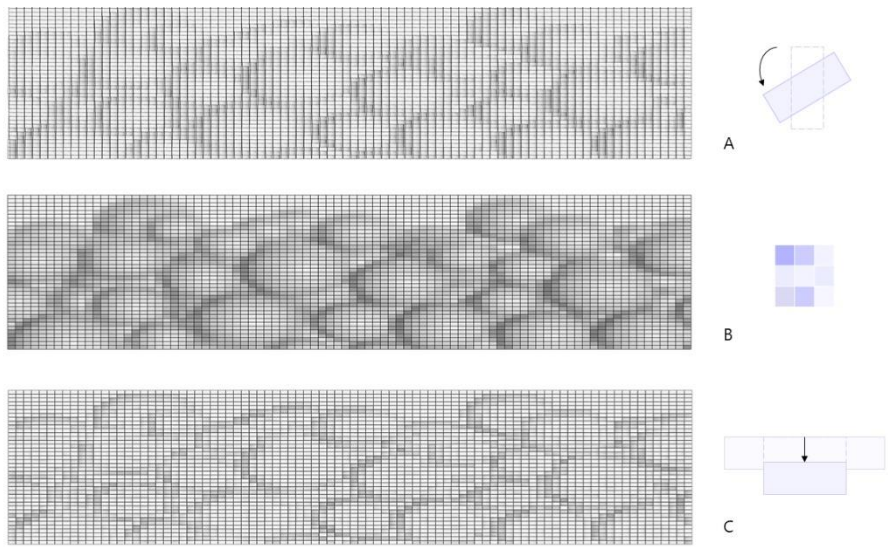


**Fig 6.** Three examples resulting from the *pictorial* algorithm: creating a spheres-inspired pattern by strategically rotating (A), coloring (B), and protruding (C) the bricks.

## 4.2. Fabrication

This category contains algorithms to materialize the developed solutions through different manufacturing techniques and materials. When combined with the previous algorithms, they (1) suggest different fabrication strategies by considering the designs' geometric and material characteristics, while (2) automating the production of the technical documentation needed for each one (see Fig. 7), e.g., two-dimensional drawings with different line types delimiting the areas to cut, engrave or fold by laser-cutting, or 3D models containing the printer-head paths for 3D printing. They also provide (3) a high-level control over the solutions' manufacturing, contributing to the latter's increased efficiency, accuracy, and viability: e.g., adjusting the printing path planning, i.e., the trajectory of the 3D printer head, to obtain either the desired structural integrity or surface quality, or manipulating the profiles' thickness of sectioning strategies to meet the desired visual effect, among others.

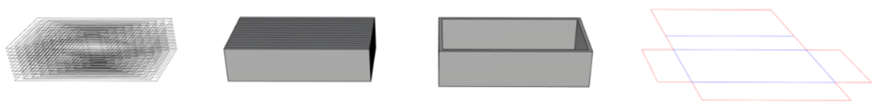


**Fig 7.** Automatic production of technical documentation for the same shape according to the selected manufacturing technique: 3D printing, sectioning, casting, and laser cutting.

In the former case, by receiving the set of shapes to produce, the available functionalities analyze their geometric characteristics to understand the suitability of different manufacturing scenarios to fabricate them, which in turn require different representation schemes and methods. As an example, while the *sectioning* strategy is based on the use of a series of profiles to create either a surface or a structure, *cutting* involves the extraction of two-dimensional planar elements from surfaces or solids, and *forming* uses molds to mass-produce elements. Therefore, when the elements to produce are, for instance, differently patterned panels, the first two strategies will probably be suggested, whereas when they are customized, three-dimensional tiles with, for instance, a round shape, it is the last one that will eventually be proposed.

In the latter cases, they allow controlling the manufacturing process by adjusting its related parameters, while evaluating the impact each strategy has on the solution's mathematical description, changing its structure accordingly. The result is a new algorithm representing the solution according to the specifications of the selected manufacturing strategy, from where all the information and technical documentation needed for the actual fabrication can be automatically extracted.

As an example, consider a pattern similar to that of Fig. 5 (left) and suppose that we intend to manufacture it using 3D printing. We therefore combine the *3D printing* algorithm ($fab_{3Dprint}$) with those producing the *cobogó* elements, automatically converting the latter's three-dimensional shapes into paths for the printer head. When planning these paths, the algorithm considers different printing specificities, such as the printer resolution ($head_{mm}$), the material layer thickness ($layer_{mm}$), and the printing strategy ($path_{plan}$), which in turn result in different numbers of printing paths and distances between them, and different trajectories. Accordingly, the resulting machining time, surface finishing, geometric accuracy, and material use also change [22]. Fig. 8 illustrates the result of this combination with the conversion of a set of *cobogó* bricks into their corresponding printing paths, where $head_{mm} = 3$, $layer_{mm} = 1.5$ , and the planning type is based on alternated paths by level.

To convert the *cobogó* elements' volumetric model into one prescribing the printer head paths, its algorithmic representation has also to change, the originally used geometric solid primitives being replaced by path operations (Equation 9), whose distribution along the generated volume is controlled by the previous parameters.

$$fab_{3Dprint}\left(shape_{cobogo}\right) \rightarrow shape_{3Dpaths} \tag{9}$$

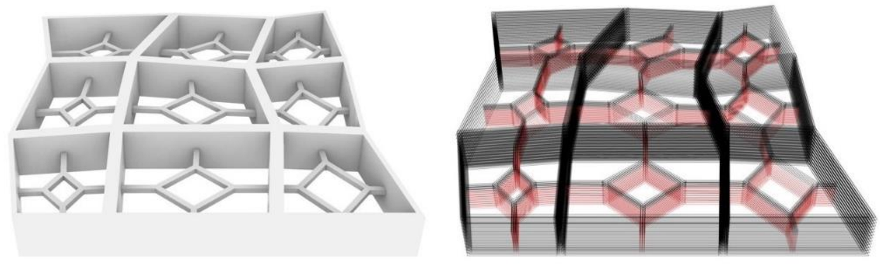


**Fig 8.** A perspective view of the design's 3D model on the left, and the resulting printing paths (black lines) and support structures (red lines) on the right.

On the other hand, if we want to produce the previous elements through casting strategies, we select the *forming* algorithm ($fab_{forming}$), automatically obtaining the 3D models of their negative shapes (Equation 10), together with cost-related information, such as number of different molds and material quantities (see Fig. 9). As, in this case, both the original and translated models correspond to three-dimensional, albeit inverse, shapes, the converted algorithm will apply the same primitives as the original one with only slight differences.

$$fab_{forming}\left(shape_{cobogo}\right) \rightarrow shape_{negative} \tag{10}$$

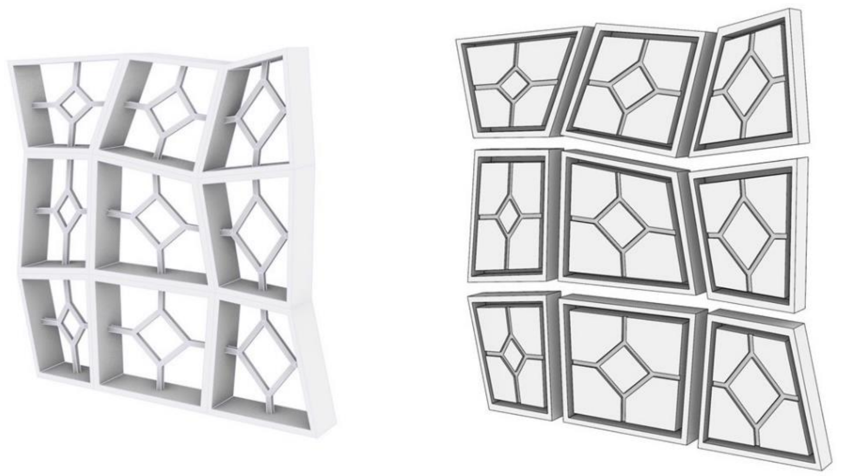


**Fig 9.** A perspective view of the design's 3D model (left) and the resulting molds with each element negative shape (right).

Having the molds' 3D models, the possibilities for their production are the same as for the *cobogó* bricks, making it possible to proceed with the combination of the resulting algorithm with those available in the *Fabrication* category and, for instance, planning the paths for their 3D printing, if we select $fab_{3Dprint}$ (Equation 11), or setting the instructions for their CNC milling, if we opt for the $fab_{milling}$ (Equation 12).

$$fab_{3Dprint}(shape_{negative}) \rightarrow shape_{3Dpaths} \tag{11}$$

$$fab_{milling}(shape_{negative}) \rightarrow shape_{millingOperations} \tag{12}$$

As a last example, if we adopt a sectioning strategy, we combine the algorithm $fab_{section}$ with those producing the *cobogó* elements, automatically obtaining a set of profiles, whose number is controlled by the $n_{profiles}$ parameter and whose superposition creates the intended shapes (Fig. 10 middle). When setting the previous parameter ($n_{profiles}$), it must be considered that, first, it depends on the thickness of the selected material and, second, its division by the element's total thickness does not always result in a whole number of sections. As, in this case, the latter thicknesses are 10 mm and 90 mm, correspondingly, we set $n_{profiles} = 9$. Given the profiles' relatively small thickness (i.e., 10 mm), the use of laser cutting techniques is therefore suggested, requiring the production of two-dimensional drawings with paths representing the areas to cut. Finally, to obtain this technical documentation, we add the algorithm $fab_{lasercut}$ to the previous combination, the results being illustrated in Fig. 10 (right).

Regarding the resulting mathematical structures, while the combination of $shape_{cobogo}$ with $fab_{section}$ originates a new algorithm using the same geometric solid primitives but in different ways (Equation 13), its combination with $fab_{lasercut}$ causes more radical changes as the latter are replaced with line primitives defining their contour shape (Equation 14).

$$fab_{section}\left(shape_{cobogo}\right) \rightarrow \left\{shape_{layer01}, \cdots, shape_{layern}\right\} \tag{13}$$

$$fab_{lasercut}.\left(shape_{cobogo}\right) \rightarrow shape_{contour} \tag{14}$$

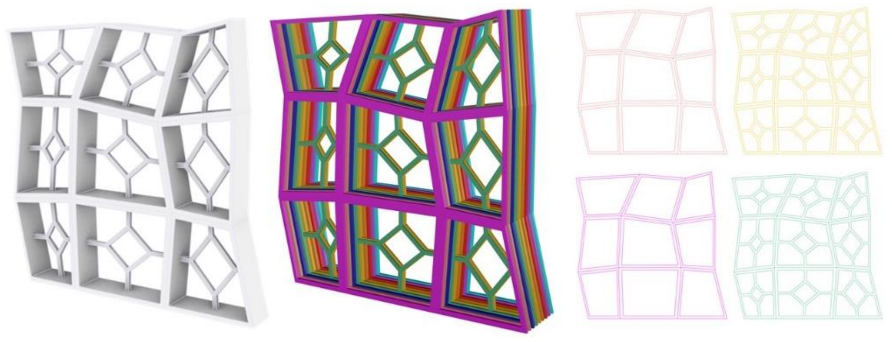


**Fig 10.** A perspective view of both the design's 3D model (left) and sectioned model (middle) together with some of the profiles' two-dimensional drawings for laser cutting (right).

Having the ability to test and visualize multiple manufacturing possibilities allows us to assess their different advantages and disadvantages, not only in terms of their production times, material waste, and overall costs, but also regarding the aesthetic quality, geometric precision, and physical properties of the resulting solutions.

## 5. Evaluation

The mathematical nature of our proposal makes its principles easy to implement in any AD tool for architectural design, either using the visual or the textual programming paradigm. In this research, we focused on the latter due to having the scalability and expressiveness needed to solve large-scale, complex design problems. To evaluate the proposal, we implemented it in the AD tool *Khepri* [23], the result being a text-based AD framework supporting the algorithmic development and manufacturing of facade design solutions. In the next sections, we compare the results of its application with other state-of-the-art AD tools in terms of (1) geometric exploration and (2) fabrication.

## *5.1. Algorithmic-based exploration*

Within the scope of building facades, there are already some AD tools supporting the algorithmic development of facade design solutions, which include (1) Para-Cloud Gem, which provides features to map 3D elements on a mesh, subdivide and edit surfaces, integrate fitness requirements, and 3D print the resulting solutions; (2) Dynamo's addon's Quads from Rectangular Grid, Ampersand, Clockwork, LunchBox, MapToSurface, Pattern Toolkit, and LynnPkg, which contain features for surface paneling, mapping elements on a surface, and creating and manipulating geometric patterns; and (3) Grasshopper's plugins PanelingTools, LunchBox, Weaverbird, Parakeet, and SkinDesigner, which integrate surface paneling and pattern generation functionalities, rationalization and mesh subdivision techniques for analysis and fabrication, mechanisms to produce facade geometries from buildings massing surfaces repeating panels, among others.

Despite facilitating the typical modeling procedures of facade design processes, these tools present several limitations, such as (1) requiring frequent manual-based interactions and thus favoring iterative user-driven processes that are potentially tiresome and error-prone; (2) suffering from the scalability and performance limitations of visual programming languages, particularly, when dealing with larger AD solutions [24–30]; (3) having limited ability to directly address relevant concepts like materiality and tectonic relation between facade elements, being often restricted to generic panelization, subdivision, and population of surfaces problems; and (4) providing a limited set of predefined operators that are difficult to adapt to respond to more specific problems [28].

Our proposal addresses most of these limitations, providing users with higher levels of design freedom by (1) automating repetitive and error prone design tasks, minimizing manual intervention; (2) facilitating programming tasks, reducing the time and effort spent in them; (3) smoothing the transition between designs stages, facilitating the coordination between their specific requirements; and (4) addressing the solutions' materiality and concretization, making it easier to consider different manufacturing scenarios.

By reducing the time and effort spent in each of the previous tasks, the user is left with more time available for creative exploration, potentially increasing the explored design space and the probability of finding better solutions, whether in terms of aesthetic, performance, or feasibility. Moreover, as our proposal supports flexible design workflows merging the different design stages' information, it also eliminates most interoperability issues resulting from their transition process, while solving the latter's tendency to accumulate errors.

These advantages are visible in the examples of the previous sections, where we used the framework to facilitate, first, the exploration of multiple geometric compositions responding to different aesthetic requirements and then, the transition between design exploration and materialization stages, not only increasing the variety of construction scenarios considered, but also improving our perception of their impact on the solutions' aesthetic quality.

## *5.2. Algorithmic-based manufacturing*

Regarding AD tools for manufacturing, relevant examples include the plugins (1) FabTools, Bowerbird, Xylinus, Droid, Kuka Prc, RoboDK, OpenNest, and Ivy for Grasshopper; (2) DynaFabrication, Fabrication API, 3BMLabs.DigiFab, and ParametricMonkey for Dynamo; and (3) Laser Slicer Addon for Blender. However, these are mostly (1) based on visual programming languages, whose limitations hinder the manufacturing of more complex solutions [24–30]; (2) tool specific, forcing architects to use multiple AD tools to assess different construction schemes; (3) limited in terms of modeling freedom; and (4) dependent on laborious, time-consuming, and error prone manual- or script-based interventions, not fully automating the design-to-fabrication conversion and the extraction of technical documentation. Given the uniqueness of architectural design problems, these interventions are, however, hardly reused in different projects without major modifications, thus hindering the testing of different manufacturing scenarios and construction schemes.

In the case of our framework, besides smoothing the design-to-manufacturing transition process by automating most of its related tasks, it provides control over the manufacturing process and its different parameters, allowing higher levels of production quality, accuracy, and viability [22,31]. Moreover, by adapting the designs' algorithmic descriptions according to the specificities of the selected fabrication technology, the framework ensures that the available functionalities are portable between design and manufacturing tools, overcoming the latter's typical interoperability issues and allowing the use of the same algorithm to obtain both the design's geometric and construction models containing the required information. This is illustrated in Figs. 8-10, where we experimented different construction schemes and manufacturing strategies for the same solution, making it easier to assess their different advantages and disadvantages.

There are, nevertheless, several aspects that need to be further developed and integrated in the proposed framework, particularly in what regards the specificity of each manufacturing technique and its sensibility to different requirements and parameters. As an example, while in 3D printing and CNC cutting the printing path planning, i.e., the specification of the tool movement, is critical for achieving higher levels of surface quality [32,33] and shape accuracy [34,35] and reducing time and material expenditure [36–38], in cutting strategies it is important to optimize the machines' cutting paths by considering parameters such as cutting speed, material thickness, laser cutting type, and material used, among others [31,39–43]. In contrast, casting strategies need to address the molds' production cost and material and geometric properties, as well as their impact on the geometric complexity, surface quality, and molding speed of the produced elements [44-56].

We are currently extending our proposal with guiding strategies supporting the iterative refinement of the solutions' manufacturing process by also considering the elements' geometric properties, materiality, and desired finished quality and precision. We also intend to integrate functionalities for cost and machining time control, as well as for comparing the trade-offs resulting from the previous requirements in different manufacturing scenarios.

## 6. Conclusion

The field of architecture is always evolving to accommodate the latest technological advances in terms of design exploration and fabrication. One such advance is Algorithmic Design (AD), a design approach based on algorithms that has the potential to flexibly coordinate conceptual, performance, and construction requirements. Besides automating repetitive and time-consuming design tasks, AD facilitates design changes and increases design freedom. By reducing the time and effort needed to explore new designs, it allows architects to explore wider design spaces, promoting the search for improved solutions. AD is particularly apt to solve intricate design problems involving multiple aesthetic, performance, and construction requirements.

Nevertheless, AD is a complex and abstract approach that requires programming skills, which most architects do not have. Despite the release of several AD tools in the last decades aiming at smoothing its learning curve, few successfully combine the architect's creative process with the need to meet multiple design requirements. This paper addressed this problem by systematizing the formal methods behind the algorithmic generation of facade design solutions in a mathematics-based theory considering the wide variety of existing design scenarios and strategies, as well as the variability and context-specificity of architectural problems. The aim is to make AD more accessible to architects by not only reducing the time and effort spent on algorithmic-related tasks, but also guiding the search for improved solutions in terms of aesthetics, performance, and constructability.

To evaluate the proposal, we implemented it in an AD framework specialized in facade design processes, placing particular emphasis on the geometric exploration and materialization of three-dimensional unconventional elements. Based on the results, the proposed formal approach has enough flexibility to coordinate the geometric exploration of facade design solutions of different volumetric compositions and their concretization using different manufacturing means and strategies.

As a still ongoing investigation, current efforts have been placed on the improvement of the available functionalities and their extension with more advanced features, such as providing higher levels of control over the different manufacturing strategies; extending the range of shapes supported; adding cost and machining time control strategies; and including a recommender system comparing the trade-offs resulting from different requirements and manufacturing scenarios.

## Acknowledgements

This work was supported by national funds through *Fundação para a Ciência e a Tecnologia* (FCT) (references UIDB/50021/2020, PTDC/ART-DAQ/31061/2017) and the PhD grant under contract of FCT (SFRH/BD/128628/2017).

## References

[1] I. Caetano et al., 'Computational design in architecture: Defining parametric, generative, and algorithmic design', Front. Archit. Res., vol. 9, no. 2, pp. 287–300, 2020.

[2] C. Schittich, Building Skins. Birkhäuser, 2006.

[3] Y. S. ElGhazi, 'Building Skins in the Age of Information Technology', Faculty of Engineering at Cairo University, 2009.

[4] S. Dritsas, 'Design-Built: Rationalization Strategies and Applications', Int. J. Archit. Comput., vol. 10, no. 04, pp. 575–594, 2012.

[5] C. K. Boswell, Exterior Building Enclosures: Design process and composition for innovative facades. John Wiley & Sons, Inc., 2013.

[6] L. De Boeck et al., 'Improving the Energy Performance of Residential Buildings: A literature review', Renew. Sustain. Energy Rev., vol. 52, pp. 960–975, 2015.

[7] Y. Huang and J. Niu, 'Optimal Building Envelope Design Based on Simulated Performance: History, current status and new potentials', Energy Build., vol. 117, no. September, pp. 387–398, 2015.

[8] R. Castelo-Branco et al., 'Digital representation methods: The case of algorithmic design', Front. Archit. Res., 2022.

[9] I. Caetano and A. Leitão, 'Mathematically Developing Building Facades: An algorithmic framework', in Formal Methods in Architecture: Advances in Science, Technology & Innovation (IEREK Interdisciplinary Series for Sustainable Development), S. Eloy, D. Leite Viana, F. Morais, and J. Vieira Vaz, Eds. Springer, Cham, 2021, pp. 3–17.

[10] I. Caetano et al., 'From Architectural Requirements to Physical Creations', Journal of Façade Design and Engineering. TU Delft / Faculty of Architecture and The Built Environment, vol. 8, pp. 59–80, 2020.

[11] I. Caetano et al., 'From Idea to Shape, From Algorithm to Design: A Framework for the Generation of Contemporary Facades', in The next city - New technologies and the future of the built environment [16th International Conference CAAD Futures], 2015, p. 483.

[12] R. Woodbury et al., 'Some Patterns for Parametric Modeling', in 27th Annual Conference of the Association for Computer Aided Design in Architecture, 2007, pp. 222–229.

[13] Z. C. Qian, 'Design Patterns: Augmenting Design Practice in Parametric CAD Systems', Simon Fraser University, Burnaby, Canada, 2009.

[14] S. Chien et al., 'PARADE: A pattern-based knowledge repository for parametric designs', in Emerging Experience in Past, Present and Future of Digital Architecture, Proceedings of the 20th International Conference, 2015.

[15] R. Hudson, 'Strategies for parametric design in architecture: An application of practice led research.', University of Bath, 2010.

[16] H. Su and S. Chien, 'Revealing patterns: Using parametric design patterns in building façade design workflow', in Living Systems and Micro-Utopias: Towards Continuous Designing, Proceedings of the 21st International Conference on Computer-Aided Architectural Design Research in Asia, 2016, pp. 167–176.

[17] I. Caetano and A. Leitão, 'Weaving Architectural Façades: Exploring algorithmic stripe-based design patterns', in "Hello, Culture" – Proceeding of the 18th International Conference on Computer Aided Architectural Design Futures, 2019, pp. 1023–1043.

[18] L. Santos et al., 'Uncertainty in daylight simulations of algorithmically generated complex shading screens', in 17th International IBPSA Building Simulation Conference, 2021.

[19] I. Caetano et al., 'Case Studies on the Integration of Algorithmic Design Processes in Traditional Design Workflows', in Learning, Adapting and Prototyping, Proceedings of

the 23rd International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2018, 2018, pp. 129–138.
[20] I. Caetano et al., ‘Creativity inspired by analysis: An algorithmic design system for designing structurally feasible façades’, in RE: Anthropocene, Proceedings of the 25th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2020, Volume 1, 2020, pp. 599–608.
[21] I. Caetano and A. Leitão, ‘Integration of an Algorithmic BIM Approach in a Traditional Architecture Studio’, J. Comput. Des. Eng., vol. 6, pp. 327–336, 2019.
[22] J. Jiang and Y. Ma, ‘Path planning strategies to optimize accuracy, quality, build time and material use in additive manufacturing: A review’, Micromachines, vol. 11, no. 7, 2020.
[23] M. Sammer et al., ‘From Visual Input to Visual Output in Textual Programming’, in Intelligent & Informed: Proceedings of the 24th International Conference of the Association for Computer-Aided Architectural Design Research in Asia, 2019, vol. 1, pp. 645–654.
[24] T. Wortmann and B. Tunçer, ‘Differentiating Parametric Design: Digital workflows in contemporary architecture and construction’, Des. Stud., vol. 53, pp. 173–197, 2017.
[25] D. Nezamaldin, ‘Parametric Design with Visual Programming in Dynamo with Revit: The conversion from CAD models to BIM and the design of analytical applications’, KTH Skolan för arkitektur och samhällsbyggnad, 2019.
[26] A. Leitão et al., ‘Illustrated Programming’, in ACADIA 2014: Design Agency, 2014, pp. 291–300.
[27] A. Leitão et al., ‘Programming Languages for Generative Design: A Comparative Study’, Int. J. Archit. Comput., vol. 10, no. 1, pp. 139–162, 2012.
[28] M. A. Zboinska, ‘Hybrid CAD/E Platform Supporting Exploratory Architectural Design’, CAD Comput. Aided Des., vol. 59, pp. 64–84, 2015.
[29] P. Janssen et al., ‘Möbius: A parametric modeller for the web’, in Living Systems and Micro-Utopias: Towards Continuous Designing, Proceedings of the 21st International Conference of the Association for Computer-Aided Architectural Design Research in Asia, 2016, pp. 157–166.
[30] P. Janssen, ‘Visual Dataflow Modelling: Some thoughts on complexity’, in Fusion - Proceedings of the 32nd eCAADe Conference, 2014, no. December, pp. 305–314.
[31] A. A. Petunin et al., ‘On the new Algorithm for Solving Continuous Cutting Problem’, IFAC-PapersOnLine, vol. 52, no. 13, pp. 2320–2325, 2019.
[32] M. L. Jensen et al., ‘Toolpath Strategies for 5DOF and 6DOF Extrusion-Based Additive Manufacturing’, Appl. Sci., vol. 9, no. 19, 2019.
[33] B. Ezair et al., ‘Volumetric covering print-paths for additive manufacturing of 3D models’, Comput. Aided Des., vol. 100, pp. 1–13, 2018.
[34] Y. Jin et al., ‘Optimization of tool-path generation for material extrusion-based additive manufacturing technology’, Addit. Manuf., vol. 1–4, no. Complete, pp. 32–47, 2014.
[35] D. Ding et al., ‘A tool-path generation strategy for wire and arc additive manufacturing’, Int. J. Adv. Manuf. Technol., vol. 73, no. 1, pp. 173–183, 2014.
[36] A. V Shembekar et al., ‘Trajectory Planning for Conformal 3D Printing Using Non-Planar Layers’, 2018.
[37] H. T. Bui et al., ‘Tool Path Planning Optimization for Multi-Tool Additive Manufacturing’, Procedia Manuf., 2019.
[38] D. Coupek et al., ‘Reduction of Support Structures and Building Time by Optimized Path Planning Algorithms in Multi-axis Additive Manufacturing’, Procedia CIRP, vol. 67, pp. 221–226, 2018.
[39] R. Dewil et al., ‘An improvement heuristic framework for the laser cutting tool path problem’, Int. J. Prod. Res., vol. 53, no. 6, pp. 1761–1776, 2015.
[40] R. Dewil, ‘On Generating Tool Paths for Laser Cutters’, 2014.

[41] S. U. Sherif et al., ‘Sequential optimization approach for nesting and cutting sequence in laser cutting’, J. Manuf. Syst., vol. 33, no. 4, pp. 624–638, 2014.

[42] R. Dewil et al., ‘Construction heuristics for generating tool paths for laser cutters’, Int. J. Prod. Res., vol. 52, no. 20, pp. 5965–5984, 2014.

[43] T. A. Makarovskikh and A. V Panyukov, ‘Software for the Problem of Constructing Cutting Tool Paths inCAD/CAM Systems for Technological Preparation of Cutting Processes’, Autom. Remote Control, vol. 82, no. 3, pp. 468–480, 2021.

[44] D. Zhao et al., ‘3D sand mould printing: a review and a new approach’, Rapid Prototyp. J., vol. 24, no. 2, pp. 285–300. 2018.

[45] E. Pagone et al., ‘Sustainability Assessment of Rapid Sand Mould Making Using Multi-criteria Decision-Making Mapping’, in Sustainable Design and Manufacturing 2020, 2021, pp. 345–355.

[46] M. Upadhyay et al., ‘3D printing for rapid sand casting—A review’, J. Manuf. Process., vol. 29, pp. 211–220, 2017.

[47] K. Salonitis et al., ‘Improvements in energy consumption and environmental impact by novel single shot melting process for casting’, J. Clean. Prod., vol. 137, pp. 1532–1542, 2016.

[48] O. Estrada et al., ‘Energy gap method (EGM) to increase energy efficiency in industrial processes: Successful cases in polymer processing’, J. Clean. Prod., vol. 176, pp. 7–25, 2018.

[49] J. Thiel et al., ‘Advancements in Materials for Three-Dimensional Printing of Molds and Cores’, Int. J. Met., vol. 11, no. 1, pp. 3–13, 2017[Online]. Availablehttps://doi.org/10.1007/s40962-016-0082-y.

[50] R. Ramakrishnan et al., ‘3D Printing of Inorganic Sand Moulds for Casting Applications’, in WGP Congress 2014, 2014, vol. 1018, pp. 441–449.

[51] R. Singh, ‘Process capability study of rapid casting solution for aluminium alloys using three-dimensional printing’, Int. J. Automot. Mech. Eng., vol. 4, pp. 397–404, 2011.

[52] E. S. Almaghariz et al., ‘Quantifying the Role of Part Design Complexity in Using 3D Sand Printing for Molds and Cores’, Int. J. Met., vol. 10, no. 3, pp. 240–252, 2016.

[53] D. Snelling et al., ‘The Effects on 3D Printed Molds on Metal Castings’, in 2013 International Solid Freeform Fabrication Symposium, 2013.

[54] M. Chhabra and R. Singh, ‘Obtaining desired surface roughness of castings produced using ZCast direct metal casting process through Taguchi’s experimental approach’, Rapid Prototyp. J., vol. 18, no. 6, pp. 458–471, Jan. 2012.

[55] S. S. Gill and M. Kaplas, ‘Efficacy of powder-based three-dimensional printing (3DP) technologies for rapid casting of light alloys’, Int. J. Adv. Manuf. Technol., vol. 52, no. 1, pp. 53–64, 2011.

[56] M. Chhabra and R. Singh, ‘Mathematical Modeling of Surface Roughness of Castings Produced Using ZCast Direct Metal Casting’, J. Inst. Eng. Ser. C, vol. 96, no. 2, pp. 145–155, 2015.