From Architectural Requirements to Physical Creations An Algorithmic-based Approach for Façade Design

Inês Caetano[1*], **António Leitão**[1], **Francisco Bastos**[2]

* Corresponding author
1 INESC-ID/Instituto Superior Técnico, University of Lisbon, Portugal, ines.caetano@tecnico.ulisboa.pt
2 CiTUA, Centro de Inovação em Território, Urbanismo e Arquitetura, University of Lisbon, Portugal

Abstract
Computation-based approaches are increasingly important in architecture, namely Algorithmic Design (AD), which is based on the use of algorithms to generate designs. Besides enhancing design exploration, AD helps architects deal with recurrent design changes and with the pressure to quickly obtain results. Moreover, AD supports the search for better-performing solutions that satisfy environmental demands. Unfortunately, the complexity and specialized knowledge required by AD are still restraining architects due to the amount of effort and time needed to implement the necessary algorithms. To make AD and design optimization techniques more accessible, we propose a theoretical framework to help architects with the algorithmic generation, evaluation, and manufacturing of a large variety of designs, by following a continuous design workflow that merges the typically detached design stages.
In order to be useful, the framework needs to focus on a domain of application, and, in this paper, we target the development of buildings' façades due to their aesthetical and environmental relevance.
We evaluate the framework in the context of a real case study resulting from a collaboration between a team of architects specialized in AD and a traditional design studio.

Keywords
Algorithmic design, façade panels, geometric patterns, fabrication techniques

DOI 10.7480/jfde.2020.2.4543

1 INTRODUCTION

Nowadays, digital tools and computation-based design approaches play a relevant role in the conception, analysis, and production of architecture. Algorithmic Design (AD) is one such approach that uses algorithms to generate designs. AD promotes design exploration and the emergence of complex shapes and geometric patterns that would be difficult and costly to produce without it (Moneo, 2001). Additionally, AD enables design optimization, i.e., the search for better-performing design solutions, and fabrication (Gibson, Rosen, & Stucker, 2010).

Unfortunately, AD is not yet widespread, mainly due to its complexity and the specialized knowledge required. The challenge, then, is to make AD more accessible and more purposeful. To this end, we explore its application in building design, more specifically, in envelopes. In order to help architects with the algorithmic development of a large variety of façade designs, we propose a theoretical framework that implements, in a single and continuous workflow, the design exploration, analysis, optimization, and rationalization stages. Differently from other approaches, the one proposed here is based on a set of mathematical principles that underlie the computational description of façade designs, independently of the specific implementation of those principles. This allows the framework to be portable across different design tools.

2 BACKGROUND

2.1 ON THE IMPORTANCE OF THE FAÇADE

Considered the "public face of architecture" (Baper & Hassan, 2012), the architectural façade is regarded as the outer layer of a building, separating the indoor environment from the outside. This element plays an important role in building design, having many associated functions: environmental performance (Picco, Lollini, & Marengo, 2014), acting as a filter towards energy-reduction, daylighting, ventilation, and good indoor quality spaces (El Sheikh, 2011); structural function (Al-Kodmany & Ali, 2016), integrating the building's structural elements; cultural identity (Schittich, 2006; Schulz, 1971), marking the aesthetical evolution of the city and its architecture; and urban communication (Picco et al., 2014; Stojšić, 2017; Venturi, Brown, & Izenour, 1972).

With the increase of architectural design constraints, including environmental concerns, economic limitations, and tight deadlines, façade design grew in complexity. Moreover, designing intricate geometries/patterns that are low-cost and simultaneously have good performance is a hard and error-prone process. AD helps overcome these problems by allowing a more flexible and iterative design process that shortens the time needed to explore different design solutions. Still, designing façades using an AD approach requires the use of different programming techniques and algorithmic strategies, for which most architects are not yet well-trained. Consequently, its use often ends up limiting their creative and design exploration processes. We believe that solving this challenge entails the delivery of a flexible computational framework that tames the complexity of procedural modelling, analysis processes, and optimization strategies.

Previous research structured the intricacy of architectural façades (Moussavi & Kubo, 2006; Otani & Kishimoto, 2008; Pell, 2010). Inspired by works confirming that sets of algorithms can be generalized and reused in the exploration of new designs (Chien, Su, & Huang, 2015; Qian, 2009; Woodbury, Aish, & Kilian, 2007), Caetano, Santos, & Leitão (2015) developed an algorithmic-oriented framework for façade design that identifies sets of predefined algorithms and strategies addressing the modelling needs of different designs. Nevertheless, this framework is restricted to initial design stages and does not consider design analysis or manufacturing stages.

2.2 ON THE IMPORTANCE OF PERFORMANCE

The idea of an informed and conscientious design process has always been present in architecture, but it has evolved throughout the last decades. Currently, it persuades architects to focus more on the exploration of multiple solutions in order to find those with the best performance regarding different metrics. This trend is particularly relevant in the design of façades, which are increasingly approached as filters between the environment and interior spaces that aim at providing better indoor environmental quality (e.g., regarding natural light, ventilation, and thermal comfort), while reducing the buildings' energy costs (El Sheikh, 2011).

Improving buildings' performance clearly benefits from an AD approach, namely, in automating the generation of design variations and in analysing and optimizing their performance (Kolarevic, 2005). Moreover, this combination enables a continuous workflow where design exploration, analysis, and optimization are merged into a single design process.

Nevertheless, analysis and optimization processes are often time-consuming, lack real-time feedback (Choo & Janssen, 2015), and, generally, hardly evaluate all possible solutions, usually resulting in just a partial improvement (Nguyen, Reiter, & Rigo, 2014). Moreover, optimization processes are scarcely applied at early design stages (Lin & Gerber, 2013) and are typically postponed to later phases where the integration of design changes is much more toilsome, hindering the search for better performing solutions (Konis, Gamas, & Kensek, 2016). Finally, given the variety of available methods, the selection of the most suitable optimization algorithm for a given problem becomes a non-trivial task (Nguyen et al., 2014).

3 METHODS

The aim of this research is to support architects in obtaining better-performing design solutions in a shorter amount of time, while respecting their design intent. To that end, we propose a theoretical framework to help architects with the algorithmic generation, evaluation, and manufacturing of a large variety of designs, by following a continuous design workflow that covers design exploration, evaluation, and rationalization.

This investigation is part of a larger research project focusing on the design of algorithmic façades, which has already studied (1) the use of Computation-based Design approaches in architecture, (2) the current architectural design processes, especially those using AD, and their limitations, (3) the trends found for modelling different geometries and geometric patterns in architecture, and (4) the most used analysis, optimization, and rationalization processes in architecture. Based on the previous findings, we conducted this investigation in a three-stage process that includes:

- Developing a mathematical theory of the building façade (Section 4.1);
- Implementing the proposed theory in an algorithmic-based framework, containing a collection of predefined algorithms and known design strategies specifically targeting buildings' façades (sections 4.2 and 4.3);
- Evaluating the framework's suitability for architectural design (section 5).

The first stage involved the analysis of the current trends in façade design patterns and the structuring of a classification to organize them mathematically.

The second stage entailed two main tasks: implementing the previously developed mathematical-based perspective into an algorithmic framework specifically dedicated to façade designs and structuring the combination of the different types of algorithms implemented. The result is a framework tailored to support an architectural design process based on informed design decisions, which promises to:

- Facilitate the use of AD processes in the design, analysis, optimization, and rationalization of façade designs, providing algorithms that address a wide variety of geometries, evaluation techniques, and design scenarios;
- Solve the current problem of having to use multiple design tools and specialized digital environments, since the available algorithms are portable between the different specialized tools, being capable of automatically adapting the obtained model to the specific modelling requirements of each tool;
- Inform architects about the most suitable algorithms and design strategies for a specific design scenario.

Finally, in the last stage, we evaluated the framework in the development of several case studies of diversified design characteristics and distinct design scenarios, from an early design stage to preparation for manufacturing. In this paper we present one of such case studies, which resulted from our collaboration with a design studio without AD experience, and we discuss the advantages of using the framework throughout its development.

4 THEORY: ALGORITHMIC UBIQUITY IN FAÇADE DESIGN

Notwithstanding AD's advantages in the design, analysis, and optimization of buildings, its use requires architects to learn these new techniques. Considering this, we propose a theoretically supported algorithmic-based framework for façade design that encompasses the entire design process. For each scenario, it provides a set of algorithms and strategies according to the different design stages, helping the architect not only to select the most appropriate algorithms, but also to avoid extensive and potentially error-prone programming efforts. This results in AD models that are geometrically precise and informed about their performance. The next sections present the proposed theory and its implementation in an algorithmic framework specialized in façade design.

4.1 FRAMEWORK STRUCTURE

Architectural practice is highly dependent on the specific circumstances of the design brief and, thus, it is unlikely that the exact same approach can be used in different projects. This applies to both non-AD and AD approaches. Since we are addressing the latter, we must deal with the variability that typically occurs in architectural design in a perspective that is simultaneously understood by a computer.

Computational tools operate by following a set of instructions described using Programming Languages (PLs). Despite the expressive power of modern PLs, they still follow a formal perspective increasingly inspired by the universally understood language of mathematics. Therefore, this research considered the formalism of mathematics in (1) structuring the overall AD theory, (2) defining the algorithmic strategies, and (3) implementing them in an algorithmic framework specialized in façade design.

A mathematical theory that supports AD tasks for the design exploration, analysis, optimization, and rationalization stages, should provide:

- Flexibility, to support architectural design variability;
- Diversity, to address a large range of design problems;
- Multiplicity of criteria, to accurately evaluate different scenarios;
- Coherency, to correctly link the diverse types of information to the different design stages in a single and continuous workflow.

To cover a wide range of design scenarios, we explored some existing modular programming and Design Pattern (DP) techniques (Qian, 2009; Woodbury et al., 2007), which reuse and adapt ideas from different projects. Having these techniques available at initial design stages allows architects to spend much less time and effort in the programming task, avoiding writing algorithms from scratch every time they start a new design. Accordingly, despite not considering all possible design scenarios, the proposed theory responds to the most common ones and can be adapted to more specific ones.

The theory is structured in a multi-dimensional classification that organizes the available algorithms by typology:

- Geometry: defines the building façade geometry;
- Pattern: explores the combination of geometric elements and transformations;
- Distribution: distributes the geometric elements;
- Optimization: adapts the design solution to a fitness goal;
- Rationalization: makes a balance between design intent and feasibility.

The next section explains the classified algorithms and how they are combined with each other.

4.2 FRAMEWORK APPLICATION

In general, our framework provides $\mathbb{R} \to \mathbb{R}$, $\mathbb{R} \to \mathbb{R}^2$, $\mathbb{R}^2 \to \mathbb{R}^2$, and $\mathbb{R}^2 \to \mathbb{R}^3$ algorithmic functions for each of the different typologies, which are then combined with one another through different function compositions. To better explain this process, we first introduce some notions regarding the framework's implementation and, then, we explain its application.

To simplify the mathematical presentation, all surface-related functions in the framework $S(u,v)$ range over the domain $0 \le u \le 1,\ 0 \le v \le 1$. Moreover, fundamental operators that can be arbitrarily combined are also available, namely the one-dimensional linear variation function $linear(a,b) = \lambda(t).a + (b-a)t$, wherein $\lambda(t)$ is the λ-calculus notation for an anonymous function with parameter t (Leitão, 2014), and the (paradoxical) constant "variation" function $constant(c) = \lambda(t\ldots).c$. Note that the latter returns an anonymous function that can be combined with functions of any number of arguments, adapting its number of arguments according to those of the combined function.

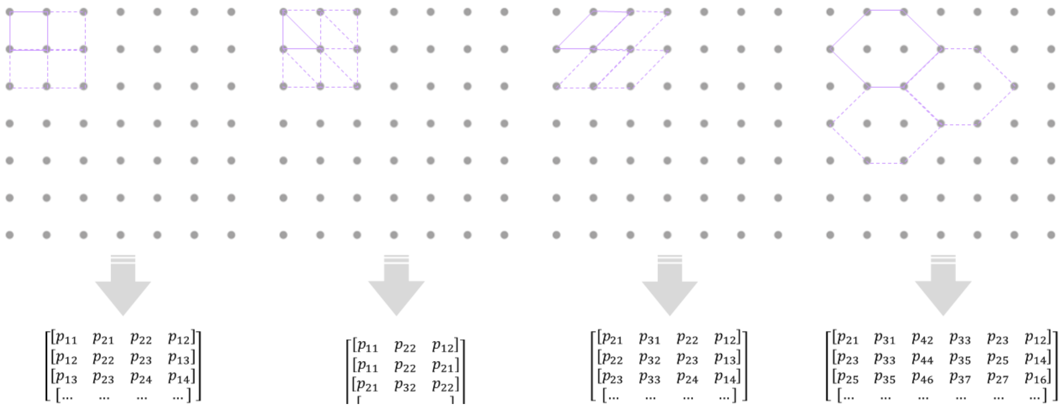
Given that the façade geometry domain is usually two-dimensional, it is necessary to extend the domain of the above one-dimensional variations into $\mathbb{R}^2$ and, to this end, we use higher-order functions (HOFs), i.e., functions that receive other functions as arguments and/or return other functions as results (Leitão, 2014). We define the two HOFs $\dim_u(f) = \lambda(u,v).f(u)$ and $\dim_v(f) = \lambda(u,v).f(v)$, which cause a function f to vary only in one dimension, i.e., in the u or v dimension accordingly. Finally, for function composition, we provide an operator to generalize this operation:

$$\circ(f,\ g_1,\ \cdots,\ g_n) = \lambda(x_1,\ \cdots,\ x_m).f\left(g_1(x_1,\ \cdots,\ x_m),\ \cdots,\ g_n(x_1,\ \cdots,\ x_m)\right)$$

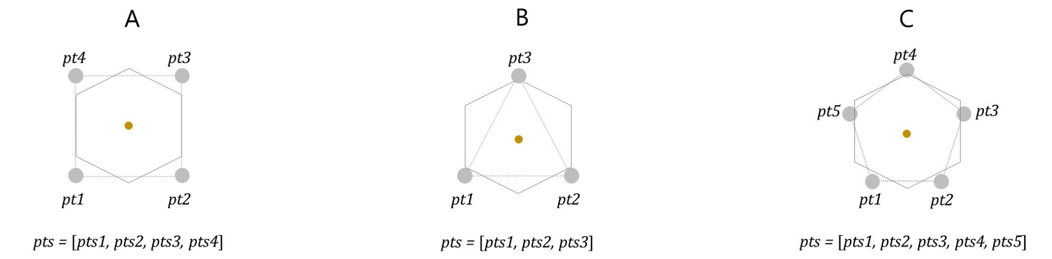
To obtain a simplified notation, we define $u \otimes lim = \dim_u\left(linear(0, lim)\right)$ and $v \otimes lim = \dim_v\left(linear(0, lim)\right)$, wherein lim is the domain's limit, we treat all numbers n in a function context as $constant(n)$, and we represent any first-order function f that receives functional arguments $g_1, \cdots, g_n$ as $\circ(f, g_1, \cdots, g_n)$. Based on this, $f \times g$ is the same as $\circ(\times, f, g)$.

By using HOFs, we can move from the numeric space, in which numbers are combined using numeric operators, into the functional space, where functions are combined using functional operators. In this space, a straight façade can be defined by the $straight$ algorithm $straight(w,h) = \lambda(u,v).XYZ(u \times w,\ 0,\ v \times h)$ of the Geometry category, which produces a matrix of vectors of points representing a $w \times h$ planar parametric surface, and whose equivalent representation is now $straight(w,h) = XYZ(u \otimes w,\ 0,\ v \otimes h)$. Then, to create a geometric pattern on that same surface, we select one or more algorithms from both the Pattern and Distribution categories: the first one creates the shape(s) to be applied, whereas the second one distributes the shapes on the surface domain.

Note that any function in the Distribution category receives as input not the surface itself but a matrix containing the surface points on which the distribution will be made; information provided by the functions of the Geometry category. As a result, they return another matrix with the same points but rearranged to generate different types of distributions. For example, the function $grid_{squares}$ receives a matrix of points describing a surface and returns a matrix of four points vectors describing squared areas on that same surface. This is visible in Fig. 1, which, from left to right, shows the distributions implemented by $grid_{squares}$, $grid_{triangles}$, $grid_{parallelograms}$, and $grid_{hexagons}$ when applied to a $straight$ surface.


[p11 p21 p22 p12] [p12 p22 p23 p13] [p13 p23 p24 p14] [... ... ...]
[p11 p22 p12] [p11 p22 p21] [p21 p32 p22]
[p21 p31 p22 p12] [p22 p32 p23 p13] [p23 p33 p24 p14] [... ... ...]
[p21 p31 p42 p33 p23 p12] [p23 p33 p44 p35 p25 p14] [p25 p35 p46 p37 p27 p16] [... ... ...]
FIG. 1 Algorithms to iterate elements: a conceptual representation of how the surface points are organized in the framework.

Now, consider the Pattern function $shape_{polygon}$ that creates regular polygons. It receives as input the set of points where to centre each polygon (pts) (Fig. 2A-C), as well as the number of sides (n_{sides}) and radius size (r_{size}). To distribute regular-polygonal shaped elements on a straight façade following a matrix of points distributed in squares, we have to combine the three algorithms $straight$, $grid_{squares}$ and $shape_{polygon}$: the first informs the program about the points describing the surface, the second rearranges those points into a matrix of square vertices, and the last one creates a regular polygon fitting each square. Fig. 2D illustrates the result of this combination. The use of a functional representation therefore simplifies the composition of these functions: e.g., $\circ(grid_{squares}, straight)(w,h)$ can be simplified to $grid_{squares}(straight(w,h))$. To facilitate the mathematical representation of functions that deal with matrices containing vectors, e.g., the function $shape_{polygon}$, we introduce the notion of *broadcasting*. Broadcasting allows us to apply a function f to an array of elements, even if it has a different number of dimensions from the other arguments. Syntactically, broadcasting is represented by the *dot syntax* $f.(args\ldots)$ and it can be applied in single or nested calls $f.(g.(args\ldots))$. Based on this, $shape_{polygon}(\circ(grid_{squares},\ straight)(w,h),\ constant(n_{sides}),\ constant(r_{size}))$ can be simplified to $shape_{polygon}.(grid_{squares}(straight(w,h)), n_{sides}, r_{size})$.


A
B
C
pt4
pt3
pt1
pt2
pts = [pts1, pts2, pts3, pts4]
pt3
pt1
pt2
pts = [pts1, pts2, pts3]
pt4
pt5
pt3
pt1
pt2
pts = [pts1, pts2, pts3, pts4, pts5]

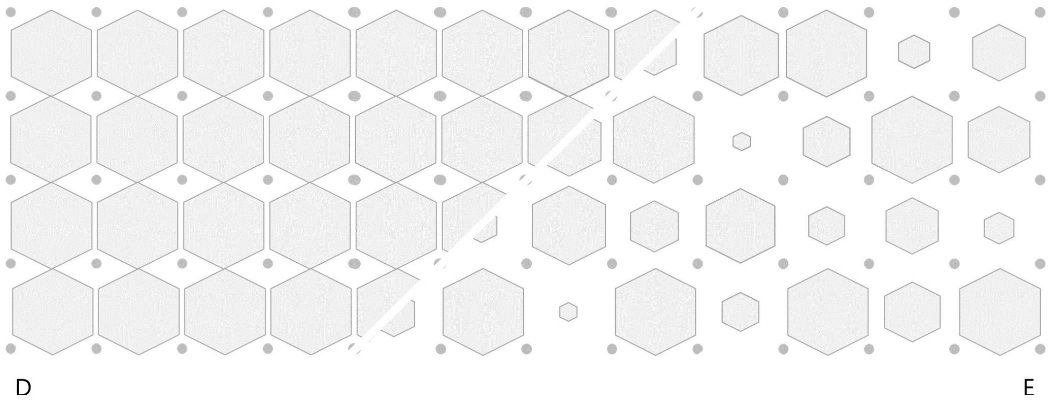


FIG. 2 Algorithms to create and transform geometrical elements: a conceptual representation of the algorithm $shape_{polygon}$; A to C illustrate how the algorithm adapts the polygonal shape to the set of points received (pts); D represents the result of the composition of functions $shape_{polygon}$ and $grid_{squares}$, and *E* the result of the previous composition plus the scale algorithm.

Equally important is the Transformation category, which provides algorithms to deal with different types of geometric transformations, including contracting/expanding (represented by the algorithm $scale$), reflection (algorithm $mirror$), rotation (algorithm $rotate$), among others. Combining these algorithms with those available in the Pattern category allows the generation of a wider variety of more dynamic geometric patterns.

As another example, consider a pattern based on hexagonal apertures, horizontally and vertically aligned, and whose size vary randomly (Fig. 2E). The algorithms selected are, therefore:

- $grid_{squares}$ to distribute the apertures in a squared grid;
- $shape_{polygon}$ to create the hexagonal apertures;
- $scale$ to control their size.

Note that the last algorithm returns, based on the surface position (pt) and a rule to control the transformation process (f_{scale}), the information about which scaling factors to use and where to use them.

We can combine these three algorithms through function composition and, given that the parameters pts and r_{size} of the algorithm $shape_{polygon}$ are informed by the output of the algorithms $grid_{squares}$ and $scale$, respectively, we have the following composition $shape_{polygon}\left(grid_{squares}\left(ptss\right), n_{sides}, r_{size} \times scale\left(pt, f_{scale}\right)\right)$. Note that $grid_{squares}$ receives as argument a matrix of points ($ptss$) representing a surface, whereas $scale$ receives a position on that surface (pt). Assuming the surface is flat, we use the algorithm $straight$ to obtain the surface points $ptss$ needed for both functions $grid_{squares}$ and $scale$. Although $ptss$ constitutes a direct input for the former, i.e., $grid_{squares}\left(straight\left(w,h\right)\right)$, in the case of $scale$, each of its points has to be applied individually, suggesting the application of the broadcasting technique, i.e., $scale.(straight(w,h),\ f_{scale})$. In the end, we obtain the composition $shape_{polygon}.(grid_{squares}(straight(w,h)),\ n_{sides},\ r_{size} \times scale.(straight(w,h),\ f_{scale}))$, in which we apply the broadcasting technique in nested function calls.

To optimize or rationalize the developed solution, we use the algorithms available in the Optimization and Rationalization categories to either drive the geometric modifications made to the elements or control the number of different elements composing the final solution.

For example, to improve the natural ventilation of the previous design (Fig. 2D), we select an appropriate algorithm from the Optimization category, namely, $opt_{ventilation}$, which receives information about the design's desired performance, context, and geometric characteristics, returning a matrix containing the ideal aperture area for each façade opening. We use $opt_{ventilation}$ to iteratively inform the parameter f_{scale} of the $scale$ algorithm, i.e., $scale.(straight(w,h),\ opt_{ventilation}(args\ldots))$, and, consequentially, the parameter r_{size} of the $shape_{polygon}$ algorithm, obtaining the following composition $shape_{polygon}.(grid_{squares}(straight(w,h)),\ n_{sides},\ r_{size} \times scale.(straight(w,h),\ opt_{ventilation}(args\ldots)))$.

Finally, to minimize the manufacturing costs of a design solution, while addressing its construction viability, we have the algorithms of the Rationalization category, which includes both algorithms $discretize$ and $tallying$, among others. These are important for balancing the design intent and performance and the number of different façade elements, as well as for identifying and counting different typologies. Mathematically speaking, a discretization process aims at reducing the number of values accepted by a continuous variable, converting a continuous set of values into a finite range grouped into n intervals. Similarly, $discretize$ allows setting the maximum number n_{max} of intervals of the discretized range, which, in turn, corresponds to n_{max} different elements.

In practice, to reduce the manufacturing cost of the previous design, we select the $discretize$ algorithm to control the number of different opening sizes. This is important because the fewer window sizes are allowed, the less expensive the solution's manufacturing will be. When combined with the previous algorithms, $discretize$ affects the range of values received by the $scale$ algorithm, being that the fewer they are, the more viable the solution gets. Note that we now have two algorithms, $opt_{ventilation}$ and $discretize$, informing the parameter f_{scale} of the $scale$ algorithm, originating the following composition $scale.(straight(w,h),\ discretize(n_{max},\ opt_{ventilation}(args\ldots)))$. In practice, it is the composition $discretize(n_{max}) \circ opt_{ventilation}(args\ldots)$ that balances the design's performance with the maximum allowed number of opening sizes: $opt_{ventilation}$ finds the best values for f_{scale}, which in turn can originate a very broad set of values, while $discretize$ reduces that same range. In the end, we are provided with a set of better-performing design solutions of minimized manufacturing costs and different geometric characteristics, from which we can select the one that best fits both the design's context and intent.

To proceed to the manufacturing stage, it is important to access information about the range of values used in the final design and where they are used. To this end, we can use the $tallying$ algorithm to force the $discretize$ algorithm to store this information. Regarding the previous example, this means we can know the exact positions of the different elements on the façade, allowing us to manage their placement on site.

4.3 WORKFLOW

In order to take advantage of the framework, we propose the use of the workflow in Fig. 3. It starts with the architect's design intent (arrow 1), which he then implements using the framework's algorithms (arrow 2). During this stage (Design Exploration), he tests several design variations and, when satisfied with the result (arrow 3), he moves to the following stage (Design Optimization) and focuses on improving the design regarding different criteria, such as natural lighting illumination, energy consumption, privacy levels, among others. To this end, he selects and combines one or more optimization algorithms from the framework, obtaining the design solution that best fits the requirements (arrow 4). The architect then evaluates this solution and if it does not meet his design intent, he goes back to testing further design variations and their subsequent optimization. Otherwise, the architect proceeds to the ensuing stage (Design Rationalization), which aims at assessing the solution's construction feasibility (arrow 5). For such, he applies the framework's algorithms to control the diversity of elements composing the design solution, thus reducing its manufacturing cost and material waste. The solutions that balance both design intent and feasibility become the set of acceptable design options (the Design Space in Fig. 3) from which the architect then chooses the one that most pleases him (arrow 6).

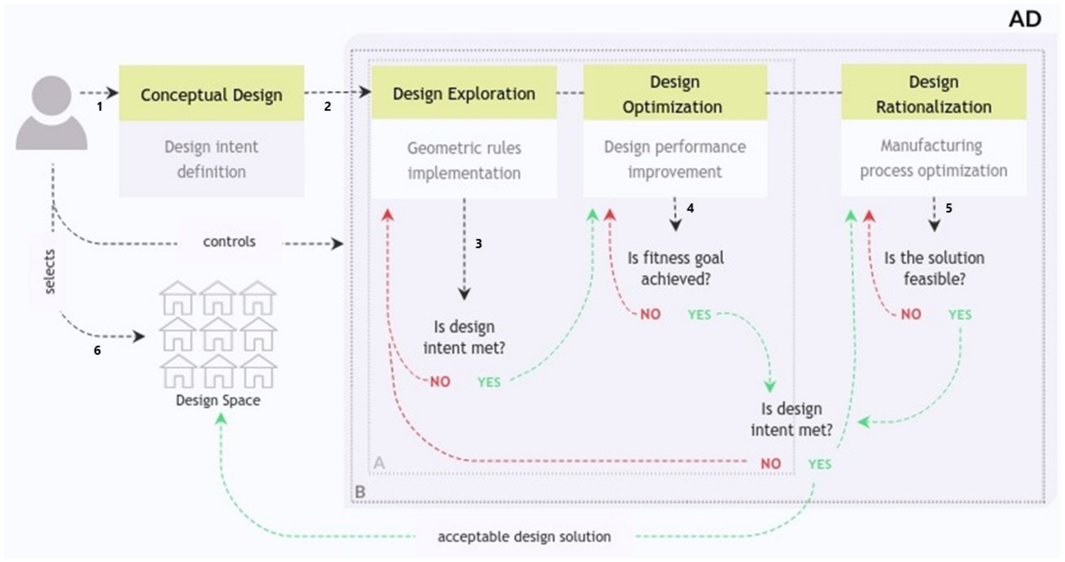


FIG. 3 Design workflow supported by the algorithmic framework: 1. definition of the architect's design intent; 2. algorithmic implementation of the design intent using the framework's algorithms; 3. geometric implementation and design exploration; 4. improvement of the design solution regarding a certain fitness criterion; 5. optimization of the solution's manufacturing costs and waste; 6. selection of the final design solution; A. the iterative process of adjusting the solution's geometric characteristics to meet the fitness goal set; B. the iterative process of balancing the solution's geometry, performance, and feasibility.

In the next section, we evaluate both the framework's suitability for architectural design and the proposed workflow in a real case study entirely based on the algorithms provided and embracing the previously mentioned design stages.

5 RESULTS AND DISCUSSION

We were recently involved in the design of a façade for a residential building. The original design concept for the façade was a composition of geometrically different *cobogó* bricks (Rytel, 2013). As architects specialized in AD, we collaborated with a traditional design studio and applied the framework not only to geometrically explore different options based on the initial design intent, but also to improve, rationalize, and fabricate the final solution. This collaboration demonstrated the framework's flexibility and adaptability to multiple design scenarios and processes, while highlighting the growing need to integrate teams of differently skilled architects.

5.1 CASE STUDY: *COBOGÓ* FAÇADE PANELS

Cobogó is a type of hollow brick used for air circulation and light penetration control in geographical areas with a hot and humid climate (Rytel, 2013), regulating the interior space comfort of buildings. In Portugal, these elements were widely used during the Modern architecture period.

The design studio aimed at integrating some of the local culture in this case study, whose location was Lisbon (Sousa & Batista-Bastos, 2015). Therefore, the architects decided to use *cobogó*-inspired elements in the design of the building's façade. Nevertheless, they wanted to avoid the use of standardized *cobogó* elements and take advantage of geometric variations to improve the building's natural lighting and ventilation performance. The result was an intricate façade pattern combining the influence of the local architecture with the search for an architectural identity.

The next sections explain the design process using the proposed framework.

5.2 DESIGN EXPLORATION

At this stage, together with the design studio, we established a simple geometric rule that, when varying its parameters, originated a set of different brick options. The algorithm implementing such rule receives as input a set of points defining the brick's corners (Fig. 4A) and, based on these points, it then outlines the brick's shape and calculates its centre (Fig. 4B). Thereafter, it allows the user to choose between two options: link the brick's central position either with its edges' midpoints or its vertices (Fig. 4C). Lastly, given the need to integrate different natural lighting and ventilation degrees on the façade, we combined an additional algorithm to create bricks with different opacity levels (Fig. 4D). Fig. 4E illustrates all possible brick options for a given set of four points (Fig. 4A).

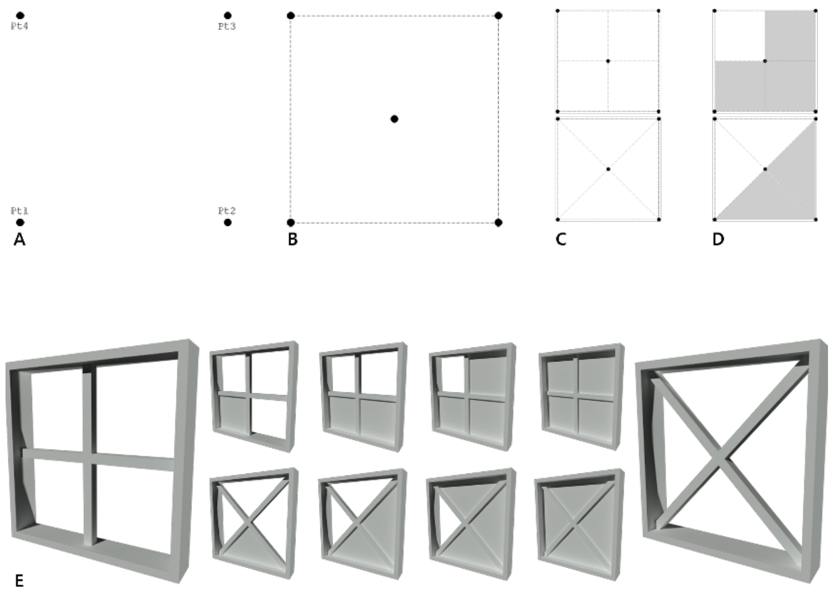


FIG. 4 The application of the cobogó geometric rule when receiving as input a set of four points: A. the received points defining the frame's shape; B. the creation of the frame; C. the selection between two possible rules – connect the frame's centre to the corners or to the edges' midpoints; D. the application of different opacity degrees; E. the range of possible solutions.

In a second stage, we developed different façade designs using these elements. We explored different combinations between the algorithms available in the Distribution category and the algorithm producing each brick. This ensured the bricks' shape was adapted to correctly fit the façade surface. Fig. 5 shows some of the stacking possibilities explored.

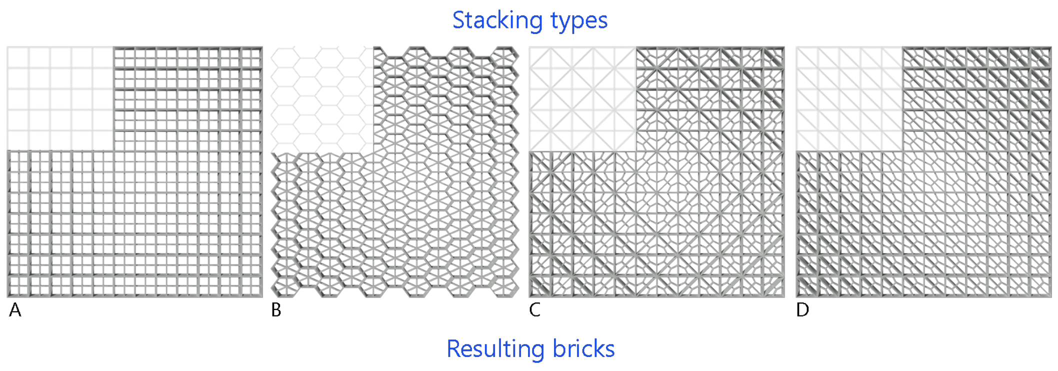


FIG. 5 The different stacking options selected: each scheme from A to D represents a different type of stacking, as well as the set of bricks resulting from its combination with the algorithm producing each brick (the brick's shape adapts to allow a perfect fit).

After deciding on the stacking option(s) to explore, we implemented gradual geometric variations to the façade design that responded to the initial design intent (Fig. 6):

- Apply both geometric rules (Fig. 4C);
- Explore different opacities (Fig. 4E);
- Combine two types of stacking (Fig. 5A and 5D).

One of the advantages of AD is the ability to generate a wide range of design solutions in a short time. However, this raises the problem of how to present a potentially large set of solutions to the architect. This was already addressed by others (Erhan, Wang, & Shireen, 2015), however no consensus on a practical solution currently exists. Therefore, we opted for gradually exploring the design space in a process guided by the architect himself: the decision-making process results from repeating the presentation of design solutions to the architect and using his feedback to guide the generation of additional solutions until the architect is satisfied. So, despite not presenting the entire design space to the architect, we allowed him to control the navigation through the set of potential solutions, among which he could choose the one that most pleased him.

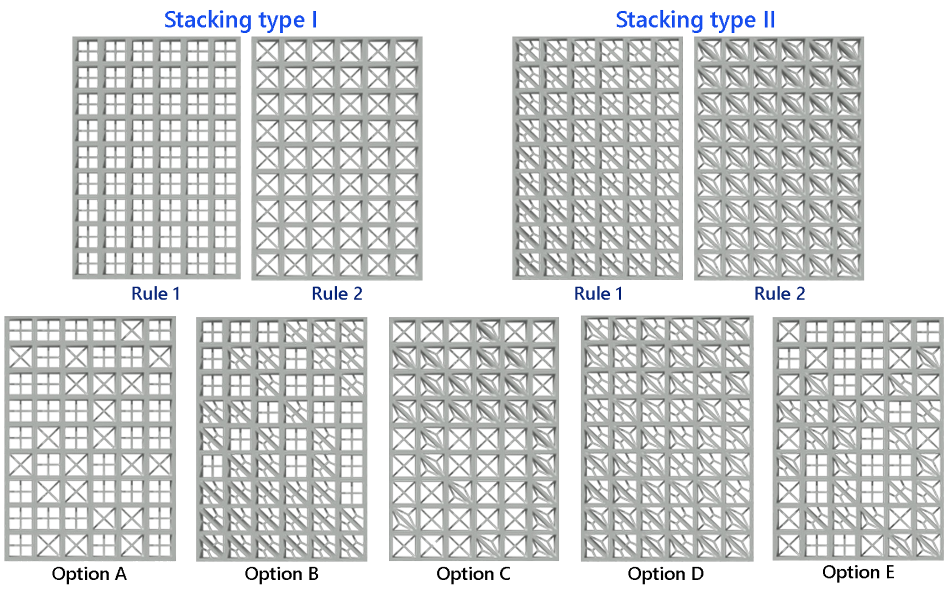


FIG. 6 The evolution of the façade's geometric pattern: option A combines both rules and the stacking type I; option B combines rule 1 and both stacking types; option C combines rule 2 and both stacking types; option D combines both rules and the stacking type II; and option E combines all rules and stacking types.

5.3 DESIGN IMPROVEMENT

This stage aimed at improving the façade design of our case study. Therefore, we considered more design requirements regarding (1) privacy levels, (2) natural lighting, and (3) natural ventilation, that were entirely based on the architect's own practical experience. To address these, we took advantage of another set of algorithms to control the geometric characteristics of each brick in order to respond to the following criteria:

- In private areas and circulation spaces, the fraction and size of the voids should be smaller, meaning that the use of triangular bricks should supplant those of squared ones in these areas;
- In living rooms, terraces, and courtyards, a larger number of squared, less opaque bricks were advisable to provide better views and greater amounts of natural daylight.
- In areas exposed to direct sunlight there should be a preference for more opaque bricks;
- To promote a general natural ventilation, the façade opacity levels should always be higher at its middle than at its top or base.

These requirements originated a set of values for each spatial functionality, each one matching the desired natural lighting, privacy, and natural ventilation levels (Fig. 7) as well as the ratio of squared-shaped to triangular-shaped elements (Fig. 8). Integrating this information with the framework's algorithms allowed the team to iteratively visualize several design options and then, based on their evaluation, produce new solutions exploring only the most favourable design characteristics. Similar to the previous design stage, this process enabled the team to gradually explore the design space in a controlled and conscientious way, quickly refining the façade design solution. Fig. 7 and 8 synthetize the values established for the final solution, after several design iterations.

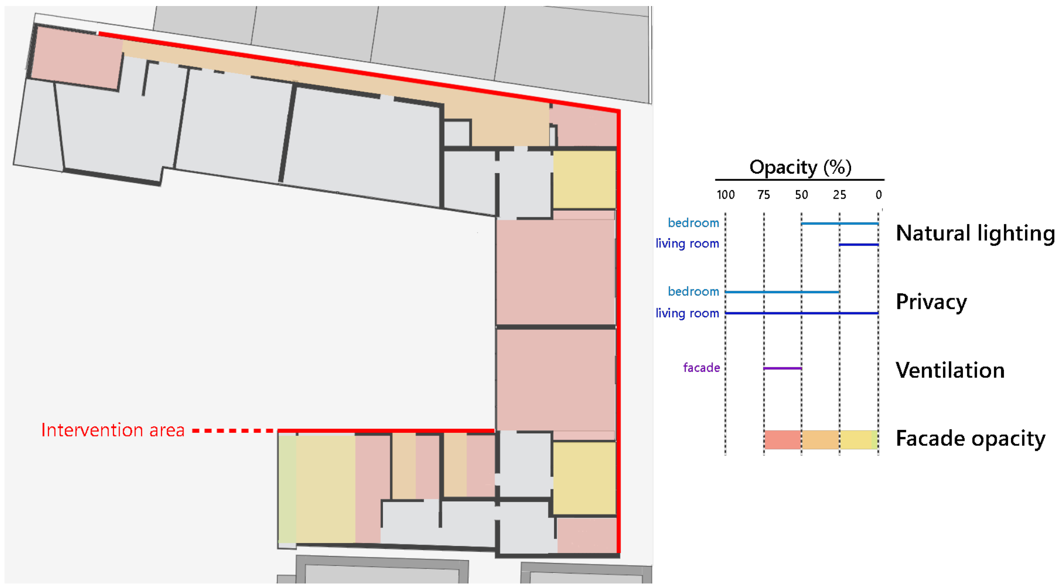


FIG. 7 Case study plan: the red lines represent the façade intervention areas and the coloured areas the different opacity levels explained in the right-side diagram.

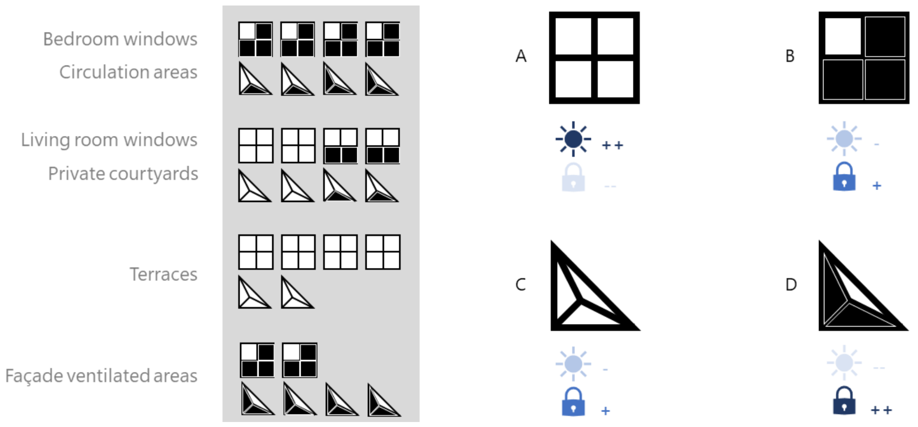


FIG. 8 Left: the ratios of squared-shape to triangular-shaped elements by indoor spatial functionality; right: privacy (the lock symbol) and natural light (the sun symbol) levels provided by each brick type, e.g., A has the highest natural light and the lowest privacy levels, whereas D has the opposite; B and C provide higher levels of privacy than of natural light.

In the end, the process was entirely controlled by the architects, who were responsible for the iterative design choices and derivations throughout the process. Therefore, the use of the framework gave them autonomy and consciousness about the impact their decisions had on the design's final performance. This feedback had a major role in achieving improved design solutions that performed according to the requirements set by the architects and that simultaneously satisfied the initial design intent.

5.4 DESIGN RATIONALIZATION

As AD allows for further design exploration and geometric complexity, it is important to ensure the feasibility of the obtained solutions. Therefore, the framework integrates algorithmic strategies that make their manufacturing viable. These strategies, mainly those of rationalization, become even more relevant in designs whose elements vary in shape, material, size, etc., and where it is, thus, important to control:

- the number of different elements;
- the positioning of elements;
- the manufacturing strategy to follow.

Given the case study's geometric characteristics, the team first considered a manufacturing strategy based on the traditional use of ceramic and moulds to produce each façade element. We used our framework to obtain each brick typology frequency, list of positions (Fig. 9), and 3D model of the mould. This information allowed us to proceed to the following stage, i.e., the bricks' fabrication and subsequent assembly.

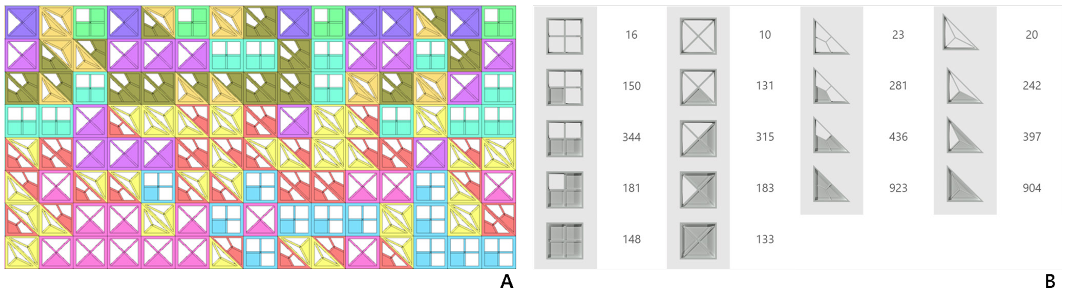


FIG. 9 The application of the framework's rationalization algorithms: A. a section of the façade's 3D model identifying each brick typology (each colour corresponds to a brick type); B. the frequency table of the typologies used.

During this stage, the architects evaluated the constraints resulting from the use of customized bricks. These included, on the one hand, the cost and production of the eighteen types of customized bricks used in the final design solution, whose manufacture required eighteen customized moulds, suggesting the use of the framework's rationalization algorithms to reduce the number of brick typologies. On the other hand, there were façade areas that should act as movable window shutters, which constituted a more serious constraint due to the massiveness and fragility of the *cobogó* bricks.

To deal with these problems, the team decided to follow a different fabrication strategy based on lightweight wood façade panels representing the composition of different *cobogó* elements, allowing to maintain the geometric diversity and plastic expression of the original solution while removing the need for mould production. To evaluate this strategy, we decided to manufacture the wood panels by layers that, when overlapped, originated the desired thicknesses and shapes (Fig. 10). This process again took advantage of the framework's algorithms to, first, generate the façade panels with different geometric compositions and levels of opacity and, then, automatically produce the information needed for their subsequent manufacturing. Fig. 11 illustrates some of the prototypes developed and Fig. 12 the final design solution.

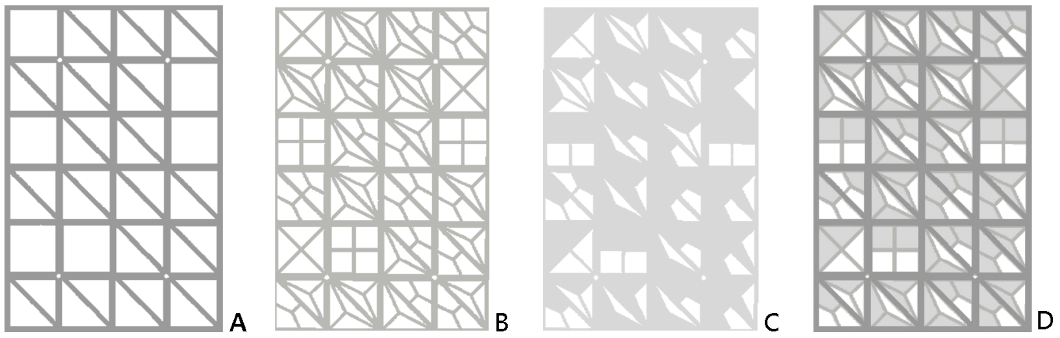


FIG. 10 The set of layers of a panel: A. the outer frame delineating the general shape of the cobogó elements; B. the middle frame outlining their interior elements; C. the inner layer delimiting the existing void and opaque areas; D. the wood panel after overlapping the different layers in the following order: A-B-C-B-A.

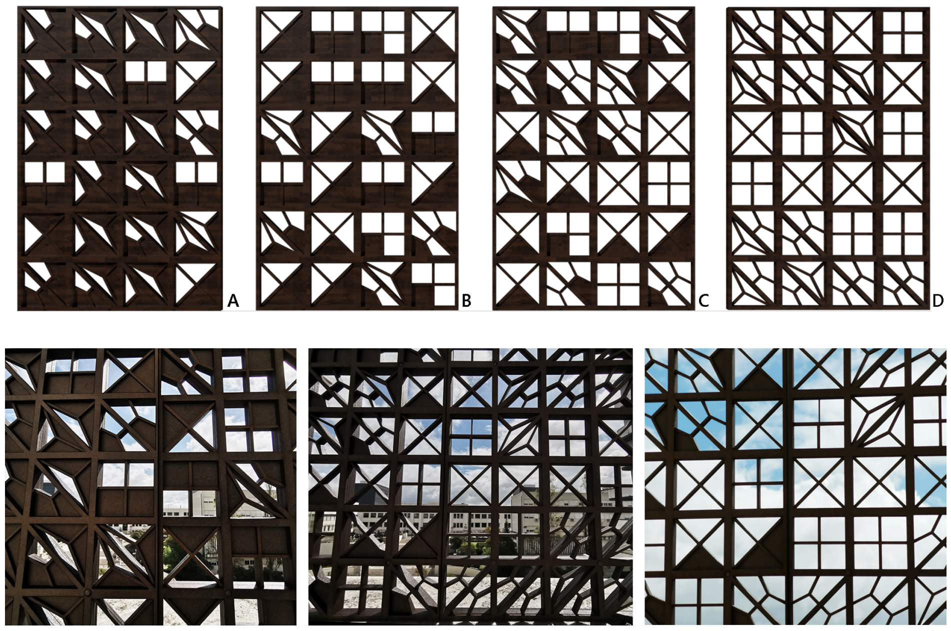


FIG. 11 Prototypes with different opacity levels and ratios of squared to triangular-shape elements. Top: four typologies whose opacity decreases from A to D; Bottom: some of the produced prototypes.

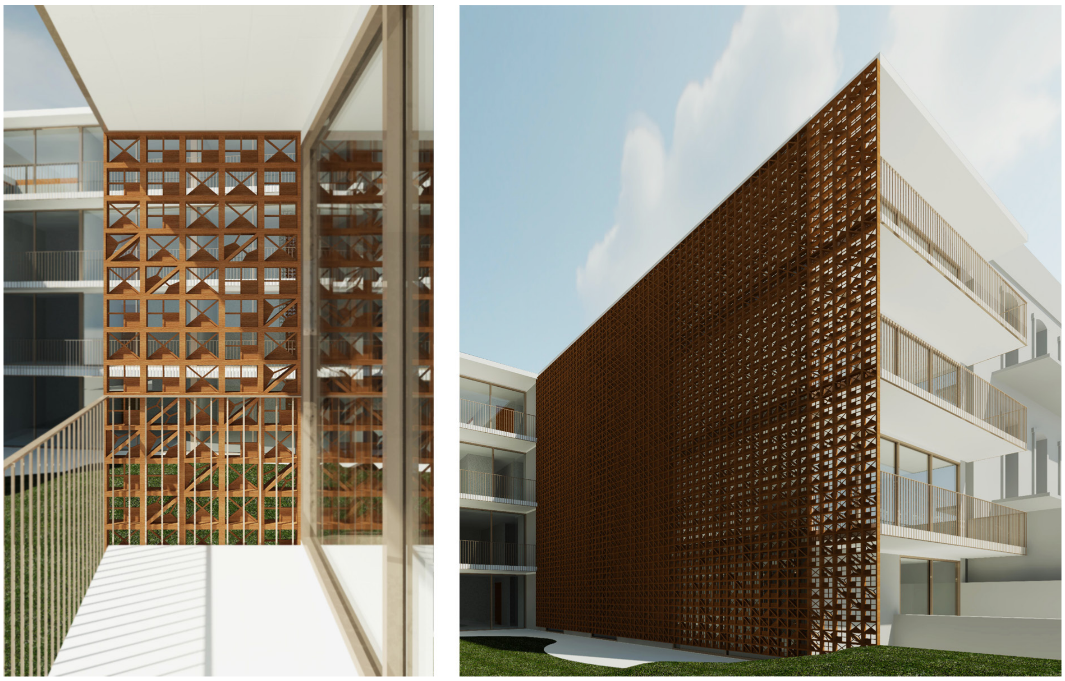


FIG. 12 Two perspective views of the final solution's 3D model in Revit.

5.5 EVALUATION/DISCUSSION

This case study was done in the context of a collaboration between a team of AD specialists and a traditional design studio with no AD experience. After understanding the architect's design intents and his difficulties to make several alternative drawings and model experiments, it was proposed to try an AD approach. We applied the AD framework from an initial design stage, to develop the underlying geometric rules of the façade's elements, until a final design stage, to manufacture the set of panels. The evaluation criteria for measuring the success of this process was the ability to:

- Elaborate a rational for dealing with the principles imposed by conceptual intents;
- Create different geometric patterns following the same principles;
- Balance the most visually interesting patterns with the initial programmatic intents.

By comparing this design workflow with a non-AD one, we can highlight their main differences and the advantages resulting from the use of an AD approach and, more specifically, of the proposed framework.

Regarding the Design Exploration stage (Fig. 3), the generation of different *cobogó* elements would have been more limited if we had used a non-AD design approach: not only would it have taken more time to explore each of the suggested ideas, but it would have also hindered the application of small-to-large design changes during the entire process. The process would probably result in a different set of façade elements from those obtained; it would either not present as many options or not respond as well to the architect's design intentions. In contrast, the use of the framework in an AD approach proved to facilitate not only the exploration of different element geometries, by implementing the set of geometric rules resulting from the initial design intent, but also the application of continuous and iterative design changes, by automatically adapting the solutions being explored according to the team's feedback (Fig. 3, arrow 3), which in turn is an asset for the ensuing design phases, as we will show.

Considering the Design Improvement stage, where the focus was to improve the solution obtained (Fig. 3, arrow 4), the framework played an active role in enhancing the design options regarding their natural ventilation performance and privacy levels. In practice, it supported a continuous, two-way design process between this stage and the previous one, allowing us to iteratively adjust the geometric characteristics of the façade's elements in order to achieve the fitness goals set by the architects. The rectangular area A in Fig. 3 illustrates this process. The result was a balance between the façade's desired geometric characteristics and overall performance.

Furthermore, the different inputs provided to the algorithm that shaped each *cobogó* brick produced geometric variations that were totally controlled by the team: depending on the values received, the algorithm could generate, at each position, either a squared brick or two triangular ones whose geometric subdivision followed either rule 1 or rule 2 and had a smaller or larger opaque area. As previously mentioned, the allowed options and the ratio between them were both set by the team, meaning that most of the variations obtained were within the design space established by the initial design intent.

Moreover, given the limited time provided for this iterative process, the team could explore more design options, as well as increasingly improve their corresponding performance. If we were to use a non-AD design process instead, it would require manually changing the design after each evaluation process so as to include the modifications needed for the following iteration, which would mean more time spent at each iteration and, therefore, less time available to explore other design variations. Hence, as the range of options analysed would be smaller, the final façade design would probably have a poorer performance compared to those produced with the framework.

Regarding the Design Rationalization stage, whose main concern was the design's feasibility (Fig. 3, arrow 5), using the framework proved to facilitate the search for a constructively viable solution that simultaneously met both performance and aesthetic requirements. The rectangular area B in Fig. 3 represents this process. During this process, we controlled the maximum number of brick typologies allowed, forcing the algorithm to automatically adapt to the design's constraints. Furthermore, the algorithm also produced the information regarding the different typologies used in the final design (including the number of bricks of each typology and their exact positions on the façade), providing the team with the necessary data for the ensuing manufacturing stage. In the end, the team received a set of solutions fitting the design requirements (Fig. 3's design space), from which we could choose the final design (Fig. 3, arrow 6).

Table 1 compares a non-AD design approach with an algorithmic one, evidencing both their advantages and shortcomings. Although AD has a higher initial investment and steeper learning curve for most architects, its long-term application is advantageous to the architectural design practice, as it:

- Improves the creative process – it is flexible, automates repetitive tasks, facilitates design changes, and supports higher levels of design complexity.
- Promotes design improvement – it automates the *generation-evaluation-regeneration* process that improves the design in a viable time.
- Facilitates design rationalization – it supports the generation of designs with controlled variations between elements.

– Supports a continuous workflow – it enables the design changes made at one design stage to have a direct impact on the others.

TABLE 1 Comparison between a non-AD and an AD approach on each design stage

	DESIGN STAGE ACTIVITIES	NON-AD APPROACH	AD APPROACH
Design Exploration	Design concept definition	Intuitive	Reasoned
	Design instantiation	Hard	Easy
	Design space explored	Small	Large
	Implementation of small changes	Easy	Easy
	Implementation of large changes	Difficult (sometimes unfeasible)	Easy
	Propagation of changes	Tedious and error-prone	Automatic and accurate
	Application of iterative changes	Time-consuming and hardworking	Fast and mechanized
	Feedback on the impact of changes	Reduced	Increased
	Model flexibility	Small	Large
	Traceability	Obvious	Non-obvious
	Ability to support complexity	Low	High
Design Improvement	Integration of evaluation results	Manual	Automatic
	Execution of iterative evaluations	Time-consuming and error-prone	Automatic
	Design space explored	Limited	Extensive
Design Rationalization	Ability to support rationalization	Reduced	Increased
	Integration of rationalization rules	Manual	Automatic
	Generation of rationalized models	Manual	Automatic
	Design space explored	Limited	Extensive
	Production of analytical models	Manual	Automatic
	Production of construction information	Dependent on the design tool	Automatic

Typically, any design exploration process starts with the architect using a non-AD approach: he thinks and materializes his design ideas through sketches and handmade schemes to help him establish the design intent. When already delineated, the architect can explore his design idea by following the same approach throughout the entire design process or he can instead opt to transition to an AD approach and benefit from its advantages, for example, in the geometric exploration, in the automation of repetitive design tasks, in the application of rationalization processes, among others. Despite its multiple advantages, the decision to use AD should take into account the initial investment required. When the project being developed is geometrically simple and does not require design variations, the use of AD is not justified, as the initial investment needed to algorithmically implement the design is not paid off later. In most other cases, however, the adoption of an AD approach becomes advantageous, as shown in Table 1.

According to Table 1, the geometric exploration stage generally benefits from the use of an AD approach. Although a non-AD approach is more intuitive, resorting to freehand sketching and physical modelling, when the process is deepened, more time is spent on design exploration, as it relies on repetitive manual design changes. Therefore, within the same time period, it produces fewer complex design options whose corresponding models are also less flexible. Moreover, the ability of AD to automatically and coherently integrate small-to-large design changes in the model not only provides visual feedback on the impact of changes, but also enables the iterative application of changes to meet aesthetic requirements. Nevertheless, identifying the parts of the code that correspond to the parts of the model that we want to change is not always straightforward or obvious in an AD approach. Contrarily, although the same design changes in a non-AD scenario are more easily tracked because they are directly made to the model, they require manually modifying the model or even redoing the model from scratch. This repetitive and time-consuming process therefore hinders the iterative application of design changes, thus resulting in only slightly improved design solutions.

Table 1 also evidences the differences between both approaches in terms of improving designs. In the non-AD approach, improving a solution requires the manual production of different designs to then evaluate them. Additionally, before proceeding to the next iteration, the evaluation results must be incorporated through manual changes to the design model. As this cycle has to be repeated multiple times until better solutions are found, it becomes extremely time-consuming, tedious, and error prone. Differently from the non-AD approach, AD automates the production of design variations, allowing us to evaluate a greater number of solutions when compared to a non-AD approach.

Table 1 also highlights the potential of AD for rationalization processes. AD allows controlling the variety of elements composing a design solution by iteratively producing variations of it with a decreasing number of different elements. This process results in a set of rationalized solutions from which we can choose the one that best balances design intention and feasibility. In contrast, the same scenario in a non-AD design approach would suffer from the same limitations regarding design improvement processes. Moreover, while an AD approach can automatically produce all the constructive information needed for the construction stage, a non-AD approach depends on the capabilities of the modelling tool being used.

Important lessons can be learned from this experiment. Firstly, when the design can be geometrically parameterized, the design exploration benefits from AD because it makes it possible to explore a wider range of solutions. Conversely, when the project is geometrically simple and has few potential design variations, there is no need to adopt an AD approach, as the required effort might not pay off. Secondly, when the design includes both aesthetical and performance requirements, switching to AD is advantageous, as the initial investment required to algorithmically implement the design is recovered in the iterative evaluation and balancing of the different requirements. Finally, when the project is geometrically complex and/or its manufacture is likely to benefit from rationalization processes, it is advantageous to use AD, as it allows the design feasibility to be increased while maintaining its design identity.

6 CONCLUSION AND FUTURE WORK

The relevance of Algorithmic Design (AD) in architecture is a reality. It not only provides design flexibility and the ability to explore larger design spaces, but also supports the search for better performing design solutions and their subsequent fabrication. Still, in many countries, AD is not yet widespread due to its complexity and the specialized knowledge required. To make AD and design optimization techniques more accessible to architects, we focused on their application, particularly in the design of building envelopes.

In this paper, we proposed a theoretical framework to support architects with the generation, evaluation, and manufacturing of several façade designs by following a single and continuous design workflow. Firstly, we explained the framework's mathematical structure and how to apply and combine the provided algorithms. Then, we evaluated the framework in a real case study: a façade design composed of different elements inspired by traditional *cobogó* bricks. The final solution resulted entirely from the application of the framework's algorithms to (1) explore the geometric characteristics of the façade's design; (2) improve the design according to a set of requirements; (3) rationalize the design to make its manufacturing process viable; and (4) produce the necessary information for the fabrication of the façade elements.

The case study demonstrated the framework's ability to help the development of a façade design from its conceptual design phase to its actual fabrication, reducing the effort and time needed. Moreover, it proved the framework's capacity to inform the architect throughout the entire design process, providing him with a more conscious and autonomous view on the solutions' performance regarding different requirements and their constructive viability. Finally, its application in a traditional design context, i.e., within a design studio without AD experience, proved the framework's flexibility and adaptability to multiple design scenarios and processes, while evidencing the current need to integrate architects with AD skills within architectural design studios.

As future work, we plan to continue developing geometric rules inspired by traditional *cobogó* bricks and evaluate different fabrication strategies that are more appropriate for other materials (e.g., 3D printing with polymers and customized moulds with concrete). We also intend to continue refining the framework by applying it to other façade designs and extending it with additional analysis tools and optimization techniques.

Acknowledgments

This work was supported by national funds through *Fundação para a Ciência e a Tecnologia* (FCT) with references UIDB/50021/2020 and PTDC/ARTDAQ/31061/2017, and by the PhD grant under contract of FCT with reference SFRH/BD/128628/2017. We also thank *Atelier dos Remédios* for participating in this research.

References

Al-Kodmany, K., & Ali, M. M. (2016). An Overview of Structural and Aesthetic Developments in Tall Buildings Using Exterior Bracing and Diagrid Systems. International Journal of High-Rise Buildings, 5(4), 271–291.

Baper, S. Y., & Hassan, A. S. (2012). Factors Affecting the Continuity of Architectural Identity. American Transactions on Engineering & Applied Sciences, 1(3), 227–236.

Caetano, I., Santos, L., & Leitão, A. (2015). From Idea to Shape, From Algorithm to Design: A Framework for the Generation of Contemporary Façades. In The next city - New technologies and the future of the built environment [16th International Conference CAAD Futures 2015 (p. 483). São Paulo, Brazil.

Chien, S., Su, H., & Huang, Y. (2015). 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 of the Association for Computer-Aided Architectural Design Research in Asia.

Choo, T.-S., & Janssen, P. (2015). Performance-based parametric design. In Emerging Experience in Past, Present and Future of Digital Architecture, Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (Vol. 1, pp. 1–10).

El Sheikh, M. M. (2011). Intelligent building skins: Parametric-based algorithm for kinetic façades design and daylighting performance integration. Ph.D Thesis. University of Soutern California. https://doi.org/10.1017/CBO9781107415324.004

Erhan, H., Wang, I. Y., & Shireen, N. (2015). Harnessing design space: A similarity-based exploration method for generative design. International Journal of Architectural Computing, 12(4), 217–236. https://doi.org/10.1260/1478-0771.13.2.217

Gibson, I., Rosen, D., & Stucker, B. (2010). Development of Additive Manufacturing Technology. In Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing (pp. 19–43). Springer.

Kolarevic, B. (2005). Towards the performative in Architecture. In B. Kolarevic & A. M. Malkawi (Eds.), Performative Architecture: Beyond Instrumentality (pp. 203–214). London, United Kingdom: Spon Press.

Konis, K., Gamas, A., & Kensek, K. (2016). Passive performance and building form: An optimization framework for early-stage design support. Solar Energy, 125(February), 161–179. https://doi.org/10.1016/j.solener.2015.12.020

Leitão, A. (2014). Improving generative design by combining abstract geometry and higher-order programming. In Rethinking Comprehensive Design: Speculative Counterculture, Proceedings of the 19th International Conference on Computer- Aided Architectural Design Research in Asia CAADRIA 2014 (pp. 575–584).

Lin, S. E., & Gerber, D. J. (2013). Designing-in performance: evolutionary energy performance feedback for early stage design. In 13th Conference of International Building Performance Simulation Association (pp. 386–393).

Moneo, R. (2001). The Thing Called Architecture. In C. Davidson (Ed.), Anything (pp. 120–123). New York: Anyone Corporation.

Moussavi, F., & Kubo, M. (Eds.). (2006). The Function of Ornament. Actar.

Nguyen, A.-T., Reiter, S., & Rigo, P. (2014). A review on simulation-based optimization methods applied to building performance analysis. Applied Energy, 113, 1043–1058. https://doi.org/10.1016/j.apenergy.2013.08.061

Otani, M., & Kishimoto, T. (2008). Fluctuating Patterns of Architecture Façade and their Automatic Creation. In CAADRIA 2008 [Proceedings of the 13th International Conference on Computer Aided Architectural Design Research in Asia] (pp. 375–382).

Pell, B. (2010). The Articulate Surface: Ornament and Technology in Contemporary Architecture. Germany: Birkhäuser GmbH.

Picco, M., Lollini, R., & Marengo, M. (2014). Towards energy performance evaluation in early stage building design: A simplification methodology for commercial building models. Energy & Buildings, 76, 497–505. https://doi.org/10.1016/j.enbuild.2014.03.016

Qian, Z. C. (2009). Design Patterns: Augmenting Design Practice in Parametric CAD Systems. SIMON FRASER UNIVERSITY, Burnaby, Canada.

Rytel, G. (2013). The Influence of Climate on the Forms of Brazilian Modernist Archiecture in the Years 1925-1960. Kwartalnik Architektury I Urbanistyki, 4, 57–78.

Schittich, C. (Ed.). (2006). Building Skins. Birkhäuser. https://doi.org/10.11129/detail.9783034615082

Schulz, C. N. (1971). Existence, space & architecture. New York: Praeger. Stamps.

Sousa, S., & Batista-Bastos, M. (2015). O tempo e a Diferença: Análise e Readaptação num Edifício em Lisboa [Time and Difference: Analysis and Readaptation in a Building of Lisbon]. Cadernos de Arquitectura e Urbanismo, 22(33), 58.

Stojšić, M. (2017). (New) Media Façades: Architecture and/as a Medium in Urban Context. AM Journal, (12), 135–148. https://doi.org/10.25038/am.v0i12.173

Venturi, R., Brown, D. S., & Izenour, S. (1972). Learning from Las Vegas. MIT Press.

Woodbury, R., Aish, R., & Kilian, A. (2007). Some Patterns for Parametric Modeling. 27th Annual Conference of the Association for Computer Aided Design in Architecture, 222–229. Retrieved from http://moodle.ncku.edu.tw/file.php/2687/assignments/ParametricPatterns.pdf