UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

An Algorithmic Framework for the Design, Optimization, and Fabrication of Facades

Inês Alexandra do Côrro Caetano

Supervisor: Doctor António Paulo Teles de Menezes Correia Leitão

Co-Supervisor: Doctor Francisco Manuel Caldeira Pinto Teixeira Bastos

Thesis approved in public session to obtain the PhD Degree in Architecture

Jury final classification: Pass with Distinction and Honour

2023

UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

DrAFT 2.0

An Algorithmic Framework for the Design, Optimization, and Fabrication of Facades

Inês Alexandra do Côrro Caetano

Supervisor: Doctor António Paulo Teles de Menezes Correia Leitão

Co-Supervisor: Doctor Francisco Manuel Caldeira Pinto Teixeira Bastos

Thesis approved in public session to obtain the PhD Degree in

Architecture

Jury final classification: Pass with Distinction and Honour

Jury

Chairperson: Doctor Teresa Frederica Tojal Valsassina Heitor, Instituto Superior Técnico, University of Lisbon

Members of the Committee:

Doctor Maria Gabriela Caffarena Celani, School of Civil Engineering, Architecture and Urban Design, University of Campinas, Brazil

Doctor José Pedro Ovelheiro Marques de Sousa, Faculty of Architecture, University of Porto

Doctor José Nuno Dinis Cabral Beirão, Lisbon School of Architecture, University of Lisbon

Doctor Sara Eloy Cardoso Rodrigues, ISCTE-IUL School of Technology and Architecture

Doctor Francisco Manuel Caldeira Pinto Teixeira Bastos, Instituto Superior Técnico, University of Lisbon

Doctor Patrícia Isabel Mendes Lourenço, Instituto Superior Técnico, University of Lisbon

Funding Institutions

Foundation for Science and Technology

2023

Título DrAFT 2.0: Metodologia Algorítmica para a Exploração, Otimização e Fabricação de Fachadas de Edifícios.

Nome Inês Alexandra do Côrro Caetano

Doutoramento em Arquitetura

Orientador Doutor António Paulo Teles de Menezes Correia Leitão

Co-Orientador Doutor Francisco Manuel Caldeira Pinto Teixeira Bastos

Resumo

A arquitetura tem continuamente explorado os avanços tecnológicos para melhorar a representação e produção de soluções arquitetónicas. Ultimamente, as abordagens de Design Algorítmico (DA) têm tido um papel importante na prática da arquitetura, permitindo ultrapassar várias das limitações encontradas na resolução de problemas mais complexos. Contudo, o DA não é trivial de usar pois requer experiência em programação. No sentido de facilitar a resolução de problemas arquitetónicos complexos e não convencionais, esta tese propõe uma metodologia e um *framework* para sistematizar o uso de DA, avaliando a sua aplicação no desenvolvimento de soluções de fachada.

Para tal, são identificados os problemas e estratégias recorrentes que são relevantes incluir na metodologia, os quais são posteriormente categorizados numa estrutura modular e formalizados numa perspetiva matemática. O objetivo é tornar a aplicação da metodologia o mais universal possível, sendo assim capaz de responder a diversas práticas e problemas de projeto e ao mesmo tempo adaptar-se à variabilidade e singularidade dos processos arquitetónicos.

Para avaliar a validade e rigor da proposta, a metodologia é implementada no *framework* algorítmico DrAFT 2.0 e usada para reproduzir um conjunto de soluções já existentes. De modo a provar a sua aplicabilidade e mais-valia para a prática arquitetónica, o *framework* é usado num conjunto de casos de estudo reais respondendo a diversas intenções criativas e problemas projetuais.

A análise dos resultados e do feedback obtido nestas experiências permite responder às questões de investigação, provando a capacidade da proposta em simplificar o uso de DA na resolução de problemas complexos e ao mesmo tempo aumentar a flexibilidade e a eficiência dos processos de projeto. A análise anterior também revela a capacidade da proposta em

potenciar o pensamento criativo e crítico dos arquitetos, aumentando não só as possibilidades de projeto consideradas, mas também a probabilidade de se obterem melhores soluções.

Palavras-chaves: Projeto Algorítmico; Processo Criativo; Análise e Otimização de Projetos; Fabricação Digital; Projeto de Fachadas**.**

Title DrAFT 2.0: An Algorithmic Framework for the Design, Optimization, and Fabrication of Facades

Abstract

Architecture has always explored the latest technological advances in terms of design representation and production. Nowadays, Algorithmic Design (AD) approaches play a critical role in the conception and production of architectural solutions, overcoming many of the practice's limitations in solving more complex design problems, such as those of facade design. Nevertheless, AD is a complex design approach that requires programming skills and deviates from the visual nature of architecture. This research addresses this problem by focusing on the field of facade design, proposing a methodology and framework that reduces the complexity of AD in solving complex facade design problems.

To that end, the thesis examines recurrent problems and strategies in facade design, identifying the existing design patterns, formulating their underlying principles, and organizing them in a categorical way. To generalize the proposal and facilitate its application in different design practices and briefs, the thesis adopts a mathematical approach and organizes the formulated principles in a modular way. Since the aim is to support design-to-fabrication AD workflows, it is important to ensure the adaptability of the proposal to the context-specificity and variability of architectural practice, as well as to the various methods and tools used.

To assess the proposal's validity and accuracy, the mathematical methodology is implemented in the algorithmic framework DrAFT 2.0 and is used to reproduce an existing set of building facades. Then, to evaluate the proposal's applicability and usefulness for architectural practice, the framework is applied in a set of application studies involving architects with different AD skills and responding to various design intents and problems. After reflecting on the research findings, the thesis concludes that the proposal simplifies AD, increasing the flexibility and efficiency of facade design processes and thus the design space considered. It also demonstrates how the proposal enhances creative and critical thinking processes, increasing the likelihood of achieving better solutions.

Keywords: Algorithmic Design; Creative Process; Design Analysis and Optimization; Digital Fabrication; Facade Design.

Acknowledgements

First of all, I would like to thank my supervisors, Professor António Menezes Leitão and Professor Francisco Teixeira Bastos, for guiding this thesis and for their availability, tireless support and, above all, their enthusiasm for this research. This work would not have been possible without their endless contributions and therefore much of its merit is theirs as well. I am also grateful for the trust they have placed in my work and in my ideas. They provided me with total freedom in the selection of the research methods, as well as in the development of the research solutions, which helped, if not the research itself, at least my personal growth as a researcher. I also sincerely appreciate the numerous opportunities provided throughout these years of research, both in terms of theoretical contributions and practical collaborations. In particular, I owe a special thanks to professor António Menezes Leitão for all the time he dedicated to this research and, above all, to my journey as both a PhD student and a junior researcher at INESC-ID. He continuously invested in my educational and technical growth while relentlessly mentoring me throughout this challenging process.

I also give a special thanks to my parents, my family, and my close ones, who had a critical role in this long journey and made this thesis possible. I thank them particularly for all their emotional support and, above all, their understanding and companionship.

I am also extremely grateful to the members, and ex members, of the Algorithmic Design for Architecture (ADA) research group, headed by Professor António Menezes Leitão, for all their feedback, fellowship, knowledge sharing, and collaboration, particularly to Catarina Belém, Inês Pereira, Luís Santos, and Renata Castelo Branco. I strongly value their multiple research contributions, their tireless support, and the time and effort spent on final improvements and revisions.

I also want to thank my close friends, who directly or indirectly made part of this journey, especially for their understanding and patience.

Lastly, I would like to thank INESC-ID for the support and recognition given to this research, as well as the design studios Atelier dos Remédios and FOR-A for their collaborations.

Institutional acknowledgments

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with references UID/CEC/50021/2013, UID/CEC/50021/2019, UIDB/50021/2020, and PTDC/ART-DAQ/31061/2017, by the PhD grant under contract of FCT with reference SFRH/BD/128628/2017, and by the PhD grant under contract of University of Lisbon (UL), Instituto Superior Técnico (IST) and the research unit Investigação e Inovação em Engenharia Civil para a Sustentabilidade (CERIS).

Table of Contents

I

Introduction

1. INTRODUCTION

Although being designed as a whole, architecture has always been the result of a composition of different elements of varying shapes, functions, and materialities. With the field's tendency to constantly explore the latest technological advances both in terms of architectural representation and fabrication, the process of designing and constructing grew in complexity. Not only is the use of more specific and specialized design approaches required, but the need to independently address the different parts that constitute the architectural whole also increased.

This investigation focuses on one of these parts, the building envelope, due to its relevance in materializing conceptual and creative intents and in improving the quality of the built product. Indeed, as threshold elements between indoor and outdoor, facades have a critical role in mediating the buildings' indoor and outdoor environments. More specifically, this thesis proposes a mathematical framework to support the geometric exploration, evaluation, optimization, and construction of building facades, implemented as the DrAFT 2.0 algorithmic library. This chapter first presents the motivation behind this investigation and the research assumptions and questions that guide it. It then elaborates on the research goals, as well as on the challenges and outcomes of the investigation. Finally, it describes the research methods adopted in this thesis together with its structure organization.

1.1. RESEARCH PROBLEM

Architecture has always used the latest technological advances in terms of design representation, development, and construction. The emergence of Computational Design (CD) approaches in the last decades facilitated the process of searching for design solutions that best meet the existing requirements, and the production of non-conventional shapes by allowing the use of computation in the design process. Algorithmic Design (AD) is one such approach that describes designs through algorithms. AD brings several advantages to architectural practice, such as automating different design procedures and analysis and manufacturing tasks, increasing design freedom, and facilitating design changes, providing the required flexibility to coordinate multiple conceptual, performance, and constructions requirements. Among AD's potential applications, the design of building facades stands out because of its aesthetic, structural, and environmental relevance. AD's

greater design efficiency and flexibility allowed the design of increasingly complex building facades, being the resulting solutions often characterized by their unconventional shapes and intricate geometric patterns.

Despite its multiple advantages, there are still barriers to the widespread adoption of AD, the most evident ones being its technical complexity and high level of abstraction. In addition to requiring programming skills, which most architects do not usually acquire, AD has an algorithmic nature that greatly deviates from the visual and tactile nature of architectural design, which is heavily based on iterative feedback loops where architects evolve their designs. Nevertheless, AD has the potential to solve many of the limitations of current architectural design practice, particularly in responding to the complexity that results from its (1) variability and unpredictability, (2) multiple and context-specific constraints, (3) ever-shorter deadlines, and (4) the need to respond to contemporary social and environmental concerns. Given the amount of time and effort required to develop and evaluate new design solutions using traditional design approaches, only a restricted *design space*, i.e., the set of possible design solutions, is often considered, not only restraining the architects' creative potential, but also compromising the quality of the resulting solutions. These limitations are especially evident in facade design due to the complexity of this architectural element in terms of design conception and production.

So far, most of the solutions proposed to facilitate the use of AD in architecture resort to visual programming, which describes algorithms through interconnected graphical elements containing data, forming dataflow graphs. Visual programming is more popular among designers because of its graphical nature and higher interactivity and intuitiveness when compared to textual programming. Modern Integrated Development Environments (IDE) supporting visual programming for Computer-Aided Design (CAD) and Building Information Modeling (BIM) have increased the adoption of AD by allowing architects to take advantage of it without having almost any programming experience.

However, the features that make visual programming so appealing, namely its simplicity and intuitiveness, are also the ones that hinder its application in larger-scale designs. In these cases, visual programs become very complex and difficult to understand and manipulate. To deal with higher levels of design complexity, programming languages need to provide abstraction mechanisms, but these are usually only available for textual programming. Since this thesis addresses complex design problems requiring higher levels of abstraction, it will focus on the use of text-based AD strategies.

1.2. RESEARCH ASSUMPTIONS

This research assumes that AD is a powerful design approach, opening new opportunities to architectural practice by improving the flexibility, efficiency, and accuracy of design processes. Nevertheless, it also recognizes that using AD is not trivial, especially in a field so heavily based on visual and physical elements,

such as sketches, plans, mock-ups, among others, as architecture. However, as evidenced in the study presented in Part II of this thesis, the use of AD has been growing in recent years, mostly because of the new social, cultural, and environmental needs to which architecture must respond. According to this study, architects are becoming more aware of AD's potential to address the growing complexity of architectural design problems, facilitating performance evaluation and optimization of buildings as well as the coordination of structural and manufacturing constraints. The growing integration of AD in several architectural studios and university curriculums worldwide reflects this. As such, this research assumes that the area demands for methodologies and tools supporting AD will visibly increase in the next few decades and thus it proposes to address this need in the field of facade design. Based on the previous assumptions, this thesis formulates the research questions and goals, which are presented in the following sections.

1.3. RESEARCH QUESTIONS

To make AD more accessible to architects, we need to answer the following research question:

How can we reduce the complexity of AD to address intricate design problems?

Considering that the effort of developing algorithms has been addressed in other fields, such as computer science, through the creation of frameworks facilitating specific applications, this thesis proposes a similar solution for architecture. Given that these frameworks derive from the systematization of a particular domain of application, it is possible to reformulate the previous question, while narrowing its scope to the field of facade design:

How can we systematize AD to address intricate design problems, particularly those of facade design processes?

Given the scope and diversity of the concepts covered, the previous question is decomposed into sub-questions focusing on specific aspects.

How to structure AD according to the specificities of each facade design stage, while smoothing the transition between them?

How to convert conceptual design criteria into algorithmic strategies that respond to different creative intents and design briefs?

How can the provision of a framework reduce the time and effort spent in AD tasks, while handling several design criteria since early design stages?

These questions guide the investigation, not only helping to identify the challenges of the use of AD in different design stages, but also making it easier to understand its implications in the architects' design practice, particularly when addressing facade design problems. They were also useful to assess the validity and relevancy of the thesis contributions.

1.4. Research Goals

As an architect with an AD background, the author is conducting this research to support architects that intend to use AD in their design practices, particularly to solve large- scale or unconventional facade design problems. To that end, this thesis proposes an algorithmic-based methodology addressing the geometric exploration, analysis, optimization, and fabrication of building facades. To generalize its application, the thesis adopts a mathematical approach. Additionally, to facilitate the methodology's use and its continuous extension, it is organized in a modular, categorical structure. This allows the methodology to be independent of specific design methods and tools, as well as capable of responding to the context-specificity and variability of architectural problems.

Through the implementation of the proposal in an algorithmic framework, this work expects to reduce the technical complexity of AD and facilitate its use in the solution of large-scale facade design problems, allowing architects to:

Goal 1. Enhance their creative and critical thinking processes.

Goal 2. Articulate different design constraints, tools, and tasks since early stages.

Goal 3. Improve design space exploration and the chance of finding better solutions.

Goal 4. Materialize their solutions through different manufacturing strategies.

The next section elaborates on the challenges of this investigation, as well as on its potential contributions.

1.5. Challenges and Contributions

AD is based on algorithms, which are abstract means of representation. This poses several challenges to architectural practice, the first being that AD approaches will always be less intuitive than traditional design ones. No matter how much experience an architect has in programming, using AD will always be less intuitive and technically more complex.

The second challenge results from the fact that architectural practice has a strong visual and tactile nature, therefore making the integration of AD far from trivial. AD requires architects to think in a way that largely deviates from the visual nature of the practice.

The third challenge is the need to express tangible building elements and abstract creative intents as algorithmic descriptions. Given the complexity of this conversion, it is often the case that architects spend more time in algorithmic-related tasks than in design exploration.

Although the number of professionals using AD within the architectural community is increasing, the challenges of integrating AD in architectural practice are still substantial. It is therefore critical to address these challenges and provide relevant contributions to the architectural theory and practice, such as:

- Simplifying the development of design solutions.
- Increasing the flexibility and expressiveness of design processes.
- Facilitating the coordination of creative intents with performance criteria since early stages.
- Promoting wider and simultaneously more informed design space explorations.
- Automating the transition from design to manufacturing.

1.6. RESEARCH METHODS

To achieve the enumerated research goals, this thesis proposes a mathematical methodology and framework to support AD processes within the scope of facade design. To that end, it adopts the research approach shown in Figure 1.1, which articulates the research question(s), goals, and steps according to the *Design Inclusive Research* (DIR) methodology [1], integrating *design* as a means of knowledge collection and synthesis [2]. As such, the selection of the research methods in each step considers knowledge from multiple domains and aims to generate design know-how and problem-solving strategies that fit the context of the problem addressed while responding to its needs [1].

Figure 1.1. Conceptual representation of the research approach.

METHODOLOGICAL APPROACH

To successfully respond to the research questions and goals, this investigation adopts the five-stage methodology of Figure 1.2, which encompasses the following research steps:

1. **Data Collection** – This step involves the literature review of several research topics within the scope of building facades, computational design methodologies and technologies, performance evaluation and optimization, and fabrication. It also encompasses the collection of a large corpus of contemporary building facades (named the *design corpus*) with different geometries and materialities. The aim is to understand the existing challenges, as well as to collect relevant design knowledge, particularly in the fields of AD and facade design.
2. **Data Analysis** – This step entails the analysis of the design corpus and the identification of recurrent design problems, common design constraints, and applied design strategies. Based on the patterns found, it then organizes the collected design knowledge by type and role in facade design processes and studies how such patterns can be mathematically expressed. The goal of this step is to establish the strategy adopted by the proposal, as well as to define both its structure and content.
3. **Proposal** – With the goal of simplifying AD in handling larger-scale facade design problems, this step focuses, first, on the conversion of the collected design knowledge into mathematical formalisms targeting facade design processes; second, on the modular organization of the proposed principles into six main categories; and, finally, in the algorithmic implementation of the resulting facade-oriented theoretical framework.
4. **Evaluation** – To assess the expressiveness and applicability of the framework in architectural practice, particularly in the design of building facades, this step imparts, first, the testing of the proposed mathematical principles in reproducing examples from the design corpus; second, their application in a set of novel architectural examples developed in practice-based design scenarios; and, finally, the analysis of the results in terms of implementation accuracy as well as the framework's flexibility and ability to generate a corpus of novel facade designs in real-world scenarios.
5. **Discussion** – The last step encompasses the critical reflection on the proposal's ability to (1) simplify the use of AD within the scope of facade design, (2) reduce the time and effort spent in AD tasks, (3) increase the design freedom and efficiency of facade design processes, and (4) respond to different design problems and workflows, whether AD-based or not. The aim is to identify the merits and limitations of the proposal, as well as potential improvements and extensions that may be relevant to consider in the future.

The first two steps fit into the pre-study phase of the DIR methodology [1], which consists of the collection and critical analysis of knowledge within the research context and the definition of possible strategies to respond to the research problem. The third step frames within the DIR's study phase, which encompasses the formulation of the research proposal. Finally, the last two steps match the post-study phase of DIR [1], which addresses the assessment of the research proposal as well as the generalization of its findings.

Additionally, given the practical-reflective nature of most of the research methods adopted in these phases, the thesis also adopts Practice-based Design Research [1], which involves the use of theoretical principles in practice to learn from it while informing and improving it: for example, the information gathered in the first research step is both theoretical and practical and encompasses specific design processes and contexts; the strategy definition in the second step is mostly reflective and directed towards the abstraction and generalization of practical knowledge; and the assessment of research is mostly based on practical tests and design applications.

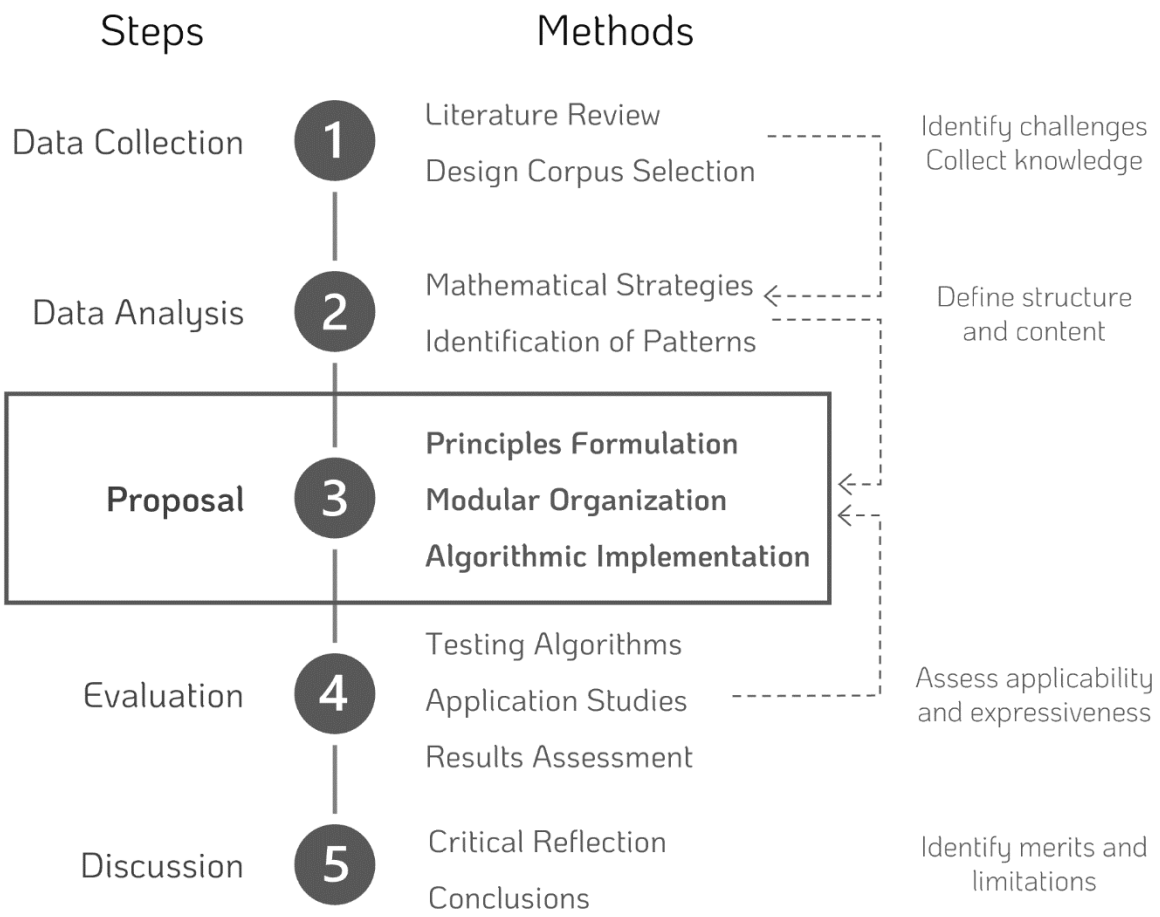


Figure 1.2. Methodological approach: the five main steps of this investigation (on the left) together with the research methods adopted (in the middle) and the reasons for their adoption (on the right).

The next sections further elaborate on the research steps and methods adopted in this thesis together with the reasons for their adoption.

Identifying Design Problems

To gain a better insight into current architectural design practice and identify the existing gaps, this investigation starts with the literature review of the field's last decades, placing particular emphasis on the emerging computation-based design approaches and fabrication technologies and the current environmental and social challenges. As the scope of the thesis is building facades, the background of this architectural element is also studied to understand its historical evolution in terms of function, symbolism, and architectural expression, and a wide range of contemporary building facades is analyzed to identify recurrent design problems and collect relevant design knowledge.

Selecting the Design Corpus

The aim of this thesis is to harvest, organize, and build upon prior design knowledge regarding facade design processes to facilitate the generation of new design knowledge through an AD approach. Therefore, establishing the framework's basic structure requires collecting and analyzing a large corpus of building facades (the design corpus) to identify (1) recurrent design problems and constraints, (2) common design solutions and strategies, and (3) the different design scenarios and practices that currently exist. The collected data is important to enumerate the principles to include in the framework as design knowledge, as well as to guide their future application, therefore generating new design knowledge. The premise is that, although the proposed framework is based on prior design knowledge (the design corpus), it allows reusing such knowledge in numerous ways, naturally leading to novel results [3–5]. Moreover, the collected data is also important to guide the framework's overall structure and the organization of its principles, resulting in a multidimensional space where both the collected design knowledge and principles are classified according to their type and role in facade design processes.

Proposing a Framework

AD is a powerful but complex design approach that is unlikely to achieve the same ease of use and understanding of traditional design approaches. One possible solution to simplify AD is through the use of frameworks, which is already common practice in other research fields, such as computer science [6]. By providing a conceptual structure supporting and guiding specific applications, not only is the time and effort spent in algorithmic-related tasks reduced, but the quality of the results is also improved. Moreover, the provision of frameworks also promotes code reuse, avoiding repeated reinvention and leading to better structured and optimized results. Given the possibility to create frameworks systematizing specific tasks from

various domains, this thesis proposes the use of frameworks in architectural design practice to facilitate algorithmic-based facade design processes.

Defining the Mathematical Approach

Architectural design is highly dependent on the specific circumstances of the design brief, as well as on the way architects approach it. Therefore, it is unlikely that the exact same approach can be used in different projects, whether in an AD or non-AD context. As this investigation addresses the former context, it considers the variability and unpredictability of architectural design processes in a computation-based perspective. Thus, it adopts the formalism and modus-operandi of computational tools, which follow a set of instructions expressed through programming languages. Considering that the most used programming languages are increasingly approximating the language of mathematics, this research considers the latter's formalism in defining the facade design principles and establishing the framework's structure. Using this notation is critical to make its application universal across different design briefs and practices, allowing its principles to be (1) understood and usable by the widest possible audience, (2) independent of the design methods and tools used, and (3) adaptable to the context-specificity and variability of architectural design processes.

Organizing the Modular Structure

Architectural design depends on multiple global and context specific design constraints, which makes its practice a complex and unique process. This thesis focuses on the limitations found during this process, particularly in solving more intricate and larger-scale facade design problems using AD. Given the endless variety of possible design scenarios, it is only natural that the principles provided do not cover all possibilities and thus it is important to structure the framework so that it can be continuously extended with additional design knowledge and categories of design principles as the need arises. Considering the ability of modular programming, i.e., software design techniques that separate the programs' functions into independent, interchangeable pieces, to embrace a wide range of problems by combining finite sets of solutions [7], this thesis adopts a similar strategy, organizing the proposed principles categorically to facilitate their selection and combination. As a result, although the design knowledge available is finite, the framework can respond not only to the most common design problems, but also to more specific ones, while incorporating them in its structure whenever the need arises, becoming thereafter available for future cases.

Implementing the Framework

Regarding the implementation of the framework, two main approaches are considered: a grammar-based implementation, such as (1) shape grammars [8], which constitute sets of shape rules whose step-by-step application originates sets of designs, and (2) a purely algorithmic-based one, such as AD, which encodes

shapes and both combination and transformation rules in a parametric way. While the former is often more intuitive to apply, it is also more difficult to implement and control [9]. The latter, in turn, requires more technical expertise but is more flexible and expressive [10]. As the goal of the framework is to support facade design processes, where architects have full control over the course of the project and its expected results, this thesis adopts the latter approach. The importance of supporting a procedural and deterministic design process is to enable a conscious and informed design space exploration that allows critical thinking and the achievement of useful results. Moreover, the algorithmic implementation approach also allows the automation of repetitive and time-consuming design tasks, as well as the integration of different analysis, optimization, and fabrication routines, which are critical in facade design processes.

To assess the success of the algorithmic implementation and understand the usefulness of the collected design knowledge in the context of AD, the framework is then used in the reproduction of part of the design corpus, proving its ability to reproduce already existing facade designs in a procedural and structured way.

Developing Application Studies

To assess the applicability of the proposed framework in real design scenarios and thus understand its benefits and limitations for architectural design practice, its principles are applied in a set of architectural application studies. Applying the proposed mathematical principles in real-world design scenarios helps build the bridge between the generality and universality of the theoretical framework and the context-specificity of architectural design practice. Given the practical purpose of the framework, this method seems to be the most appropriate to (1) identify the strengths and weaknesses of the framework, (2) acquire and compare practice-based design knowledge, and (3) identify new design knowledge and requirements that are relevant to include in the mathematical methodology.

To ensure the heterogeneity of the evaluated sample, the application studies present varying levels of design complexity and respond to different design intents and constraints. Moreover, while some of them are entirely developed by architects with AD skills, others are the result of collaborations between architects with different levels of AD experience. In the end, the application of a practice-based evaluation method makes it possible to assess the ability of the framework to:

1. Use the collected design knowledge to generate novel facade designs responding to different design briefs.
2. Simplify the associated AD tasks by guiding the reuse and combination of algorithmic strategies.
3. Integrate different design practices involving different creative processes and tools.
4. Extend the freedom and efficiency of facade design processes, increasing the design spaces explored while smoothing the transition between design stages.

Assessing the Research

This thesis assesses the validity and feasibility of its outcome using a qualitative approach [11] because it provides the flexibility needed to adapt to the variable, non-measurable, and innovative nature of the research problem [11], while allowing capturing real-world processes and information that may be relevant to solve it [1, 11]. Therefore, the assessment of the research involves two main stages: the first evaluating the accuracy and the viability of the research methods and the second assessing the merits and limitations of their outcomes.

The first stage comprises the testing of the proposed strategies in small procedural examples to check if they (1) entail the expected results and (2) are able to reproduce some of the design corpus examples. The result is a circular process where the research methods are being iteratively applied, evaluated, and improved according to their results [1].

The second stage involves critical reflection on the feedback received from the application studies, which contributes to the framework's incremental refinement in terms of content and structure, either integrating new methods or adjusting the existing ones to meet new design needs, and to identify the research merits and limitations. The aim is to assess if the proposed generalization of design knowledge can successfully (1) represent real-world design processes and information, (2) solve practice-based design problems, (3) adapt to different design contexts and problems, and finally (4) improve the flexibility and efficiency of design processes.

1.7. Thesis Structure

This thesis is structured into five main parts.

Part I (Introduction) presents the research motivation, questions, and goals, as well as the methods adopted to address them (chapter 1).

Part II (Literature) reviews the existing literature on the topic, organizing it into five main chapters: *Architectural Facade* (chapter 2), *Architecture Meets Computation* (chapter 3), *Sketching Through Algorithms* (chapter 4), *Facing New Design Challenges* (chapter 5), and *Making Digital Real* (chapter 6).

Part III (Research) presents the proposed mathematical principles together with their organization and application in the framework for facade design (chapter 7).

Part IV (Evaluation) describes the application of the algorithmic framework in five application studies responding to different design briefs (chapter 8). Based on the results, it then discusses the merits and limitations of the research, while answering the research questions (chapter 9).

Part V (Conclusion) elaborates on the previous analysis, drawing some final considerations and outlining relevant future research directions (chapter 10).

1.8. TERMINOLOGY

3D Viewer	Computer application specialized in the visualization of 3D objects.
Abstraction	The removal of features to keep only the relevant ones.
AD Workflow	Sequence of design steps and tasks adopted in an AD approach.
Affine Transformation	The mapping of an affine space onto itself, while preserving the spatial relations between its points, straight lines, planes, and sets of parallel lines, and the length ratios of parallel segments.
Algorithmic Analysis	The use of AD to perform design analysis routines.
Algorithmic Design	The use of algorithms to generate designs where there is a correlation between the parts of the algorithm and those of the generated design.
Analysis Process	Evaluation procedure of a design or a design process according to a well-defined performance criterion.
Architectural Design Model	Digital model containing the project's geometric and sometimes construction information.
Attractor	Geometric entity (e.g., point or curve) influencing one or more properties of its surrounding entities.
Computational Design	The use of computation to develop designs.
Design Concept	Creative response to a design brief that integrates the design intent and the reason/directions to the end product.
Design Intent	The set of ideas, goals, concepts, and criteria to consider in the design process.
Design Knowledge	Strategies and solutions resulting from prior design experience.
Design Pattern	Generic description of a solution to a recurrent design problem.
Design Space	The set of possible design solutions.
Design Space Exploration	The process of finding one or more design solutions that best meet the existing requirements.
Design Thinking	Cognitive and practical processes that generate design concepts.

Design Tool	Tool used to develop architectural design solutions.
Digital Design	The use of digital tools in architectural design processes.
Digital Fabrication	Manufacturing processes controlled by computers.
Discretization	Converting a continuous function, sample, or domain into a discrete one.
Empty Function	A function with an empty set as domain.
Expressiveness	Ability to describe an idea or to convey visual and sensorial experiences.
Game Engine	Software application including a rendering engine for 2D and/or 3D graphics with real-time graphical capabilities that is typically used as a framework for computer games.
Generative Design	The use of algorithms to generate designs.
Identity Function	A function that returns its argument.
Indoor Environmental Quality	The set of conditions inside a building (e.g., thermal, lighting, and ergonomics).
Khepri	An AD tool tailored for architectural design.
Metaprogramming	The creation of programs that create or manipulate other programs.
Modeling Tool	Software application that can represent and manipulate 3D shapes.
Modular Programming	Software design techniques where the program's functions are separated into independent interchangeable pieces.
Optimization Process	Finding the value(s) in a domain that yield the best result(s) for a given objective function.
Parametric Design	Describing designs through parameters.
Pattern	General solution to a recurring problem or a repeated arrangement of shapes.
Performance-based Design	The use of performance criteria to guide design processes.
Randomness	The apparent lack of predictability in a sequence of events.
Rationalization	Geometric simplification of a design for manufacturing purposes.
Recursion	Rule or procedure defined in terms of itself.
Traceability	The relationship between parts of the program and those of the model.

1.9. ACRONYMS

AD	Algorithmic Design
ASE	Annual Sun Exposure
BIM	Building Information Modeling
CAD	Computer-aided Design
CAM	Computer-aided Manufacturing
CD	Computational Design
CNC	Computer Numerical Control
DD	Digital Design
DF	Digital Fabrication
DGL	Descriptive Geometric Language
DGP	Daylight Glare Probability
DIR	Design Inclusive Research
DP	Design Pattern
GD	Generative Design
GDL	Geometric Description Language
GUI	Graphical User Interface
HOF	Higher-order Function
IDE	Integrated Development Environment
MF	Matrix of Functions
MP	Matrix of Points
MI	Matrix of Integers
PD	Parametric Design
sDA	Spatial Daylight Autonomy
sUDI	Spatial Useful Daylight Illumination
TI	Textual Input
TPL	Textual Programming Language
VI	Visual Input
VIM	Visual Input Mechanism
VPL	Visual Programming Language

II

Literature

Architecture has always explored the latest technological advances both in terms of design methods and representation and building technology and fabrication. Nowadays, new digital tools and, more importantly, new computational design approaches play a relevant role in the conception, analysis, and production of architecture. These include Algorithmic Design (AD), a design approach that creates designs through algorithms and whose flexibility brings several advantages to architectural design practice, such as (1) automating time-consuming and repetitive design tasks; (2) facilitating design changes; (3) providing higher levels of design freedom; (4) increasing the chances of finding better performing solutions; and (5) facilitating the manufacturing of nonstandard solutions. Nevertheless, and despite its advantages, AD is not yet widespread, mainly due to its complexity and required specialized knowledge, such as having programming experience. The challenge, then, is to make AD more accessible to architects, particularly in solving more complex or unconventional design problems.

This thesis addresses this challenge by investigating the application of AD strategies in the design and manufacturing of building facades. The focus on this architectural element is due to its relevance in terms of building aesthetic, performance, and urban role, and the complexity and exigency of its design, which often requires responding to multiple, and context-specific, design constraints, such as creative intents, local climate, building regulations, performance requirements, among others. It is also motivated by the growing interest in exploring the tectonic potential of unconventional building facades and achieving solutions with high-quality inside spaces and low environmental impact.

This Part II presents the state-of-the-art of the main research topics addressed in this investigation, which are organized in the following chapters:

Architectural Facade, which studies the historical evolution of building facades in terms of role and expression.

Architecture Meets Computation, which presents the technological and scientific background behind computational design approaches and their impact on architecture.

Sketching Through Algorithms, which describes the evolution of AD strategies and tools and their application in architectural practice.

Facing New Design Challenges, which elaborates on contemporary environmental and social concerns and their impact on architectural practice.

Making Digital Real, which analyzes the emerging digital fabrication technologies and strategies and their increasing application in building facades.

2. ARCHITECTURAL FACADE

The architectural facade is the "public face of architecture" [11, p.227] and its design is unique for a particular project and context [13]. This building element is often defined as the outer layer of a building that separates the inside from the outside, providing protection from the weather, while maximizing the indoor environmental quality [13–15]. Architectural facades play an important role in the design of buildings, having several functions:

- Environmental performance [13–19]: mediating the indoor and outdoor by acting as environmental filters and shaping the building's energy performance and indoor environmental quality [20].
- Structural [13, 14, 18, 19, 21]: often integrating structural elements and thus contributing to the buildings' physical stability.
- Safety [14, 15, 22]: providing both protection against external elements and privacy to the users.
- Aesthetic [14, 15, 19, 22]: contributing to the buildings' visual expression and identity.
- Cultural [14, 15, 23–25]: expressing both cultural and social values and thus documenting the architectural and technological evolution.
- Urban role [15, 17, 22, 26–28]: causing feelings and transmitting values to urban dwellers, while engaging and being part of their surroundings.

The way these functions have been addressed throughout time depends on several factors, such as cultural and social context, local weather, and available resources [22]. Therefore, cultural changes and technological developments are among the main contributors to the ever-varying expression and conception of building facades over time [15]. Recently, the use of computational design tools and methods, such as Algorithmic Design (AD), has largely influenced architectural practice, affecting not only the buildings' visual expression, cultural function, and urban communication, but also the way designers engage and shape their environmental and structural functions. Using these methods, a wide variety of design approaches has emerged, being the design of building facades one of their main applications. While some approaches are more performance-oriented, others place particular emphasis on the buildings' visual expression, and others on their integration and communication with

the surroundings [15]. In either case, the result is an increase in the design diversity and complexity of building facades.

2.1. A Brief History of Building Facades

When mankind became sedentary, the need to build permanent shelters arose [22]. Handmade shelters were made from local materials and had as main function weather protection. With time, the exterior walls that separated the inside from the outside of buildings became commonly known as facades. Despite their initial goal being protection, it was not long until this building element gained other functions, such as structural, acting as wall, environmental performance, improving the buildings' indoor environmental quality, and aesthetic and visual communication [14, 15], being either sculped, painted, or covered with wood, mud, straw, or other materials. By acquiring a more expressive and visual role, this element became responsible for conveying the main governing semantics of buildings, communicating "hierarchies of values and claims of power" [28, p.66], while representing their culture and history. The word *facade* itself reflects the visual importance of this architectural element: adopted from the French, *façade* or *facade* derives from the Italian word "facciata" that means face [22]. In fact, it works as the building's face, communicating with the surrounding environment, while covering the inside spatial structure.

From many centuries, the expression of building facades was highly linked to the local culture and available resources, therefore presenting quite different material and geometric characteristics. With the European Maritime Expansion of the 15th century and the resulting intercultural relationships that occurred more and more since then, the diverse geographically separated cultures ended up gradually influencing each other. Gutenberg's invention of the printing press around 1440 also facilitated this interchange, allowing the dissemination of several architectural treatises written at the time [30–32], promoting a wider sharing of knowledge and ideas [33] and, consequently, the widespread of architectural thought.

Throughout this and the following centuries, several social, cultural, and technological changes occurred worldwide, having an impact on architectural practice and, consequently, on the visual expression of building facades. One example is the influence of the Renaissance humanist theories on the architecture of the 15th and 16th centuries, which inspired several architectural treatises, such as Alberti's *De re Aedificatoria* [30], Serlio's *Regole generali d'architettura* [34], and Palldio's *I Quattro libri dell'architettura* [32], while shaping the architectural production and thus the way the design of building facades was addressed. Another example is the Catholic church's reaction to Luther's Reformation movement in the 16th and early 17th centuries, which gave rise to Baroque art and in turn

to highly ornamental and visually complex building facades. The last example is the importance of the scientific and technological developments of the Age of Enlightenment in the Industrial Revolution of the 18th and 19th centuries, causing one of the most pronounced turning points in architecture and thus considerably affecting the visual expression of building facades. It brought new manufacturing methods and materials, such as glass and iron, that allowed for new design opportunities [14], such as the dematerialization of building facades into visually lighter, regular elements that could be free of any structural role [14].

The first half of the 20th century witnessed several architectural movements that addressed ornament in different ways, originating building facades with different geometric configurations and visual expressions that took advantage of new materials, such as concrete and steel. While some of these movements adopted new forms of ornamentation that expressed the cultural and social contexts lived, such as Art Nouveau[1] and Art Deco[2], others began to gradually avoid the use of ornament towards a purely functional architecture devoid of any decoration or historical association, such as Rationalism and International Style. The result was non-ornamented, austere facades [35, 36] that broke with the classical compositions of the past and existing traditions [14].

Reactions to the simplicity, uniformity, and repetition of buildings facades occurred from the mid-sixties onwards, causing drastic and quite distinct changes to the buildings' visual expression. As an example, while Postmodern architecture gives prominence to the building facades, adopting historical design references and nonconventional materials in the search for a highly communicative and iconic architectural expression, High-tech architecture pursues highly technological building facades with no connection to the past, visually exposing the buildings' underlying structure and function while extensively using new building materials and technologies. Another example is the apparently fragmented and discontinuous facades of the Deconstructivism architectural movement, whose unconventional shapes and compositional unpredictability creates visual chaos. Nevertheless, they all shared the same goal of recovering the identity of the architectural facade [14].

More recently, the need to express the prevailing information age and respond to current cultural and social contexts has led to the adoption of new design strategies and technologies, such as *computational design*[3] and *performance-driven design exploration*[4] strategies, which in turn affected, once again, the expression of building facades. Some examples include the digital/media

[1] An architectural style reacting against the eclecticism and historicism of the 19th century that used modern materials and was often inspired by natural elements.

[2] An architectural style reacting against Art Nouveau that explored the compositional potentialities of using simple geometric ornaments, such as squares and triangles.

[3] Design approaches based on the use of computation to develop designs.

[4] Computation-based design strategies where the design exploration process is mostly guided by performance-related principles.

facades [27] made of lights, images, and texts [37, 38] that catch the urban dwellers' attention and trigger different visual experiences, and the parametric [39], responsive [40], and kinetic facades [41] made of either freeform surfaces, complex geometric patterns, intelligent materials, or even responsive devices reacting to human interactions or environmental stimuli.

2.2. The Role of the Facade

Already in Ancient Rome, Vitruvius mentioned three essential qualities of Architecture [42]: *firmitas* (strength), which ensures that buildings stand up; *utilitas* (utility), which targets the effective use of space; and *venustas* (beauty), which provides buildings aesthetic qualities. Throughout time, the relationship between architectural theory and the first two characteristics has been quite peaceful, but the same did not happen with the third, which has been more controversial. This might be explained by the close relation of *beauty* and the concept of *ornamentation*, a quite unstable problematic in architecture [43] that ranged over time from being widely accepted [44–47] to totally rejected [35]. As a result, the role of *beauty* in architectural design has also oscillated, as also did the buildings' visual expression. Nevertheless, the recognition of *beauty* as an essential quality in architecture demonstrates the importance of the architectural facade in terms of building aesthetics and its impact on both human experience and urban image [48–50]. This perspective has been supported by several authors, who have recognized the importance of this architectural element in the buildings' composition due to its protective, aesthetic, and cultural functions [14]. Pedersen, for instance, acknowledged the role of the facade in accommodating the building to its surroundings [51]. Le Corbusier, in turn, addressed the design of the facade in one of his *five points of architecture* [36]. Venturi defended the return to the emphasis on the facade, stating that it allows urbanity and the revaluation of the street [27], and Koolhaas regarded the facade as one of the fifteen key elements of architecture [52].

Despite the important role facades have played in architecture throughout time, the modern movement between the two world wars progressively changed the way this architectural element was addressed, minimizing the use of ornamental elements, which had previously defined its identity. Functionalism defended an architecture based only on the purpose and function of buildings. Together with the massification of construction that came after World War I, Functionalism gave rise to geometrically regular and clean architectural facades through which the buildings' functions were exhibited. The resulting geometric rigidity and monotony of building facades was later criticized by some architects and theorists, such as Frank Lloyd Wright [53], Gordon [54], and Venturi [55], who pointed out the buildings' lack of identity and contextualization, and often resulting lack of ability to

convey emotions and create architectural memory, as well as to relate with both local culture and environment [55, 56].

In the last quarter of the 20th century, the compositional, visual, and symbolic aspects of building facades were again valued in a perspective that expressed the cultural and social needs of contemporary societies. More recently, architects have been addressing these aspects by increasingly resorting to the emerging digital means and computational design approaches, which allow them to explore new facade design possibilities that fit the needs of the new information age by playing with various textures, patterns, materials, and shapes in a highly graphical and expressive way [57].

2.3. The Role of Ornament

The relationship between ornament and architecture goes back a long way, with the former being mainly associated with the latter's *beauty* quality. This section discusses the historical evolution of ornament. To that end, it first presents the definition of ornament within the scope of this research.

The word itself derives from the Latin word *ornare*, which means to adorn. In the literature, the term is often regarded as a synonym of *decoration* [58], but it should not be confused with it. *Pattern* is another term that is frequently associated to ornament, although often expressing a different meaning that is closer to geometry [59]. For McNicholas, *decoration* is the temporary distribution of objects or adorns for embellishment purposes, whereas *ornament* is inherent to objects and provides them with beauty [59]. Similarly, Moussavi and Kubo describe *ornament* as something necessary and inseparable from the object that unintentionally transmits affects and meaning [47]. According to Ghada, *ornament* is a detail, which can be a geometry, color, or even texture, that grabs the attention of the observer [60]. For Mitrache [61] and Miller [62], it is the composition of elements intentionally added to an object to improve its aesthetic attributes and highlight its symbolism.

In this thesis, we regard *ornament* as a part of buildings that has an aesthetic and sensorial function, causing different visual and tactile experiences, as well as a semantic and symbolic function, connecting buildings to their cultural and temporal contexts. *Decoration*, in turn, shares the same aim but achieves it through the addition of temporary elements.

Ornament has long existed in architecture as a means of expression and differentiation of buildings. It resulted from the perception and interaction of the human being with its environment [63] and the search for visual pleasure [60]. Since the beginning of its existence, mankind instinctively added qualitative features to its creations to communicate values or traditions [63], as well as to enhance their aesthetic characteristics [60, 61]. Similarly, we added ornaments to our constructions

to provide them with individuality and meaning, unconsciously creating links between them and the surrounding society and culture [25].

According to Moussavi and Kubo, with cultural evolution comes architectural progress, which often results in new materialities and aesthetic compositions. For the authors, architecture needs to be connected with both its culture and society in order to have meaning and be successfully integrated with its surroundings [47]. As this connection is historically linked to the buildings' visual expression, it has given ornament the ability to convey the historical and cultural values over time, while expressing the symbolic and aesthetic meaning of, for instance, some constructive elements [43, 60].

As communication is one of the main roles of ornament [47], its application has, since ancient times, primarily targeted the most visually exposed building elements. The architectural facade is one of such elements, which explains its tight relationship with ornament. As mentioned in the previous section, this relationship has, however, changed throughout time, mostly influenced by the way ornament was used in architecture at that time.

Ornament in architecture has evolved from simple carving or painting decorations to the differentiation of buildings according to their function or cultural context or even to an expression of power [63]. This evolution is therefore reflected in the expression of buildings' facades along history, which ranges from simple reliefs, to colored tiles, or elaborated frescos. As an example, in ancient Egyptian architecture, ornament was mostly used to express religious, political, and social principles [15], whereas in Greco-Roman architecture it pursued beauty and harmony through the use of both geometric proportion and plant-based elements. Islamic architecture used geometric, calligraphic, and plant-based elements in the pursuit of symmetry and visual complexity, while conveying meaning and hiding the buildings' inner functions and spaces [15]. Finally, while the International Style minimized the use of ornament in the search for transparent, smooth surfaces that clearly reflected the buildings' structure and functions, High Tech architecture visually exposed the buildings' structural and technological systems, making them compositional elements of facades [15].

The role of ornament has long caused debate in architectural theory and practice [62]. Literature on this topic is vast and goes back to the 1st century BC. In the Roman period, Vitruvius was one of the first authors to address ornament in his writings, although in an indirect way. For him, ornament aims at pursuing visual pleasure and beauty, i.e., to achieve the *venustas* quality, making it an integral element of architecture that cannot be separated from it [42, 59]. Much later, in the 15th century, Leon Battista Alberti theorized the concept of ornament in a similar way, i.e., as an element

that enhances beauty[5], which for the author was one of the most important requirements in architecture [30]. A century later, Wendel Dietterlin stated that ornament is more than an element attached to a surface; it is an architectural element itself that has evolved from the classical times [64]. In the 17th century, Christofer Wren defended that the relationship between architecture and ornament had a social and psychological context in which buildings should be regarded as individual ornaments [63]. Already in the 18th century, James Gibbs argued that architectural beauty depends on proportion and not on the richness of its materials and details, and thus, it can be reached through plain, unornamented surfaces [63, 65].

The following two centuries were marked by controversial theories on the relationship between ornament and architecture, from which two main and opposite perspectives stood out: one defending that ornament is part of architectural expression and the other stating it is superfluous and thus should be avoided [43].

In the 19th century we find authors like Alexander Heideloff, who claimed that ornament is inseparable from architecture but it must pursue harmony and proportion and it must not be applied randomly [63]; Augustus Pugin and George Aitchison, who defended an unornamented architecture because, according to them, architectural beauty lies on a purely functional design where ornament plays a secondary role; Gottfired Semper, who stated that ornament is an integral part of architectural practice, coordinating both functional and structural requirements of a building [25, 66]; and John Ruskin and Owen Jones, who regarded ornament as the essence of architecture and the reflection of its beauty and spiritual values. Similarly, Louis Sullivan believed ornament played an important role in providing buildings with liveliness and identity [63, 67], growing from the materials' organization and organicity and thus being inseparable from buildings [47].

In the early 20th century we have, on the one hand, Hans Poelzig's perspective, which defended an architecture that was not controlled by nor hidden beneath ornament, and, on the other hand, Frank Lloyd Wright's ideas, which regarded ornament as part of buildings and their structure [53]. A more radical point of view was adopted by Adolf Loos, who pursued an architecture without ornament [35]. This idea of transparency as representative of space, structure, and function greatly influenced well known architects of the first half of the century, such as Naum Gabo, Antonie Pevsner, Mies van der Rohe, and Le Corbusier, who adopted the anti-ornament movement [25, 63].

The second half of the 20th century was marked by the return to ornament [63]. From this era, the contrasting perspective of Robert Venturi stands out, which radically replaced the Modernism

[5] According to the author [30], beauty is the perfect balance between all parts, where nothing can be added, removed, or changed without making it worse.

austerity with ornament in a highly expressive form of communicating architecture that separated the building's function from its representation [41, 49]. More recently, ornament has been theorized as a building element that can have multiple roles, namely functional, aesthetic, tectonic, environmental performance, or even performative, i.e., dynamically interacting with urban dwellers by either moving or changing its physical characteristics [62].

Nowadays, architects are addressing ornament in new ways that express the prevailing social and cultural context [57], which is dominated by technology, information, and real-time communication [60], while benefiting from the latest technological developments in the design, manufacturing, and assembly of new forms of ornamentation [57]. In an era where visual communication is the governing factor [60], these new technologies have facilitated the development of novel surface textures, geometries, and patterns that catch the urban dwellers' attention [57], as well as the inclusion of digital- or computation-based ornamental elements, such as digital screens or dynamic devices, that interact with external stimuli [43]. While in some cases the main concern of ornament is to express the buildings' functions, in others it is to integrate the urban space in a more scenographic perspective, originating building facades with an autonomous function.

In sum, ornament remains a controverse topic in architectural practice and theory, particularly in terms of role, semantics, and aesthetic standards. It is nevertheless expected that it continues to evolve in new directions, reflecting the social and cultural needs of future societies and integrating the new technological developments emerging from them [63].

2.4. Contemporary Building Facade Design

With the technological evolution of the last decades, the increase in design constraints, such as tight deadlines, environmental concerns, and economic restraints [68], along with the architects' ever-present ambition to go beyond conventional shapes, the process of designing building facades grew in complexity [69]. Not only does it have to deal with multiple and often context-specific design requirements and different information sources simultaneously, such as structural information, manufacturing guidelines, and clients preferences, but it also has to meet the growing need for better solutions in terms of environmental performance [70, 71]. Combined with the variability and diversity of architectural design problems [13], the result is a laborious, time-consuming, and error-prone design process [68] that requires architects to use multiple design tools that often do not directly interoperate with each other [72].

With the aim of minimizing the complexity of contemporary facade design processes, some authors focused on structuring the intricacy of building facades, classifying them according to

different criteria. Moussavi & Kubo [47], for instance, categorized building facades according to their material expression (or ornament) and how this expression relates to culture by creating sensory and emotional experiences, which the authors name as *architectural affects*. Three main classifications were suggested, the first one being *Depth*, which categorizes buildings according to the way ornament is integrated in them in four subcategories, namely *Form*, *Structure*, *Screen*, and *Surface*; the second *Material*, which addresses the way material forces influence the structuring of ornament; and the third one *Affect*, which is the result of the interaction of the previous two.

Another example is the classification proposed by Otani & Kishimoto [73] for fluctuating facades, i.e. facades whose elements, such as windows, panels and ornaments, differ from conventional ones in terms of size, shape, and frequency of use. After identifying the most frequently changed facade elements and grouping them by similar types, the authors proposed a classification composed of 33 categories, such as window shape and size, panel shape, size, and both vertical and horizontal repetition, etc.

Pell [45] categorized facades according to (1) their means of production and type of surface distribution, dividing them into four main groups, namely *Applied*, when the facade is treated as a neutral plane on which different material compositions are applied, *Perforated/Cut*, when its surface is submitted to scoring, cutting, or perforation processes, *Layered*, when it results from material superpositions, and *Stacked/Tiled*, when it is the result of a composition of multiple individual elements creating a whole; and (2) how they articulate their material and content, organizing them in terms of *integration*, *materialization*, *contradiction*, and *disinterest*.

Wassef & El-Mowafy [74] organized responsive kinetic facades, i.e., facades composed by elements that either move or change their physical characteristics to respond to environmental stimuli, in two groups: *Configuration*, which classifies facades according to the type of *geometric transformation* creating the kinetic movement and the *pattern shape*, *surface form*, and *material* of the facade; and *Function*, which categorizes facades into three main functions, namely *aesthetic*, *energy generation*, and *environmental control*.

The last example is Velasco et al.'s [75] classification for dynamic facades, i.e., facades whose physical elements move, based on two main criteria. The first one is *Movement*, which categorizes dynamic actuators according to the type of movement allowed in (1) *mechanism-based*, if the movement is based on *rotation*, *translation*, or *hybrid*, and (2) *material deformation*, if it results from environmental stimuli (*self-changing*) or artificially controlled forces (*direct external input*). The second criterium is *Control*, which can be *local*, when the actuator is autonomous, or *central*, when it is linked to a single control system.

Still, despite helping architects understand the current variety and complexity of building facades, most of these works do not help in the modeling of new solutions, particularly if they are modeled algorithmically. Inspired by works confirming that sets of algorithms can be reused in the exploration of new designs [7, 76–78], Caetano et al. [79] presented a classification addressing the algorithmic development of facade design patterns, providing sets of ready-to-use algorithms organized according to different modeling needs and design configurations. Nevertheless, the proposal is restricted to initial design stages, focusing mostly on the geometric exploration of different facade design solutions, not considering other relevant design stages such as analysis, optimization, and fabrication.

3. ARCHITECTURE MEETS COMPUTATION

Architecture has always embraced innovative ideas, materials, and techniques and contemporary architecture is no exception. It has been exploring the latest technological advancements, namely the new computational means of conception and production that offer new possibilities for architectural design and manufacturing. The emerging computation-based design approaches differ significantly from the previous design approaches since, instead of grounding architectural design representation in its geometric aspects, they base it on its computational logic. These have been addressed by several authors under the name of Computational Design (CD) and applied in several design studios to increase both the creative exploration and feasibility of their projects. CD is causing considerable changes in both design theory and practice, but most architects are not yet aware of its potential.

This chapter traces the course of CD from its origin until today and clarifies some of its related terms and design perspectives. It starts by explaining the evolution of CD since the early 60s and then discusses the advancements in CD tools, while presenting scientific events that addressed it. It continues with a chronology of the literature on CD and, finally, a clarification of the most relevant CD-related terms. It concludes that, similarly to past architectural experience, technological developments continue to shape both architectural theory and practice and are simultaneously guided by their needs and aspirations.

3.1. THE EVOLUTION OF COMPUTATIONAL DESIGN

It was the combination of programmable computers, CD, and numerically controlled technology that gave rise to the development of the well-known Computer-Aided Design (CAD), Building Information Modeling (BIM), and Computer-Aided Manufacturing (CAM) tools, among others. The use of these tools has been causing changes in architectural practice since the 60s due to increasing the ability to handle complex geometries with greater precision and efficiency [80].

Ivan Sutherland disseminated the concept of CAD in 1963, with the creation of Sketchpad. This innovative system was considered by many authors [81–84] as the first parametric tool for architectural design, proving that engineers and designers could communicate with a computer and use it as a medium for thinking and making [85]. For Kalay [86], Sketchpad marked an important turning point in architectural practice, allowing the replacement of hand-based design tasks with digital-based ones. Until then, architects designed by hand directly on paper and, in the event of a

mistake, part (or the entirety) of the design had to be erased and redone. Designing on a computer made it easier for architects to erase and redraw parts of a design and obtain more precise results more efficiently. In the next decades, Sutherland's ideas [87] were further explored and adopted by other authors, such as Eastman [88], Leler [89], and Gross [90].

The biggest shift in the field, however, happened in the early 80s with the commercialization of CAD tools for architectural design, such as AutoCAD and MicroStation. These tools massified the conversion of the paper-based design process into one based on the computer, which brought several advantages to architectural practice, such as greater design precision, easier correction of design errors, and the ability to reuse parts of a design. In the late 90s, digital design tools were fully integrated in architectural practice, resulting in a 'computational turn' in which architects already used computational processes as a means of design exploration [91].

3.2. COMPUTATIONAL DESIGN TOOLS

In the 21st century, the use of digital technologies was already part of architectural practice. For Rocker [91], CD processes have become a means for design exploration, extending the capability of traditional processes, while challenging and, therefore, changing the design conventions and praxis. In this scenario, the development of digital tools for architectural design, namely CAD, BIM, simulation, and analysis tools, played an important role.

Regarding CAD tools, in 1982 Autodesk released AutoCAD, a 2D digital drafting tool suitable for architecture, project management, and engineering. In 1985, it was extended to integrate 3D modeling. In the same year, BentleySystems launched MicroStation, an application with a limited interface supporting only basic 2D drawings; 3D modeling was incorporated a decade later. Another important development occurred in 1987, with Pro/ENGINEER, a tool developed by Samuel Geisberg for mechanical engineering that allowed users to create 3D parametric components, reducing the cost of design changes, while overcoming the lack of flexibility of 3D modeling at the time [92]. In 1998, Robert McNeel & Associates launched Rhinoceros 3D, a commercial 3D CAD tool based on NURBS (non-uniform rational basis spline) [93] that targeted the generation of mathematically precise representations of curves and freeform surfaces. In 2000, @Last Software developed the 3D modeling software SketchUp, an easy-to-use tool that gave architects more design freedom. In 2006, Google acquired the company and extended the tool under the name of Google Sketchup. In 2012, Trimble Navigation (currently known as Trimble Inc.) purchased the tool and continued its development.

Regarding BIM tools, in 1982 Graphisoft started developing ArchiCAD, making it available to architects in 1987. This tool produced 3D models whose elements were parametrically connected and

embedded with their corresponding construction information. In the mid-90s, Gehry Technologies adapted the aeronautics design software CATIA, which they used for many years for architectural design [94], originating Digital Project. In 2000, Revit Technology Corporation released Revit, a tool that supports the design and documentation of buildings by creating parametric models that contain both geometry and construction information. In 2002, Autodesk purchased the company and continued to develop the tool, extending it with Revit Structure in 2005, Revit MEP in 2006, and the visual programming tool Dynamo in 2011. Further BIM tools used in architecture include BentleySystems' AECOsim and Tekla Corporation's Tekla Structures.

Alongside CAD and BIM tools, the use of analysis tools, to model the behavior of buildings and evaluate their performance, and optimization tools, to search for the best solutions, has also been noticeable in architecture. Examples of commonly used tools include EnergyPlus and Green Building (both for energy simulation), Robot (for structural analysis), Radiance (for lighting simulation), ClimateStudio (for thermal, daylighting, and energy analysis), and Galapagos (for optimization).

Finally, Integrated Development Environments (IDE), i.e., programming interfaces to help the development and debugging of algorithmic programs, to automate design tasks and extend the tools' modeling capabilities were also proposed early on. An early example is Autodesk's AutoLISP released in 1986. Later examples include Bentley's Generative Components launched in 2007, and the visual programming tool Grasshopper added to Rhinoceros 3D in 2008, which became very popular among architects due to its ease of use and ability to create complex parametric models.

3.3. Computational Design Methods

According to Aish and Bredella [95], CD evolution involves the progression from 2D drawing to 3D BIM and, then, to design computation. After several years of effort developing digital design tools targeting non-programmers, the design field increasingly felt the need to integrate programming capabilities into such tools to satisfy the need for design automation and parametrization, which was only possible by allowing the tools to support user-programmable features.

Computational tools and methods evolved through different generations, which reflected their capabilities and the way these were used by architects. According to Dorst and Dijkhuis [96], the first generation started in the early 1960s and fostered more detailed descriptions of the design activity, while regarding design as a rational problem solving process. The search for a design process based on logic and mathematical principles resulted in CD methods and tools with several shortcomings for architectural practice, such as a deterministic and linear design approach and the lack of a

Graphical User Interface (GUI) [97]. These drawbacks, along with the steep learning curve and large cost of these methods and tools, demotivated architects from using them.

With the spread of personal computers and the improvements in GUIs, CAD software became accessible to a larger architectural community. The resulting association between computation-based processes and design processes rapidly matured as a design medium and, for the first time, most computer users were non-programmers [98]. According to Asanowicz [99], this scenario lead to a second generation, which, for Reffat [97] was marked by the improvement of user-computer communication: the available software already allowed designers to draw directly on computer screens without having programming knowledge. Some authors named this generation as *2D Drafting Era* [95], *Electronic Drawing Board Era*, or even *first generation in the architectural offices* [100], characterizing it as the use of advanced technology to emulate traditional design processes. In practice, architects replaced traditional drawing tools with more efficient and precise ones but they often did not take advantage of their computational power [101], resulting in a scenario where CAD tools and the idea of *Computer Aided Drafting* were closely related [99, 102]. Nevertheless, Reffat [97] recognized that, even so, the use of computational approaches was advantageous for architectural design, facilitating the exploration and documentation of more complex solutions.

The 21st century brought new advancements in 3D modeling design tools, improving the architects' creative exploration, visualization, and documentation processes. This evolution reached two important periods: first, the BIM era, in which architects created and extracted technical drawings and construction information from 3D models but still resorted to craft-based or mechanically-assisted construction processes, and then, the algorithmic and generative design era, in which architects resorted to programming strategies to overcome the limitations of their design tools and smooth the transition between design representation and construction [95]. According to Achten, the use of computers has caused several changes in the architects' design workflow, going "beyond the first round of imitating and supporting traditional practices" [99, p.507]. For Leach [103], the role of the architect has evolved from the simple 'form-giver' to the controller of generative processes, i.e., processes based on a few a rules that can generate several unpredictable results, originating design solutions that result from the combination of human creative potential with the computers' generative capabilities.

3.4. Computational Design Theory

The scientific events of the last decades stimulated important debates on theoretical and practical issues, such as the architectural practice situation, the impact of new tools and methods, and the

practical application of scientific research, as well as social concerns, such as the increasing environmental problems and emerging living needs. These, in turn, influenced the application of CD in architecture in different ways, which is further elaborated in this section.

3.4.1. Scientific Events

Considered the field's embryonic conference, the *Conference on Design Methods* was held in 1962 to discuss topics like (1) designing better by understanding the design process, (2) externalizing the design process to allow collaborative work from early to later stages, and (3) using the computer to automate design tasks [104].

A decade later, in 1972, the *1st International Congress on Performance* introduced a new design perspective that resulted from the interest of computer scientists in both systematic design methods and design science and the use of such principles to evaluate building performance as a means to scientifically justify design decisions.

The next decade witnessed a large increase in the number of international conferences, among which stand out the north-American *Association for Computer-Aided Design in Architecture* (ACADIA) annual conference, founded in 1981 by Mitchell, Eastman, and Yessios with the aim of discussing the role of computation in architecture and encouraging innovation in the architectural design practice [104]; the annual conference *Education and Research in Computer Aided Architectural Design in Europe* (eCAADe), held for the first time in 1983, introducing education as a research topic; the biannual *Computer-Aided Architectural Design Futures* (CAADFutures) conference, founded in 1985 by Tom Maver, Rik Schijf, and Harry Wagter with the aim of fomenting CAD advancements that envisioned the quality of the built environment; the biannual conference *Artificial Intelligence in Design* (*Design Computing and Cognition* since 2004), settled in 1985 to focus on the use of artificial intelligence techniques in design; the biannual *International IBPSA Building Simulation* conference, also established in 1985 with the aim of improving the design, construction, operation, and maintenance of buildings; and, lastly, the *International Conference on Computational and Cognitive Models of Creative Design*, held in 1989 to explore the advancement of designers' understanding of computational and cognitive models of creative design.

The 90s brought conferences like the *Association for Computer-Aided Architectural Design Research in Asia* (CAADRIA), founded in 1996 to promote teaching and research in CAD in Asia, and the *Sociedad Iberoamericana de Gráfica Digital* (SIGraDi), settled in 1997 with the aim of debating the application and potentialities of the new digital technologies.

Already in the 21st century, the *Arab Society for Computer Aided Architectural Design* (ASCAAD) conference was established in 2001 and, two years later, the *Smart Geometry Conference*, targeting

the integration of CD in architecture [105], and the *Performative Architecture Symposium*, studying the gap between design and analysis and the influence of performance in architectural design, were held. The *Advances in Architectural Geometry* (AAG) conference was organized for the first time in 2008 to study the emerging geometric developments in architectural design and engineering and, in 2009, the *Digital Architecture London Conference* was held to discuss the role of technology in society. Lastly, the *Symposium on Simulation for Architecture and Urban Design* (SimAUD) was settled in 2010 with the aim of establishing a collaborative simulation workflow supporting sustainability.

3.4.2. Scientific Production

In addition to the scientific events, scientific journals also contributed to the dissemination of CD approaches in architecture. Some of them already existed but started to embrace the subject of CD in architecture, such as *Design Studies* (1979) and *Architectural Design* (AD), whereas others were created specifically to address it, namely *Automation in Construction* (1992), *Journal of Architectural Engineering* (1995), *Nexus Network Journal* (1999), *Construction Innovation (2001), International Journal of Architectural Computing* (2003), *Journal of Building Performance Simulation* (2008), *Building Simulation* (2008), and *Frontiers of Architectural Research* (2012). Finally, journals with a high impact factor on building science and technology fields often incorporate articles exploring CD techniques, especially those on building simulation, e.g., *Solar Energy* (1957), *LEUKOS: The Journal of the Illuminating Engineering Society* (1972), *Building and Environment* (1976), and *Energy and Buildings* (1977). These are temporally organized in Figure 3.1 together with the international conferences.

The increasing number of scientific events and journals on CD therefore reflects its growing importance in the field of architecture. To evaluate its evolution from a theoretical point of view, the timeline in Figure 3.2 organizes the literature on CD into three generations of thought (colored in different grey tones) and presents important technological and construction milestones, which are further discussed in chapters 4 and 5.

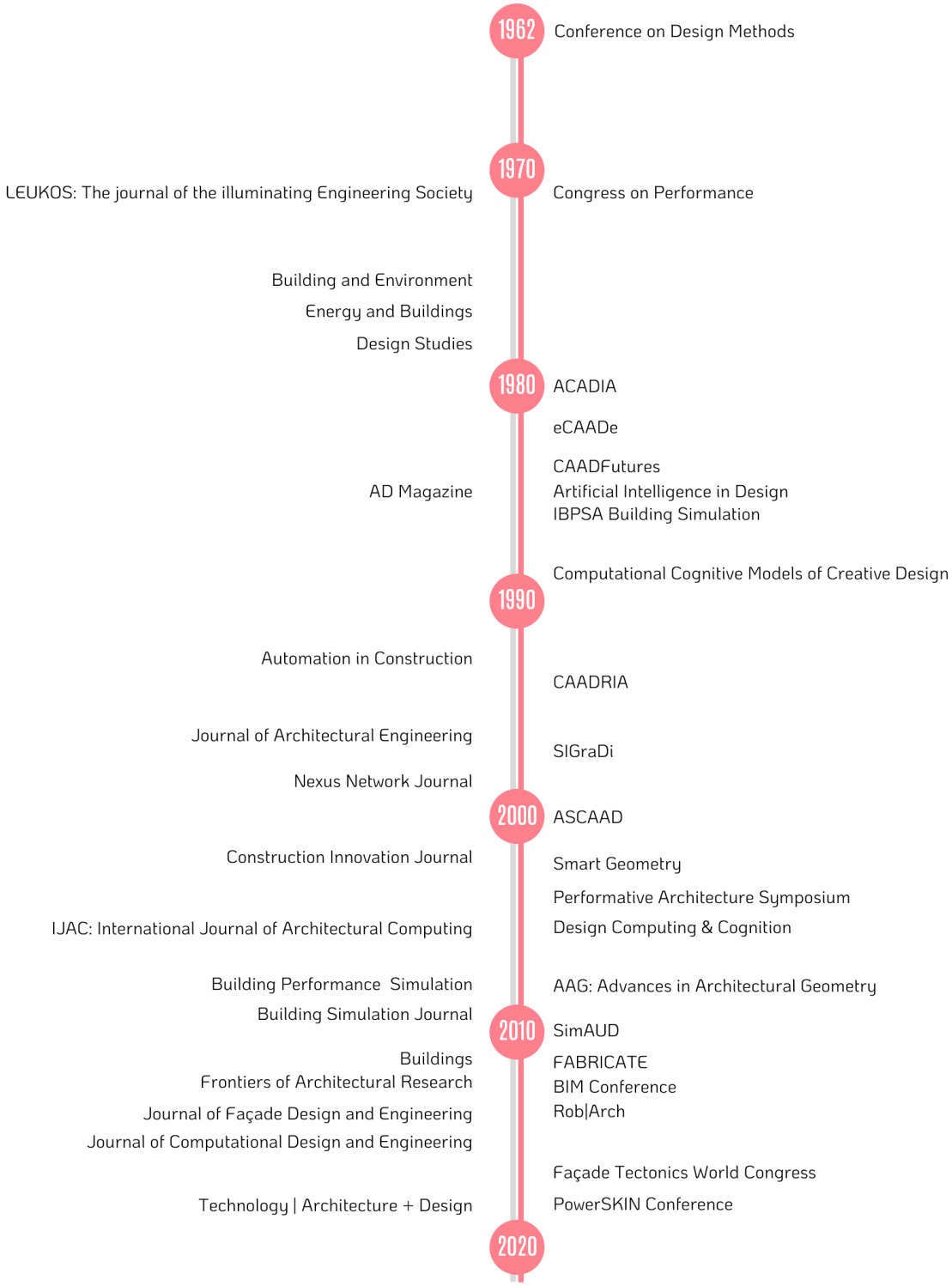


Figure 3.1. Timeline of CD-related scientific journals (left) and conferences (right).

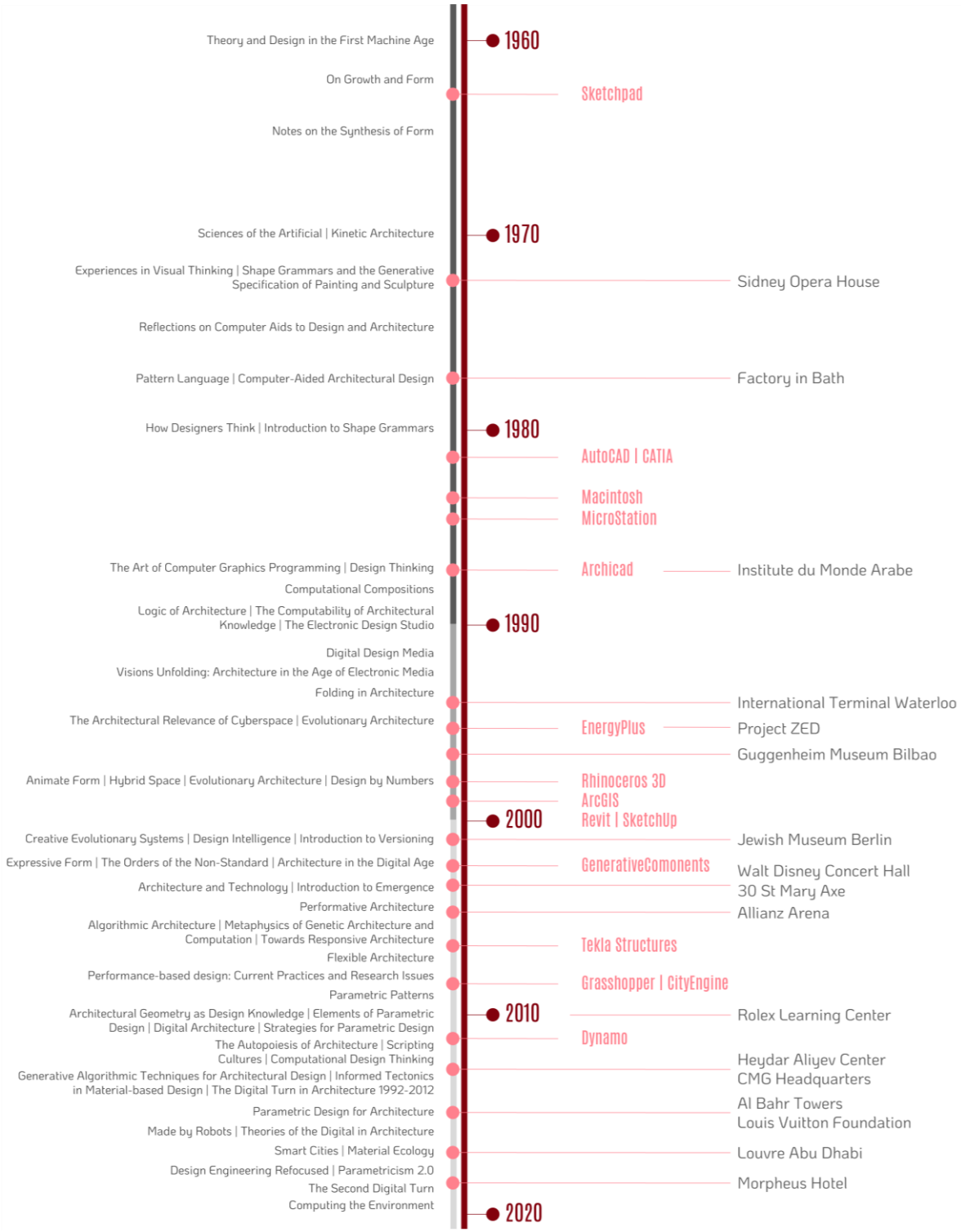


Figure 3.2. Timeline on CD literature (on the left) and innovative tools and iconic buildings (on the right).

The first generation embraces literature from the 60s to the early 90s. The early integration of computation-based methods in architecture has since long caused transformations in design theory. According to Koutamanis [106], the first steps happened in the 60s and were inspired not only by

modernism thinking, but also by the technological explorations at the time and by theories from other scientific fields, namely artificial intelligence and mathematics. The resulting literature regarded architectural design as a *thinking before acting* activity [107] that handled design problems in a rational perspective. Important works from this decade include Banham's *Theory and Design in the first Machine Age* [108], Alexander's writings on design processes [109], Negroponte's *Towards a Humanism Through Machines* [110], and Sutherland's ideas on design variation methods, design constraints, and parametric instances [111, 112].

The number of scientific publications and the popularity of some generative systems, such as space allocation techniques [113] and shape grammars [8], increased in the following decades. The 70s are marked by works like *Computer-Aided Architectural Design* [114] and *A Pattern Language* [4], as well as by the first Ph.D. theses on CD [115, 116] and by overviews on CD expectations at the time [114, 117, 118]. The 80s are characterized by works like *Introduction to Shape Grammars* [8], *How Designers Think* [119], and *Computational Compositions* [120]. Finally, the early 90s, a period marked by an increase in the popularity of computers due to the design efficiency improvements they provided, witnesses an explosion of conferences, journals, and theoretical works on CD, from which the following stand out: *The Electronic Design Studio* [121], *Logic of Architecture* [122], *Digital Design Media* [123], and *Visions Unfolding* [124].

The second generation includes literature produced from 1993 to 2000. The works from this period are characterized by (1) a discursive integration of philosophy and mathematics, (2) an attempt to define the emerging architectural production, and (3) a concern in stimulating the correct application of CD methods in architectural practice. It starts in 1993 with Lynn's *Folding in Architecture* [125] and his proposal for a new design paradigm based on "smooth transformations" that aim at replacing Post-Modernism and Deconstructivism in a visual and mathematical perspective. Several works on CD followed, among which stand out Frazer's *Evolutionary Architecture* [126], which extends the *Anticipatory Architecture* of Cedric Price [127] and advocates for an architecture acting as a living evolving system; Lynn's *Animate Form* [128], which proposes using animation software to enhance form generation processes; and Kolarevic's *Architecture in the Digital Age* [129], which explores the impact of CD in both the architecture and the construction fields and presents some of its related terms, including performance-based design and morphogenetic design.

The third generation embraces literature from the 21st century and witnesses the most accentuated paradigm shift of the three. According to Terzidis, while the formulation of new design theories in the past resulted from the understanding and reinterpretation of prior concepts, the emerging design paradigms at the time had no precedent [130]. The evolution of design tools and the increasing concern with environmental and social problems are probably among the driving

forces that motivated the perspective of design as research [131]. In this perspective, design becomes a medium for knowledge production that follows a more scientific, computation-based direction where human intuition is the starting point of design exploration and CD is the provider of the means to augment it, potentiating human creativity instead of replacing it [130]. According to Oxman [132], design processes are increasingly embracing techniques like scripting, optimization, and digital fabrication, resulting in new related terms, such as Parametric Design [83], Generative Design [133], Performative Design [134], Performance-based Design [135], and Biomimetic Design [131]. This variety of terms reflects the growing ramification of CD literature during this period into multiple perspectives, which are described in the following paragraphs.

The first perspective is the idea that intelligence can inform and guide the design process and, for some authors, even replace theory as a guiding architectural principle [136]. According to Speaks, even though in the past theory changed architectural practice, currently it "no longer has any consequences for the practice of architecture" [136, p. 209]. Authors embracing this perspective include, for instance, Oxman [138], Picon [139], Carpo [140], and Oxman and Oxman [141].

Another perspective is the use of performance to inform and guide design exploration processes. Despite the fact that the notion of *performance-based design* had already gained some visibility in the 40s and 50s with the *performative turn* movement [142], its popularity visibly increased in the early 2000s. Authors like Kolarevic [143], Tschumi [144], Kronenburg [145], Oxman [135], Leatherbarrow [146], and Picon [147] address the use of simulation, analysis, optimization, form-finding, and evolutionary methods in architectural practice to investigate how the environmental context can inform the design process.

A third perspective is the idea of *natural morphogenesis*, which involves the use of biological principles behind the development of organisms to guide design processes [112]. This perspective was inspired on the term *morphology* introduced by Goethe [148] when studying form-guiding methods inspired by natural processes, which distinguished *form* from *formation*; an idea later extended by Thompson [149] when investigating the geometric rules behind organic structures and transformations. Currently, this idea has inspired new design perspectives like *morphogenetic design*, *evolutionary design*, and *biomimetics*, as well as several authors, including Migayrou [150], Hensel, Menges, and Weinstock [151], Menges [152], and Oxman and Oxman [131].

The adoption of methods from the computer science field was proposed by Terzidis in his *Algorithmic Architecture* [101], given the advantages the use of algorithmic techniques brought to architectural design in automating tedious tasks and exploring generative processes, among others. This perspective was then adopted by several other authors, such as Burry [102], Woodbury [83], Jabi [153], and Schumacher [154, 155].

The study of the relationship between architectural forms and *tectonics*, i.e., its structural and material properties, also gained significance in the field. Examples of authors addressing this design perspective include Picon [156], Scheurer [157], Bechthold [158], Oxman [159], and Gramazio & Kohler [160].

The use of CD methods to address the growing sophistication of fabrication technologies is another design perspective recently explored by authors like Iwamoto [161], Willmann et al. [162], Oxman [163, 164], and Oxman et al. [165], as well as by different design studios, such as Design to Production, Gehry Systems, and Zaha Hadid Architects.

The quest to create designs that interact with both users and the environment, either by using responsive materials or parametrically controlled mechanisms, inspired design perspectives known as *responsive*, *interactive*, and *dynamic design*. This idea was already addressed in the past by Chareau and Bijvoet (Maison de Verre, 1932); Fuller (Dymaxion houses, 1930 and 1945); Archigram's utopian projects (1964); Rogers and Piano (Centre Pompidou, 1977); Nouvel (Institute du Monde Arabe, 1988); and Toyo Ito (Tower of Wind, 1991). Recent examples of authors include Beesley, Hirosue, and Ruxton [166] and Oosterhuis [167].

The last perspective defends the study of mathematics, geometry, and computer science to create design knowledge [168]. The increasingly complex requirements of current architectural design and construction have motivated the growing use of mathematical strategies, such as rationalization, to adjust geometrically complex designs towards feasible solutions – an idea addressed by, for instance, Andrade, Harada, and Shimada [169], Eigensatz et al. [170, 171], Flöry and Pottmann [172], Fu and Cohen-or [173], and Son et al. [174]; and design patterns, to promote the reuse of known design strategies or solutions in solving design problems [5] – an idea explored by Woodbury, Aish, and Kilian [76], Qian [7], Woodbury [83], Hudson [77], Larson [175], Chien, Su, and Huang [176], Yu and Gero [177], and Su and Chien [78], among others.

The interdependency between CD and technology is expected to continue in the future. As such, it is likely that current technological advances will cause new developments in architecture and, at least, force architects to adapt to new design techniques and processes, as it happened with the introduction of CAD in the 80s [178]. Robotics, for instance, deeply changed the automotive and aerospace industries and the same will probably happen in architecture when similar technology becomes more accessible and widespread in the field [160]. Another emerging field is machine learning, whose impact is increasing in different fields and activities and it is predictable that it will also affect architectural practice [179, 180].

3.5. COMPUTATIONAL DESIGN TERMINOLOGY

The dissemination of CD in architecture challenged traditional design processes, which were heavily based on manual drafting tasks, causing a turn in architectural design [91]. The increasing popularity of CD in the last decades is mostly due to the improvements it brough to the architects' design practice, allowing them to explore different design strategies and research paths. This scenario promoted the emergence of new design paradigms and, consequently, new related terms. Due to their newness and scope overlap, some of these terms have received ambiguous definitions that often embrace two or more conflicting ideas or overlap with other terms. Algorithmic Design (AD), Generative Design (GD), and Parametric Design (PD) are popular examples of still ill-defined terms that are widely used throughout this investigation. To avoid misinterpretations of the terms, this thesis adopts the taxonomy proposed by Caetano et al. [181], which results from the analysis of the use of some relevant CD-related terms in the literature, namely PD, GD, and AD. The next sections briefly describe the theoretical evolution of these terms, as well as the adopted definitions.

3.5.1. COMPUTATIONAL DESIGN

In the literature, CD is often regarded as an approach based on the use of digital tools to develop design solutions [182–184], which is similar to Digital Design (DD). However, for some authors, CD is different from DD because it requires taking advantage of the computers' computational capabilities in the act of designing [101, 132, 185–189]. This thesis adopts the second perspective, which distinguishes the design processes that use computers only for drafting or other representational purposes from those that use algorithmic or computational-based procedures.

This thesis considers DD as the use of computational tools in the design process and CD as the use of computation to develop designs. This perspective makes these two terms orthogonal, which means that it is possible to use CD without taking advantage of DD, e.g., Frei Otto's minimal surfaces experiments based on analogue computation [190], use DD without following a CD approach, e.g., using a CAD tool as a drafting device and not explicitly using computation, or using both, e.g., Mark Burry's work in Sagrada Familia [191].

3.5.2. PARAMETRIC DESIGN

According to the Oxford Dictionary, a parameter is either "a numerical or other measurable factor forming one of a set that defines a system or sets the conditions of its operation", or "a limit [...] which defines the scope of a particular process or activity". The word parametric, in turn, means being related to or expressed in terms of one or more parameters.

The literature on PD goes back several decades. In 1971, Moretti [192] defined parametric architecture as the study and definition of relationships between the dimensions of a design based on parameters. In the following decade, Kalay [193] defined parametric modeling as the computational representation of geometric relationships that are automatically updated upon parameter change. Already in the 21st century, Kolarevic [129] described PD as a process capable of instantiating several solutions in a consistent way by declaring the parameters of a particular design and not its shape [129]. This was followed by several similar definitions in the ensuing decades [39, 80, 138, 152, 189, 194–199], as well as by some different ones that consider PD either a process involving optimization to find a solution with an acceptable performance that satisfies the existing constraints [200], or a "contemporary architectural style that has achieved pervasive hegemony within the contemporary architectural avant-garde" [199, p. 1]. There are still other definitions of PD that narrow its scope to the exclusive use of algorithmic processes [202, 203].

Despite the existing divergences, the literature shows a predilection for the definition of PD as a design process based on algorithmic thinking that uses parameters and rules constraining those parameters [80, 83, 129, 153, 177, 192, 196, 197, 199, 204–209]. In most cases, its definition also embraces the BIM paradigm due to the latter's close connection with concepts of associative geometry and topological relationships [7, 132, 189, 210] that establish dependencies between different design elements. The definition adopted in this investigation considers PD a design approach based on the use of parameters to symbolically describe a design.

3.5.3. Generative Design

According to the Cambridge Dictionary, the word generative means the "capacity to produce or create something." In the literature, some authors define GD as a design process that involves evolutionary techniques in both the creation and production processes of design solutions [211–213], whereas others do not restrict GD to evolutionary processes, considering it a design approach based on algorithmic or ruled-based processes that generate multiple and, possibly, complex solutions [133, 138, 214–221]. As such, for several authors, approaches like algorithmic generation, cellular automata, evolutionary methods, generative and shape grammars, L-systems, self-organization, agent-based models, and swarm systems, are examples of GD [133, 138, 215, 217, 222–225].

Regarding these two perspectives, the first one excessively narrows the definition of GD, excluding other methods, beside evolutionary ones, that also generate designs, whereas the second one is often difficult to differentiate from other terms, namely PD. Therefore, this investigation adopts a third perspective that defines GD as a design paradigm that employs algorithmic descriptions that are more autonomous than PD. It involves methods that can generate complex outputs even from

simple algorithmic descriptions but where it is often difficult to correlate the algorithm with the generated output. It is, however, this lack of traceability between GD descriptions and the generated designs that allows GD methods to produce unexpected results, such as the "happy accidents" mentioned by Chaszar and Joyce [224, p. 168].

3.5.4. ALGORITHMIC DESIGN

The Cambridge Dictionary defines an algorithm as a "set of mathematical instructions or rules that [...] will help calculate an answer to a problem." This universal applicability of algorithms to any (computable) problem makes it difficult to distinguish AD from GD, which explains the tendency to consider these terms as synonymous [130, 188, 220, 227, 228]. This investigation places AD within the boundaries of GD, however, it argues that it should have a stricter definition to distinguish it from GD.

Regarding the literature, AD is the most recent term of the three, dating from the beginning of the 21st century. In 2003, Terzidis [130, 228] defines AD as an approach that involves the description of computer programs that "generate space and form from the rule-based logic inherent in architectural programs, typologies, building code, and language itself." [226, p. 70] For the author, AD allows designers to incorporate the "computational complexity and creative use of computers" [226, p.70] within the design workflow. For Bukhari [220], AD is a design approach that embraces both GD and evolutionary design approaches. For Queiroz and Vaz, AD allows "the user to design directly through code manipulation" [227, p. 797], therefore reducing the limitations of the existing modeling applications. Similarly, Humppi and Österlund [219] described AD as the process of controlling building form through user-created textual or graphical programs. Oxman [132] defined AD as the coding of explicit instructions to generate digital forms.

Within the context of this thesis, AD is regarded as a subset of GD, since it uses algorithms to generate models, but where it is possible to correlate the algorithm with the generated models (i.e., AD has traceability). In this view, AD produces fewer surprising results but provides a finer degree of control regarding the generated outcome that facilitates both debugging and maintenance tasks.

In short, this thesis regards DD and CD as two orthogonal terms, wherein the former is the use of computational tools in the design process and the latter the use of computation to develop designs. It considers that the other three terms GD, AD, and PD are within the scope of CD (Figure 3.3), defining GD as the explicit use of an algorithm to generate designs; AD as a subtype of GD that satisfies the traceability property; and PD as a design dependent on a set of parameters. Given that, typically, algorithms are parametric, it is not surprising to observe that AD can also be used to achieve PD.

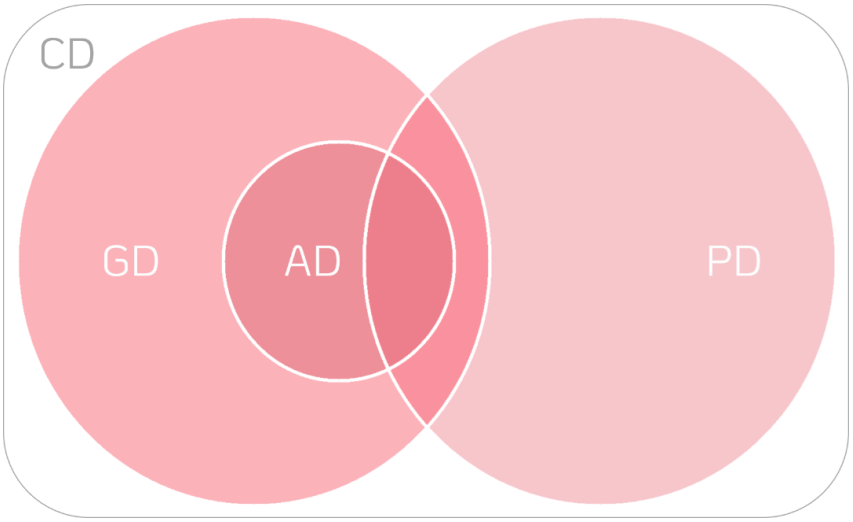


Figure 3.3. Conceptual representation of scope of the terms AD, CD, GD, and PD.

4. SKETCHING THROUGH ALGORITHMS

In the last two decades, Algorithmic Design (AD) has gained prominence in both architectural theory and practice [130, 132, 188, 194, 202, 220, 227–229]. A new generation of architects has been increasingly adopting the Integrated Development Environments (IDE) of their design tools due to the flexibility, automation, and accuracy it provides that, according to Terzidis, allows them to "go beyond the mouse, transcending the factory-set limitations of current 3D software" [129, p. 203]. However, despite overcoming traditional design possibilities, AD requires architects to learn new skills, such as programming, which is far from being trivial.

Currently, there are two main paradigms for AD, textual and visual, with the main difference between them being the way algorithmic descriptions are represented (Figure 4.1): in the first one, using textual programs prescribing the operations the computer needs to execute, whereas in the second, through an arrangement of interconnected iconic elements, forming dataflow graphs. The development of intuitive and easy-to-use visual programming IDEs made the latter paradigm more popular within the architectural community but current practices have evidenced several limitations that make it inappropriate for the development of large AD programs [202, 230, 231]. This explains why most software applications in existence, whether for architecture or not, were developed using textual programming [232].

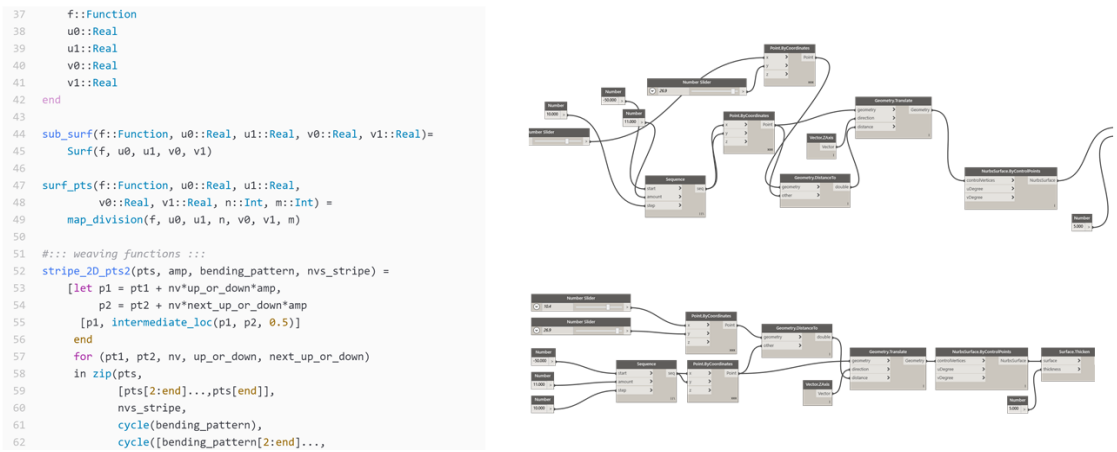


Figure 4.1. Text- and visual-based IDEs: Atom (on the left) and Dynamo (on the right) [233].

This chapter elaborates on the emergence of AD in architecture and the reasons behind its increasing use. It presents the strengths and limitations of the currently existing AD paradigms and tools, while identifying the main barriers that still hinder their widespread use.

4.1. THE BIRTH OF ALGORITHMIC DESIGN

Algorithms are everywhere and they are a fundamental part of current technology. Most of the design tools released since the late 70s support AD approaches, integrating their own IDEs to satisfy the users' need to automate design tasks and deal with more complex design problems. Unfortunately, the development of algorithms is an abstract and unfriendly task and, thus, some of the first design tools tried to hide the algorithms from their target users, providing them with mostly Graphical User-Interfaces (GUIs) containing ready-to-use menus of operations and supporting point-and-click interactions with the tool's interface. Sketchpad [234] is one prominent example: it allowed users to visualize and manipulate geometry on a computer by creating digital representations of points, lines, and shapes in a manual and intuitive way. Sketchpad was followed by many other tools with increasingly sophisticated GUIs that replicated, in the digital medium, the typical modus operandi of architecture professionals.

Interestingly, it soon became evident that, independently of the sophistication of the GUI, the interaction process itself was relatively slow and, instead of moving a mouse to select an item in a menu, it was faster to just press a few memorized keys for, e.g., copying and pasting. Similarly, instead of searching in a myriad of menu items and sub-items for an intended action, it was much more practical to just write the name of the action. This fact prompted vendors to support textual commands, including the ability to provide command arguments and then sequences of commands, in what became known as macro commands. Despite being relatively simple, these macros allowed users to automate design tasks by describing the sequence of commands that was needed to perform each of the tasks. Having this possibility, the designers' temptation to automate increasingly more complex tasks started to grow, not only requiring more advanced control features, like conditional structures and loop/while iterations, but also stressing the limits of these command-line interfaces. To address the users' ambition, the tools' developers focused on evolving the available automation mechanisms to make them more programmable and customizable. As a result, most tools started to provide their own programming languages, two important examples being ArchiCAD's Geometric Description Language (GDL) and AutoCAD's AutoLISP, announced in 1983 and 1986, respectively. In most cases, an IDE was also added to facilitate the programming task, incorporating features that increased its intuitiveness and reduced its abstractness, such as syntax highlighting. CAA (1998) for CATIA and Visual LISP (1997) for AutoCAD are two examples. This scenario opened the opportunity for AD approaches to emerge, as designers could now automate design tasks and explore a wide range of design solutions by resorting to algorithms.

Unfortunately, the programming languages that were being proposed suffered from several problems, the most serious one being the fact that they were tool-specific, thus locking-in users to that tool: for instance, AutoLISP could only be used with AutoCAD, GDL with ArchiCAD, and RhinoScript with Rhinoceros 3D. This was an advantage for the tool's vendor but a serious disadvantage for the designers, as they could not reuse their AD programs on a different design tool.

A second problem was the obsolescence of the proposed programming languages. For example, GDL was based on BASIC, a programming language that was already considered very limited at the end of the 80s but continues to be the main user-accessible programming language of ArchiCAD more than thirty years later. In the same vein, when AutoLISP was presented, advanced Lisp dialects, such as Scheme and Common Lisp, had already solved some of the problems affecting their predecessors; however, AutoLISP did not benefit from the lessons learned and, to avoid breaking the numerous user-developed programs already in existence, the language did not evolve and is nowadays an obsolete programming language.

To address these problems, while facilitating the learning process of architects, several tools started to support modern, easy-to-use programming languages, with which architects quickly became familiar. Python is one such example, which is currently available in several design tools, such as 3D Studio Max (2013), Revit (2009), Rhinoceros (2011), ArcGIS (2004), and City Engine (2010). Another strategy adopted was the integration of visual programming environments targeting both experienced and non-experienced users in AD, such as Generative Components (2003), Grasshopper (2007), and Dynamo (2011). Nevertheless, it is ironic that what was initially meant to reduce the architects' initial investment when adopting AD approaches and simplify their modeling experience, sooner became more complex and abstract due to their tendency to innovate and go beyond the boundaries already explored.

4.2. ALGORITHMIC DESIGN PARADIGMS

Despite the cutting-edge aura that surrounds it, AD emerged as the natural consequence of a design process that fully automated modeling tasks. In AD, instead of manually modeling the design in a design tool, the designer develops algorithms whose execution creates the intended model [181]. This has several advantages over traditional digital modeling processes, including precision, repeatability, and ease of change. A more relevant advantage comes from the almost unavoidable parametrization of algorithms, which makes designs intrinsically parametric, allowing the architect to explore their implicit design spaces, i.e., the set of possible solutions, by simply changing their parameters.

Initially, only the textual paradigm was allowed in most AD tools. The technology available at the time was not powerful enough to support a more graphically demanding approach like visual programming. With the technological evolution, new features were incorporated in the already existing text-based AD environments and, later, new visual-based AD tools emerged. While in the textual paradigm efforts were made to improve the development and debugging of text-based AD programs, the focus in the visual paradigm was to bring programming as close as possible to the visual nature of design thinking and thus make it less abstract and more intuitive for non-experienced programmers. Among the two, the latter paradigm became more popular in architecture due to (1) the greater intuitiveness and user-friendlier graphical-based appearance of its related tools, (2) the ease with which users can obtain results after learning just a small set of programming techniques [202, 231], and (3) the available features that facilitate the development and manipulation of AD programs, such as traceability, i.e., the relationship between parts of the program and those of the generated model, to improve program comprehension, maintenance, and debugging [218]; real-time feedback, to immediately display the impact of program changes; and user-interaction mechanisms (e.g., sliders, toggles, and button devices), to represent and manipulate the design's parameters in real-time.

Unfortunately, the features that make visual programming languages (VPLs) more attractive than textual programming languages (TPLs) for the development of simple programs become serious shortcomings for the development of large programs, the first one being their lack of scalability, which is essential to deal with more complex design problems [230, 235]. Another limitation regards the intelligibility of the resulting AD programs, whose large number of nodes and even larger number of connections make their understanding and manipulation difficult [82, 231, 236, 237]. The lack of user-programmable features of most VPLs is another common limitation, often confining users to the predefined functionalities available in each VPL and thus hindering the development of solutions that need more advanced features [202]. Another shortcoming is their accentuated drop in performance when executing large AD programs, often making the tools loose interactivity or even crash [218, 238, 239]. Lastly, the lack of version control mechanisms, i.e., mechanisms responsible for managing and tracking program changes, of most VPLs is another important limitation, making it difficult to support the division of labor typical of collaborative design processes [236, 240]. As a result, despite their attractiveness, VPLs prove to be only advantageous to solve simple design problems; in the remaining cases TPLs are preferable [238, 241, 242]. The inability of VPLs for long-term use [243] motivates the transition from the visual to the textual paradigm [241]. Nevertheless, this process is not trivial, particularly when the user is accustomed to VPLs' interaction mechanisms, which rarely exist in TPLs, and tends to lack the consolidation of important theoretical bases, such as

variable declaration and flow control structures, a drawback that gets even worse when addressing more complex design problems.

Given the limitations of VPLs, it is not surprising to verify that they were eventually extended to also support textual programming and, thus, benefit from its expressiveness and scalability. The need to learn textual programming has therefore become increasingly evident in the field but, unfortunately, it still takes time to achieve the level of proficiency needed to deal with large scale designs.

Based on the idea that learning a VPL first can facilitate learning a TPL later [243], some hybrid programming approaches were proposed, combining both textual and visual programming features in a single programming language. DesignScript [244] and KhepriGH [245] are two relevant examples. Another solution proposed to smooth TPLs' learning curve focuses on the incorporation of the most desired VPLs features in the textual paradigm to make TPLs more intuitive and more suitable for teaching purposes [246]. Lastly, the provision of ready-to-use algorithmic libraries and strategies addressing different design problems and scenarios was also proposed with the aim of facilitating and accelerating the development of AD solutions of varying complexities [4, 5, 7, 76–78, 83, 176, 177, 247], allowing users to choose and combine predefined AD program fragments in a guided way.

4.3. Algorithmic Design Tools

Since long, design tools have incorporated their own AD environments, initially mostly text-based, to extend their modeling capabilities and allow the development of AD solutions. Design Augmented by Computers (DAC-1), developed by General Motors and IBM and released in 1964, was one of the first CAD tools to include a custom programming language intended for users, the Descriptive Geometric Language (DGL), to allow them to create new modules that could be called from within the interactive environment [248].

Regarding the field of architecture, in 1983, ArchiCAD started to incorporate a text-based IDE to allow users to create their own architectural objects using the GDL programming language [249]. Despite being easy to use, the latter lacked good abstraction mechanisms [250], making it very difficult to create large programs. Moreover, the IDE was a simple text editor that did not provide sufficient support to reduce the difficulty of the programming task.

Three years later, in 1986, AutoCAD started to incorporate its own TPL, AutoLISP, to allow users to algorithmically manipulate AutoCAD's entities and extend its modeling capabilities. Still, writing AD programs was a difficult task that was only facilitated in 1997 with the integration of the

text-based IDE Visual Lisp in AutoCAD Release 14[6], which considerably improved their readability through *syntax highlighting*[7] (Figure 4.2-A) and their debugging by identifying and locating existing errors.

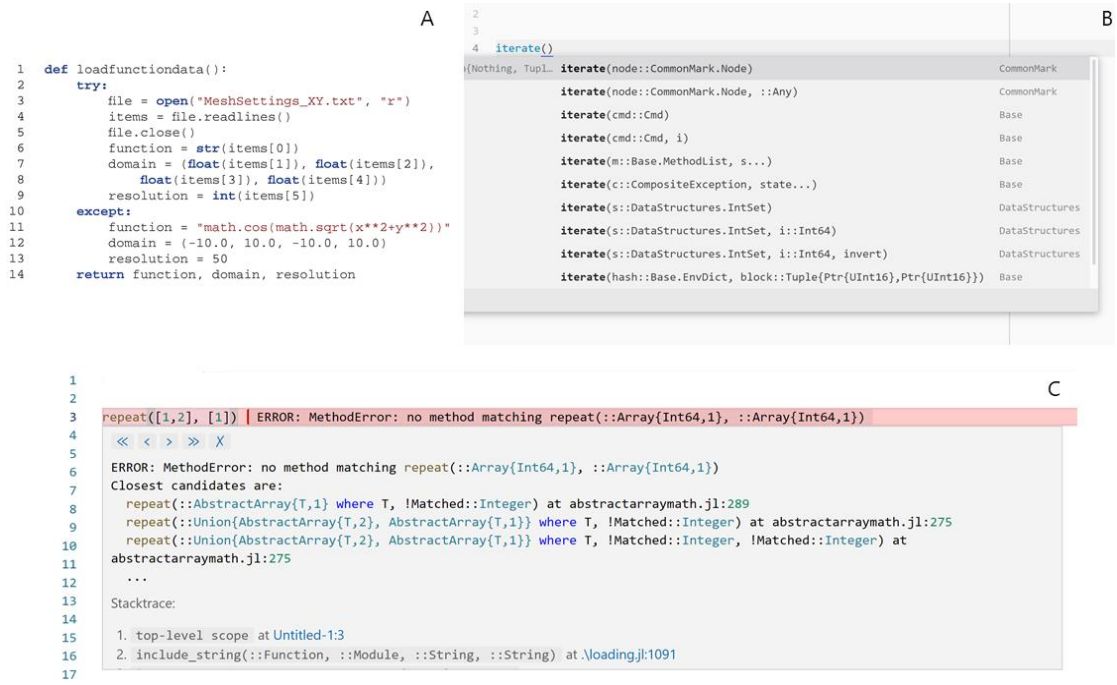


Figure 4.2. Examples of text-based IDE features: A. syntax highlighting (Rhino.Python)[251]; B. code autocompletion (Atom); C. Error highlighting and debugging support (Visual Studio Code).

In 1997, 3D Studio Max was also extended with an IDE for a new TPL, MAXscript, an object-oriented programming language with a simple syntax suitable for non-programmers. In addition to providing advanced programming mechanisms, such as *higher-order functions*[8] and automatic *broadcasting*[9], MAXscript allowed users to easily develop GUIs for their own AD programs.

In the ensuing two decades, several design tools followed this trend, integrating similar text-based IDEs. Rhinoceros 3D's Rhino.Python and RhinoScript IDEs are two examples that, in addition to the previous features, already supported *code auto-completion*[10], albeit in a limited way. Another example is Bentley's GCScript, which presented a more advanced version of code auto-completion, displaying suggestions of variables and functions in the active scope, together with their description, e.g., the parameter list and type of input received (Figure 4.2-B).

[6] First, as a paid add-on but then included in AutoCAD 2000.
[7] The use of different colors or fonts to display the different program elements.
[8] A function that either receives one or more functions as argument or returns a function as outcome.
[9] The mapping of a function over an array or a matrix by element.
[10] The ability to display possible completions for the word the user is typing in.

Despite all the progress made, most text-based AD tools currently available only provide limited support for the programming task, especially when compared to other professional IDEs, such as Atom, Eclipse, and Visual Studio Code. In addition to being free of charge, these IDEs include important features such as (1) *IntelliSense* code completion, displaying smart completions based on the variables type and function definitions (Figure 4.2-B); (2) advanced debugging support, suggesting possible solutions to solve the identified errors (Figure 4.2-C); (3) refactoring mechanisms, i.e., mechanisms to improve the algorithmic structure of a program while preserving its semantics, increasing the programs readability and maintainability; and (4) version control, facilitating the detection and correction of program errors and the adoption of collaborative work practices. It is therefore not surprising that some tools have opted to use these independent IDEs instead of developing their own, such as ArchGIS, Unity, and Unreal. In almost all cases, however, the available AD tools lack real-time feedback and traceability, two important features to make the manipulation of text-based AD programs more intuitive and closer to the architects' visual nature, while reducing the programming experience needed for their use.

Regarding visual-based AD tools, Generative Components is a pioneer example, a tool released in 2003 to extend the modeling functionalities of MicroStation. To facilitate the programming task, this tool provides users with several interface views [198], allowing them to interactively manipulate design variables through number sliders and automatically see the results. It also supports (1) real-time feedback, immediately displaying the results when connecting nodes or changing input values, (2) bidirectional traceability between the program and the model, highlighting the generated part of the model when selecting a component in the AD program (Figure 4.3-A) and the other way round, and (3) the automatic generation of visual-based code from manually created geometry.

Another example is Grasshopper, a tool released in 2007 that had the advantage of being free of charge and being constantly further extended with an increasing number of plugins. Besides supporting visual input mechanisms, i.e., mechanisms that allow using manually created geometry in the modeling tool as input to the AD program, this tool also provides debugging features, automatically changing the color of the AD program's graphical components according to their status and providing information about the existing problem (Figure 4.3-B).

Dynamo is another popular example, which was released in 2011 to extend the modeling capabilities of Revit. Besides having features similar to those of the previous tools, Dynamo has the novelty of supporting the development of AD programs in the same environment where the resulting geometry is displayed.

Examples of other, less-used visual-based AD tools include VectorWorks' Marionette, Unreal's Blueprint, and CATIA's xGenerative Component. The first two have the newness of providing

functionality highlighting, coloring the available components according to their function (Figure 4.3-C). The latter, in turn, supports two ways of developing AD programs, by either dragging and dropping the graphical elements in the canvas and connecting them, just like it is done in other competing tools, such as Grasshopper and Dynamo, or manually modeling the geometry in the modeling environment, with the AD program describing it being automatically updated accordingly. Further examples of AD tools are chronologically presented in Figure 4.4.

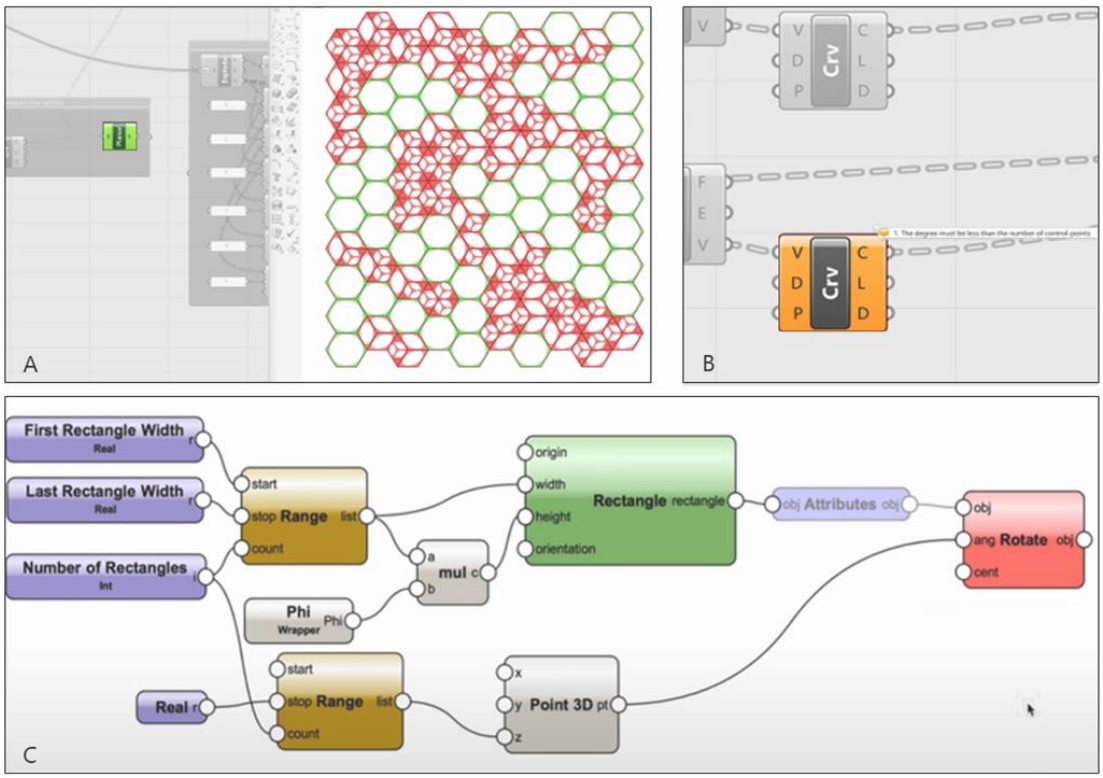


Figure 4.3. Examples of visual-based IDEs features: A. one-direction traceability (Grasshopper); B. debugging support (Grasshopper); C. functionality highlighting (©2018 Vectorworks, Inc).

Given the limitations of VPLs for large-scale architectural problems, most visual-based AD tools were extended with further functionalities aiming at solving part of the existing problems, one of them being the poor intelligibility of the AD programs. Different *modularization*[11] techniques were integrated to improve both the structure organization and readability of AD programs, allowing users to either group graphical elements by task or create new ones hiding other arrangements of graphical elements. Grasshopper's *clusters* and Dynamo's *code blocks* are two examples. These techniques, however, are rarely used by architects, the resulting AD programs therefore remaining difficult to understand in most cases.

[11] Software design techniques that separate the programs' functions into independent, interchangeable pieces.

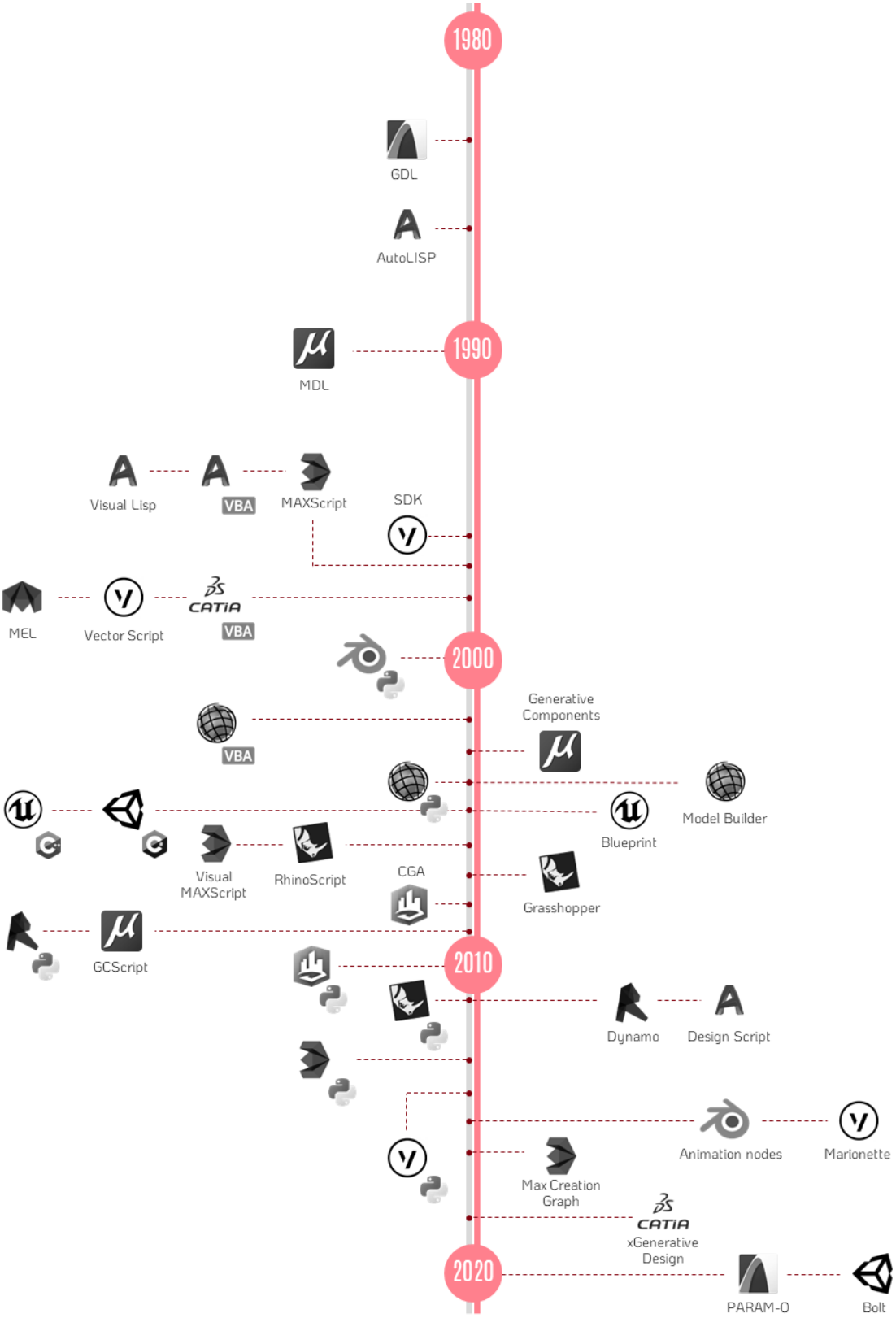


Figure 4.4. Timeline on text-based (left) and visual-based (right) AD tools.

To solve the lack of scalability of most VPLs, textual programming extensions were made available for VPL's to benefit from TPLs' basic programming mechanisms, such as loop iterations, recursive functions, and higher-order functions, which are critical for more complex designs problems [230, 235]. Unfortunately, even with these textual components, it is often difficult to develop large-scale AD programs in these tools: Grasshopper, for instance, provides components for textual programming using the languages VB.net, C#, and Python, but these are intended for small AD programs as it hardly supports large-scale development, namely, division of programs in multiple files.

Lastly, to address the poor interoperability between most visual-based AD tools and the design tools commonly used in architecture, some of them were extended with further functionalities embracing different analysis, optimization, and fabrication strategies. Relevant examples include Grasshopper's plugins for structural analysis and form finding, such as Kangaroo[12], Karamba3D[13], Millipede[14], and Peregrine[15]; for lighting and thermal analysis, which include Ladybug[16], Honeybee[17], DIVA[18], and ClimateStudio[19]; for design optimization, namely Galapagos[20], Goat[21], Octopus[22], and Opossum [252]; to export models into BIM tools, such as Hummingbird[23], Lyrebird[24], GHRevit[25], Rhino-Grasshopper-ArchiCAD[26], and RhinoBIM[27]; and for manufacturing and assembly, which include HAL[28], FabTools[29], BowerBird[30], OpenNest[31] and Kuka|PRC[32]. Other examples include Dynamo's structural and energy analysis packages, and its addons for daylighting and thermodynamic analysis (e.g., Honeybee), optimization (e.g., Optimo), and fabrication (e.g., DynaFabrication, Fabrication API, 3BMLabs.DigiFab, and ParametricMonkey). However, despite

[12] D. Piker. 'Kangaroo' (2015).
[13] C. Preisinger and B. und G. Z. GmbH. 'Karamba3D' (2014).
[14] Panagiotis Michalatos. Millipede (2014).
[15] LimitState 3D. 'Peregrine' (2019).
[16] M. Roudsari (Ladybug Tools LLC). 'Ladybug' (2013).
[17] M. Roudsari (Ladybug Tools LLC). 'Honeybee' (2014).
[18] Solemma. 'DIVA' (2019).
[19] Solemma. 'ClimateStudio' (2021).
[20] D. Rutten. 'Galapagos: Evolutionary Principles Applied to Problem Solving' (2010).
[21] S. Floery. 'GOAT' (2013).
[22] R. Vierlinger. 'OCTOPUS' (2012).
[23] M. Guttman and T. Meador. 'Hummingbird' (2012).
[24] LMN Architects. 'Superb Lyrebird' (2014).
[25] S. Davidson. 'GHRevit' (2012).
[26] GRAPHISOFT. 'Grasshopper-ARCHICAD' (2017).
[27] Virtual Build Technologies LLC. 'RhinoBIM' (2015).
[28] T. Schwartz. 'HAL | ROBOT PROGRAMMING & CONTROL' (2017).
[29] Florian Frank. 'FabTools' (2013).
[30] Thomas J. Oberbichler. 'BowerBird' (2015).
[31] Petras Vestartas. 'OpenNest' (2021).
[32] RobotsInArchitecture. 'KUKA|PRC' (2011).

solving part of the existing interoperability issues, they continue to suffer from most of the limitations typical of VPLs, often failing to respond to the complexity of current design processes.

Regarding the focus of this thesis, building envelopes, there are already some tools available to facilitate the algorithmic development of facade design solutions. One example is ParaCloud Gem[33] (Figure 4.5-A), a 3D pattern modeler that contains features to (1) map 3D elements on a mesh, (2) subdivide and edit surfaces, (3) integrate performative requirements, and (4) allow the rapid prototyping of the solutions through 3D printing. Other examples are Dynamo's packages for (1) surface paneling, (2) mapping elements on a surface, and (3) pattern creation and manipulation, such as Quads from Rectangular Grid (Figure 4.5-B), Ampersand, Clockwork, LunchBox, MapToSurface, Pattern Toolkit, and LynnPkg. Further examples include several extensions for Grasshopper, namely PanelingTools[34], which includes surface paneling functionalities and rationalization techniques for analysis and fabrication; LunchBox[35], which has functionalities to explore mathematical shapes, surface paneling, and wire structures; Weaverbird[36], which contains mesh subdivision procedures and mechanisms to help prepare meshes for fabrication; Parakeet[37] (Figure 4.5-C), which provides functionalities to develop algorithmic patterns resulting from tiling, geometric shapes and grids subdivisions, edge deformation, etc.; and SkinDesigner[38] (Figure 4.5-D), which includes mechanisms to produce facade design solutions made of repeating elements.

In addition to being easy to use and intuitive, these tools facilitate typical modeling procedures of facade design processes, such as the generation of point-grids on a surface, mapping elements in different ways, applying *attractors*[39] to control elements size, rotation, among others. However, their use is mostly manual, requiring the user to interact directly with the tool's environment instead of using algorithms, thus favoring iterative user-driven processes that can be tiresome and error-prone. Moreover, they are mostly based on VPLs and thus suffer from their limitations [231, 241], particularly scalability. Lastly, most of the above-mentioned tools are limited by the available predefined operators, which can hardly be configured by the user to respond to more specific problems [202].

[33] ParaCloud. 'ParaCloud GEM' (2011)
[34] R. Issa. 'PanelingTools for Grasshopper' (2013).
[35] N. Miller. 'LunchBox' (2011).
[36] G. Piacentino. 'Weaverbird' (2009).
[37] Esmaeil. 'Parakeet' (2019).
[38] Sgaray. 'SkinDesigner' (2017).
[39] Elements acting like virtual magnets.

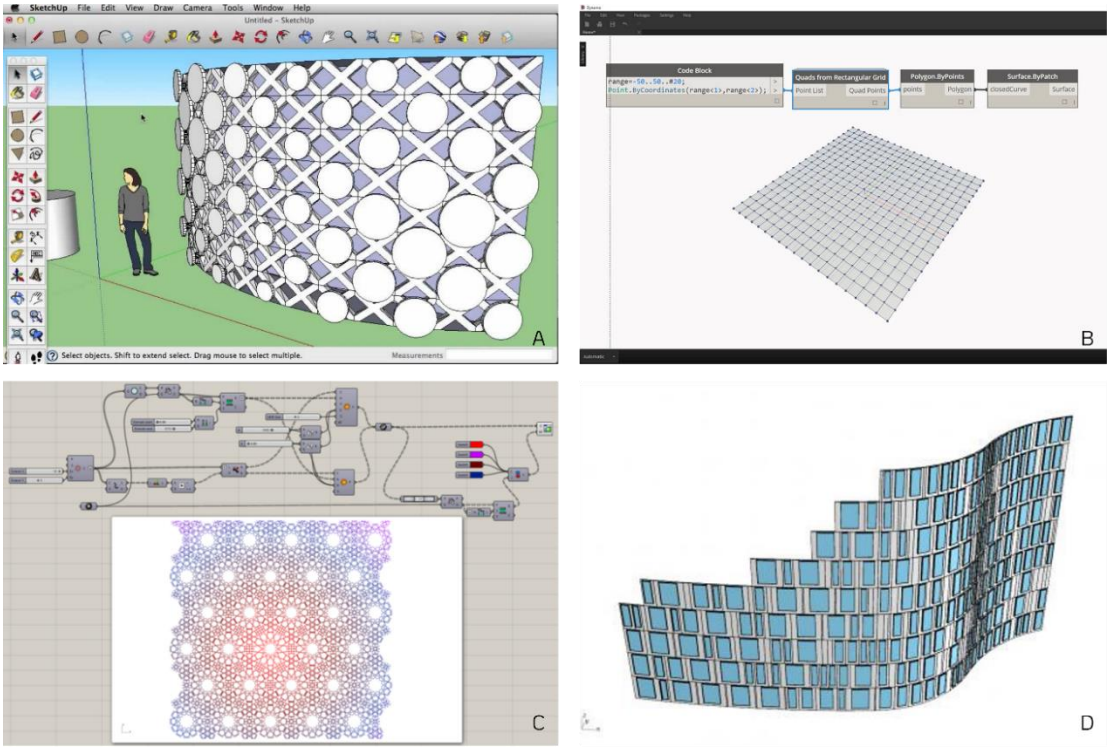


Figure 4.5. Paracloud GEM (©paracloud.d.e.s.i.g.n.e.r via blogspot); Quads from Rectangular Grid package [233]; Parakeet plugin (©Esmaeil Mottaghi); SkinDesigner (©sgaray).

4.4. ALGORITHMIC DESIGN PATTERNS

AD has been used in architecture for various purposes, one of them being the design of building envelopes. According to Kolarevic and Malkawi [143], AD is advantageous for their design exploration and optimization due to automating and, thus, accelerating the generation of several design variations, while allowing their iterative evaluation regarding multiple criteria, which are typical of facade design processes. Moreover, AD has also proved to be useful for material exploration strategies, as evidenced by Menges [253], and fabrication processes, as explored by Iwamoto [161], Dent & Sherr [254], and Gramazio & Kohler [160]. To benefit from it, however, architects have to learn new skills, such as programming, and know how to control them [161]. To smooth part of the existing limitations, some AD strategies were proposed in the literature, such as the use of modularization and *Design Pattern* (DP) techniques.

According to the literature [7, 255–257], *pattern* is a general solution to a recurring problem, often described with a high level of abstraction, that can be repeatedly re-used and re-defined in different contexts. In the same vein, DP is often described as the abstraction of known design strategies or solutions to real (and often recurrent) design problems that allow architects to take advantage of previously used design knowledge [3–5], while avoiding repeated reinvention in the

resolution of new ones. In this thesis we consider DP as a frequently applied problem-solving design strategy that is generalized in an algorithmic perspective.

The idea of DP has been closely related with the computer science field for a long time, becoming part of several well-known computation-based theories and methods. In architecture, this idea was introduced in the early 60s by Christopher Alexander [258], who developed a language based on a network of patterns representing repeatedly occurring problems and their solutions [4]. For the author, the resolution of such problems could originate generic solutions that could be used multiple times and in an infinite variety of design scenarios. In his work *A Pattern Language* [4], the author presents 253 patterns to be used as generic guiding principles for architectural design. Curiously, although his academic background was architecture, Alexander's work had a great impact in computer science [259], inspiring several design methodologies at the time, namely the design of programming languages, modular programming, and object-oriented programming, as well as different authors, including Peter Naur [260] and Richard Gabriel [261].

With the emergence of computational design approaches, such as AD, DPs also gained visibility in the field, which is reflected in the literature of the last decades. Several authors have recognized the advantages of DPs to assist architects with their computational thinking and to reduce the complexity and abstractness of AD approaches [3, 258]. Woodbury et al. [76], for instance, evaluated the capability of DPs to help architects with the identification of suitable solutions for certain design problems, concluding that these strategies are suitable to anticipate unexpected design changes during the design process. Similarly, Qian [7] demonstrated the ability of DPs in augmenting and supporting architectural design practices based on AD. According to Khwaja & Alshayeb [262], using DPs reduces the development time of AD solutions, while facilitating the communication between team elements working on the same project. Lastly, Yu & Gero [177] investigated the trends in using DPs at early design stages, concluding that these are more often used in Parametric Design (PD) environments based on AD.

Regarding their practical application, Hudson [77] developed a reusable parametric tool based on the capture of well-defined design methods and practical experiences from a group of designers, concluding that several procedures in design offices could be converted into reusable tools for design. Woodbury [83] and Chien et al. [176] developed pattern management tools containing collections of predefined algorithmic DPs, with the aim of formulating knowledge for PD users. Similarly, Lin [247] proposed an algorithmic framework to help architects in the exploration and development of algorithmic descriptions to solve their design problems. Sousa and Paio [263] elaborated a taxonomy relating and classifying patterns for small public spaces that responds to the demands of current digital design processes. Finally, in the field of facade design, Su & Chien [78]

proposed a set of algorithmic patterns to support the development of architectural facades at initial design stages.

Nevertheless, most of the previous studies focus on the modularization of AD solutions based on VPLs and, thus, are restricted to their inherent shortcomings. Moreover, they attempt to support AD processes only at initial design stages, not integrating other types of processes and algorithms, such as analysis and optimization, neither connecting them to the multiple design tools that are nowadays becoming ubiquitous. Finally, to the best of our knowledge, only Su & Chien [78] addressed the application of algorithmic patterns to explore building envelopes, concluding that the same pattern can appear in different design workflows as well as in different design stages, and that multiple patterns can be used together. The research, however, only applies generic algorithmic patterns proposed by other authors, not proposing others more specific for facade design problems. Moreover, it only targets the visual programming paradigm, therefore being circumscribed to its limitations.

5. FACING NEW DESIGN CHALLENGES

During the 50's, the need to respond to the constantly changing social and technological backgrounds, together with the growing environmental concerns and greater awareness on the buildings' ecological footprint, motivated new design approaches that went beyond aesthetic, structural, and functional levels [143]. According to Fasoulaki [215], architects started to realize that the buildings' behavior could be a relevant input in design exploration processes, contributing to more informed and conscientious design practices. For Anton & Tănase [264], integrating analysis information in the design workflow potentially leads to better performing solutions that simultaneously meet the original design intent. This idea has been evolving in the literature (Figure 5.1) under the name of performance-based, performance-driven, performance-oriented, or even performative design [125, 126, 134, 136, 142, 265].

This chapter is organized in three main sections: the first addressing current environmental concerns and the role of design analysis in current architectural practice; the second presenting some of the existing design optimization strategies and their application in facade design processes; and the last one illustrating the integration of building performance in real case scenarios, evidencing the need for architects to collaborate with differently skilled professionals and benefit from diverse Computational Design (CD) strategies.

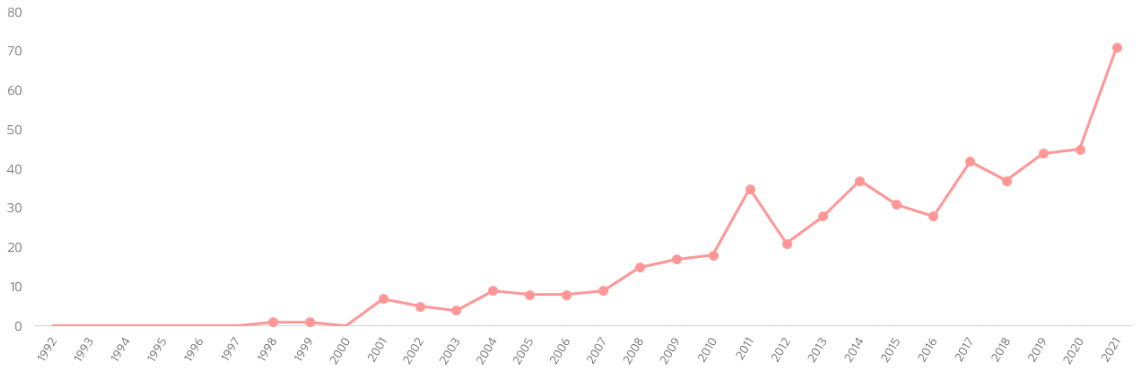


Figure 5.1. The use of performance-related terms as a keyword from 1992 to 2021 in three scientific sources: CuminCAD, Science Direct, and Scopus.

5.1. THE ROLE OF DESIGN ANALYSIS

The notion that buildings are one of the main CO_2 emitters and spenders of energy resources [266] has been motivating design practices that aim at reversing the situation. These practices have been

supported by the latest technological developments, including the increase in computational power, which have been enabling architects to explore design solutions that better address environmental requirements. As stated by Tibbits, "computational tools and methods allow designers to go far beyond what they could conceive independently" [265, p. 145].

5.1.1. Growing Environmental awareness

We are currently witnessing a dramatic environmental problem and the Architecture, Engineering and Construction (AEC) sector is one of its main contributors. According to the literature, buildings in the European Union are responsible for 40% of the total energy consumption and 36% of greenhouse gas emissions [268], which are one of the main contributors for climate change and global warming [269]. Most of these emissions result from the buildings' operational costs, such as heating, cooling, lighting, and ventilation, as well as from their construction, renovation, and demolition [268]. In 2012, for instance, buildings consumed around 75% of European's energy consumption, and almost 70% of it was for space heating [70].

The statistics therefore show that it is critical to minimize the ecological footprint of buildings [70, 71], and, to that end, several regulations and incentives were established [270] aiming at contradicting the growing trend of global CO_2 emissions [70]. The Building Research Establishment Environmental Assessment Method (BREEAM), first published in 1990, is the oldest method for assessing, rating, and certifying the buildings' sustainability, covering a wide range of environmental issues. Then, in 1997, the Kyoto protocol [271] was signed, constituting one of the first initiatives that aimed at limiting CO_2 emissions. Other examples include the European Union's Energy Performance of Buildings Directive (2002/91/EC, 2010/31/EU and COM/2016/0765) targeting the improvement of buildings' energy performance; the green building certification Leadership in Energy and Environmental Design (LEED) developed by the non-profit U.S. Green Building Council; and the Japanese green building management system Comprehensive Assessment System for Built Environment Efficiency (CASBEE).

The existing legislation, however, requires architects to evaluate the performance of their designs regarding different criteria [272] to ensure the proposed metrics are met [68]. This, in turn, made building design a more demanding task as it has now to respond to the ever-existing design requirements, such as aesthetics, structural, comfort, and economic, as well as to an increasing number of performance regulations and metrics. To assess this multiplicity of requirements, several analysis tools were released in the last decades, allowing architects to evaluate the performance of their designs regarding different criteria [71]. Nevertheless, obtaining accurate analysis results is not a straightforward task due to the wide variety of external (e.g., climate, geographic location, site

conditions, etc.) and internal (e.g., occupants behavior, operating schedule, type of activities, etc.) factors that typically affect buildings' performance [71].

5.1.2. Assessing Building Performance

In the last decades, several analysis tools for architectural design were released, popular examples being EnergyPlus[40] and TRNSYS[41], two whole-building energy simulators; Radiance [273], for (day)lighting simulations; DAYSIM[42], for climate-based daylight simulations; and Robot [274] for structural analysis. In general, these tools allow architects to simulate the behavior of their designs and thus become more aware of the impact of design changes on their performance [71]. However, despite their usefulness for architectural design, these tools are still far from being widely used in the field. On the one hand, there are still few practitioners who use analysis tools in their design processes. On the other hand, those who use rarely benefit from them at early design stages to support design decision-making processes, but rather at later design stages only to validate the performance of already well-defined solutions [68, 72, 269].

According to the literature, the need for specialized knowledge, the high cost of most analysis tools, and the idea that they restrict the architects' creative process are some of the factors that are still hindering their widespread adoption [68, 272]. Moreover, the fact that most analysis tools target final design stages and have interoperability issues with the design tools architects typically use is another common barrier to their use, often resulting in a laborious process prone to information loss and to the accumulation of errors [68]. Another critical hindering factor is the fact that most analysis tools are single domain, while architectural problems are multi-domain, making it necessary to use multiple analysis tools to evaluate different criteria. This, in turn, is further amplified by the specific requirements of each analysis tool, which often require the production of different versions of the architectural design model, i.e., a digital model containing the project's geometric and sometimes construction information, containing the information needed for the type of analysis performed [72]; also known as analytical models. Lastly, the long computation times of many analysis tools is another common barrier to their use [71, 272], often being incompatible with the projects' deadlines.

These limitations become even more pronounced with the need of architectural design processes to iteratively redesign and reanalyze design solutions, often resulting in the repeated production of several analytical models and the consecutive execution of multiple performance evaluations. Given the time and effort required for each independent task, evaluating an acceptable

[40] DOE and BTO, 'EnergyPlus' (1996). https://energyplus.net/ (Retrieved on <July 12th 2022>).
[41] TRNSYS. 'TRNSYS: a transient system simulation tool'. http://www.trnsys.com/ (Retrieved on <July 12th 2022>).
[42] C. Reinhart. 'DAYSIM: Advanced Daylight Simulation Software' (2010).

sample of possible solutions is, in most cases, an impracticable scenario [272]. To address the need for a faster and more reliable data-flow process suiting the iterative nature of architectural design, some design tools started to integrate their own analysis strategies [80], partially solving the effort associated with the use of multiple analysis tools. Nevertheless, most of them still present a limited (1) modeling flexibility, mainly in representing nonstandard solutions; (2) accuracy, mostly in analyzing less conventional geometries and construction schemes; and (3) information support, often making no suggestions about which design direction to follow and how to translate analysis results into design changes.

5.2. The Role of Design Optimization

The use of optimization in architecture was motivated by the need to explore the design space faster and more efficiently in the search for better performing solutions [272]. In general, optimization comprises the study of optima and the methods to find them [275]. Mathematically, this process aims at identifying the best element in a set of alternatives according to a criterion [276]. Such a problem can be described as $Minimise\ F(x_1, x_2, \dots, x_n)$ while subjected to $G(x_1, x_2, \dots, x_n) \geq 0\ \ x_i \in S_i$, wherein F is a vector of objective functions and G a vector of functional constraints, and x_i is a variable belonging to the domain S_i.

Similarly, architectural optimization searches for the best solution according to a fitness function within a design space of possible solutions; a process that typically entails the iterative remodeling of a design and its subsequent analysis to check if the performance goals are met [269]. This search can consider either one or more objectives, the former being classified as single-objective optimization and the latter as multi-objective optimization. In mathematical terms, the main difference between them is the number of objective functions to optimize [270]. In most cases, however, real design problems require handling multiple objectives that are often conflicting in different degrees [277].

Optimizing architectural designs requires simplifying a real-world complex problem, often dealing with multiple conflicting requirements, into a mathematical one. Given that multiple conflicting goals often result in a set of possible solutions that are not optimal for all objective functions [277], architectural optimization therefore searches for those solutions that make the best compromises between goals [272]. In this thesis, we name the set of solutions with the best trade-offs for a certain scenario as the *optimized design space*.

During this process, the higher the number of potential solutions analyzed, the greater is the probability of finding or getting close to the optimized design space [270]. Obviously, choosing from

all possible solutions would be the ideal scenario but this is not yet possible due to the insufficient computational power of existent computers. In any case, these processes are important to remind architects of possible design solutions that might otherwise not occur to them [86]. Moreover, even if the optimized design space is not reached during the optimization, the probability of obtaining a solution with a better performance is much higher than not doing any optimization at all [272, 278].

5.2.1. Architectural Design Optimization

In the last decades, the number of works on building optimization has noticeably increased in the literature [71, 270, 279–281]. For Nguyen et al. [278], this was stimulated by the stricter environmental requirements of the AEC industry, as well as the recent developments in computer science, such as the emergence of Algorithmic Design (AD) approaches, which, according to Oxman [135], have been facilitating the automatic incorporation of design changes and their subsequent performance analysis. Regarding architectural practice, the scenario is different because, even though designers and engineers already use simulation tools to analyze their projects, these are often employed without affecting the buildings' form, let alone guiding the geometric exploration process [264]. As in most cases the design space is manually explored, i.e., the solutions are adapted by hand to incorporate the analysis results [135], only a few design alternatives are evaluated, thus resulting in small design changes due to time constraints and associated effort [68]. Moreover, since only a few design requirements are usually considered in such processes, the results tend to be either unrealistic or non-effective [269].

Fortunately, the emerging CD approaches, such as AD, are changing this scenario, enabling the automation of optimization routines and thus facilitating and accelerating the evaluation of wider design spaces [277]. Nevertheless, the search for near-optimal solutions is not a straightforward task in architecture because, first, most buildings are unique and their design has to respond to multiple conflicting requirements [270]; second, it requires exploring large design spaces [270], which typically results in high computation times; third, most of the available tools are little intuitive and require specialized knowledge; and lastly, it requires architects to mathematically formulate the optimization problem [276], which is far from trivial due to depending on several context-specific factors [72].

According to Machairas et al., the formulation of optimization problems is one of the main steps of any optimization process [272] for which there are several strategies available and whose selection has a direct impact in both the duration and success of the process. In single-objective problems, for instance, this is a straightforward task [270] as it focuses on minimizing/maximizing a single objective function. This simplicity therefore makes single-objective optimization the least time-consuming strategy but also the least informative and accurate as well [72]. Regarding multi-objective

problems, there are two main strategies [270]: converting the problem into a single-objective one by combining the multiple objectives into a single function, for instance, through weighted-sums or by adding penalty functions, or adopting a *Pareto-based optimization*[43]. From the two, the former is often easier to implement, less computationally demanding, and less time consuming since it must solve a single combined objective function [72, 282]. Nevertheless, it is also less informative, often returning a single solution and requiring prior knowledge on how the different objectives interfere with each other [282]. The latter, in turn, provides a set of solutions with different trade-offs between the multiple objectives, but requires higher computation times. In either case, however, there is no information about the relationship between design variables and goals and their impact on other design criteria like aesthetics. Moreover, both strategies are very sensible to the way the optimization problem is modeled, especially the former, and, in the case of the latter, it becomes difficult to graphically represent optimization problems dealing with more than three objectives, which hinders the comprehension of the results [72, 282, 283].

According to Machairas et al. [272], the selection of the optimization tool is another critical step that depends, among other factors, on the type of problem addressed, its mathematical formulation, and the search methods available. Among the existing design optimization tools, there are less user-friendly examples that require both specialized knowledge and programming skills, such as GenOpt [284], modeFrontier[44], ParaGen [285], and MultiOpt [286], but there are also easier-to-use options, providing visualization and interaction mechanisms that interoperate with either the CAD or BIM tools architects typically use. Grasshopper's multiple plugins for design optimization, such as Galapagos[45], Goat[46], Silvereye [287], Octopus[47], Wallacei[48], and Opossum [252] are some examples, as is the optimization tool Optimo [288] from Dynamo Studio[49] and the Optimization Module from DesignBuilder[50]. However, these examples are all based on the visual programming paradigm and thus share the same scalability issues [231, 241, 289]. Moreover, not all of them address multi-objective problems (e.g., Galapagos, Goat, and Silvereye are single-objective) and most present a limited range of optimization strategies, usually evolutionary ones (e.g., Octopus uses SPEA-2 and HypE; Wallacei, DesignBuilder's Optimization Module, and Dynamo's Optimo currently only use NSGA-II), thus reducing the probability of suiting the problem at hand [290, 291]. Lastly, very few

[43] A strategy that returns the set of solutions that best balance the multiple objectives.
[44] ESTECO. 'modeFRONTIER®' (1998). https://www.esteco.com/modefrontier (Retrieved on <July 12th 2022>)
[45] D. Rutten. 'Galapagos - Evolutionary Principles applied to Problem Solving' (2010).
[46] Rechenraum. 'Goat'. https://www.rechenraum.com/en/goat.html (Retrieved on <July 12th 2022>).
[47] R. Vierlinger. 'OCTOPUS' (2012).
[48] M. Makki, M. Showkatbakhsh, and Y. Song. 'Wallacei' (2019).
[49] Autodesk. 'Dynamo Studio' (2017).
[50] DesignBuilder Software Ltd. 'DesignBuilder' (2012).

support user-interaction while running the optimization process (e.g., Octopus) or traceability between the analysis results and the generated solutions (e.g., Opossum and Octopus) [72], which are critical to address the design constraints not covered by the optimization process (e.g., aesthetic and functionality).

A last important step in most optimization processes involves the selection of the optimization algorithm, which, according to Machairas et al. [272], should consider the type of problem addressed and the algorithm's performance. Currently there are several optimization algorithms available, which are often classified according to the extent of their search (local or global), the determinism of their results (deterministic or stochastic), and the type of information used (derivative-based or derivative-free) [72]. Given their different characteristics, it is only natural that, depending on the problem addressed, some optimization algorithms perform better than others [278]. However, this raises questions regarding the different algorithms' suitability for a given scenario, making their selection a difficult task for which providing a generic rule is often infeasible due to the complexity and diversity of real-world optimization problems. In the literature, there are already some works aiming at systematizing the current variety of optimization algorithms [68, 71, 72, 270, 272, 280, 292–295]; however none present a general solution to guide their selection, only identifying the existing trends in terms of (1) most used search strategies, namely genetic algorithms; (2) most addressed type of problems and performance goals, namely single-objective problems and energy reduction, respectively; and (3) most optimized building element, namely building facades.

5.2.2. Early Design Stages Optimization

Reducing the buildings' environmental impact requires the adoption of passive and active design strategies from early design stages, where major performance improvements can be achieved [296], that consider the multiplicity of requirements affecting their performance [269], such as occupants behavior [297], building shape and orientation [298], envelope transmittance [299], and window type and area [300]. Nevertheless, only a few requirements are often considered at initial design stages, such as aesthetic and functional ones, the others being typically postponed to later stages, where the design idea is already well-established [269, 301, 302]. However, at later design stages, most design changes tend to be difficult, or even impossible, to implement due to the lack of flexibility of most architectural design models [302]. The result is often a hardworking process based on iterative manual remodeling tasks [264] that limit both the efficacy of the optimization strategy and its successful integration in the design process, thus compromising the quality of its results [302].

In the literature, there are studies that focus on anticipating the improvement of design solutions to early design stages to avoid the tiresome and time-consuming remodeling tasks.

Schlueter & Thesseling [303], for instance, proposed a prototypical tool for the BIM context to allow instantaneous energy and exergy calculations and graphical visualizations of their results since early design stages. Petersen & Svendsen [304] presented a method and a tool (iDbuild) to support informed early stage design decisions based on energy and indoor environment performance. Attia et al. [305] presented an energy-oriented tool to support early-stage design decisions based on thermal comfort and energy performance. Lin & Gerber [306] developed the Evolutionary Energy Performance Feedback for Design framework to support early-stage design decisions, which combines geometric exploration and multi-objective optimization processes in the search for improved solutions in terms of energy consumption, cost, and functionality. Konis et al. [307] applied a novel framework (Passive Performance Optimization Framework) to improve daylighting, solar control, and natural ventilation performances at early design stages. Anton & Tănase [264] discussed the integration of both parametric modeling and energy analysis strategies at early design stages. Lin et al. [308] developed a multi-objective optimization engine (MOOSAS) for early design stages that supports the automatic conversion of 3D digital models into analytical ones, real time feedback on energy and daylighting performance analyses results, and interactive building performance optimizations based on user preferences. Finally, Ampanavos & Malkawi [309] introduced a performance-driven design method to assist early-stage form finding optimization processes.

The proposed solutions, however, still present some limitations. The first one is that, most proposals, only interact with one or even none of the design tools architects typically use (for example, while [264, 307] are based on Grasshopper, [304, 308] only interoperate with Sketchup, [305] with EnergyPlus, and [303, 306] with Revit). Another one is the fact that some solutions are based on the visual programming paradigm (e.g., [264, 307]) and thus suffer from its scalability and performance issues. Other limitations are that most solutions either hardly address more than two performance requirements (e.g., [303] only calculates energy consumption, [305] encompasses only thermal comfort and energy performance, and [308] only addresses energy and daylighting performance), support a single or even no optimization strategy (for example, [306, 307] only use Genetic Algorithms and [303, 304] do not provide any specific optimization strategy), or have a narrow scope of application (e.g., [304] only supports rectangular rooms with a single fenestration, [305] targets only the Egyptian context, and [309] only supports a limited range of design variables).

5.2.3. Improving Building Facades

Based on the literature [68, 71, 270, 272, 280, 293–295], building facades are one of the most optimized elements in architecture; a trend that is also visible in several contemporary buildings whose facade design either considered or was guided by its performance. For some authors [40, 310,

311], this trend is explained by this element's importance in the design of buildings, providing them with an architectural identity, while shaping their environmental performance, indoor environmental quality, and structural stability [71].

Within the scope of facade design, Bouchlaghem [312], proposed an integrated computer-based model to design building facades based on their thermal performance; Wang et al. [313] developed a multi-objective optimization model to assist designers in the design of green buildings; Ochoa and Capeluto [311] developed the model *NewFacades* to help designers materialize their ideas based on energy and visual comfort strategies; Gagne and Andersen [314, 315] proposed a tool based on genetic algorithms to guide the design exploration of building facades based on their illuminance and glare levels; Ko et al. [316] developed a methodology to refine faceted building facade shapes according to their direct solar heat gains; Kasinalis et al. [317] proposed a multi-objective optimization method based on genetic algorithms for quantifying the impact that seasonal facade adaptation has on the buildings' performance; Jin and Overend [318] developed a multi-objective optimization tool prototype to search for optimal facade design solutions by considering functional, financial, and environmental constraints; Gamas et al. [319] studied the use of evolutionary multi-objective algorithms to optimize a building envelope in terms of its thermal and daylight performance; Elghandour et al. [203] proposed a performance-oriented approach to improve the daylight performance of building facades; Konis et al. [307] proposed a framework to improve daylighting, thermal, and natural ventilation performance of buildings since early design stages by optimizing their shape, orientation, fenestration configuration, and facade shading elements; Pantazis and Gerber [320, 321] developed an agent-based framework to optimize facade paneling by considering both lighting performance metrics and user daylight preferences since early design stages; and finally, Bertagna et al. [322] proposed a framework to support holistic facade design approaches at conceptual design stages combining shape exploration with structural, daylight, and user-defined criteria.

Despite the extensive literature, most proposals do not directly address facade design geometric exploration processes (for example, in [203, 312, 314–318, 320, 321], it is the user who produces the facade design model *a priori* in either the modeling or simulation tool and, in [311–313], the one who sets the project's input data, e.g., window-to-wall ratio, building orientation, etc.) and those that do have a limited modeling flexibility, only considering the generation of either building shapes, fenestrations, and shading devices of regular sizes and geometries [307, 314, 315] or different structural topologies for building facades [322]. Some also have a limited scope of application (for example, while the proposals [311, 320, 321] target the optimization of office building facades, [318] focuses on the optimization of commercial buildings with glazed facades, and [316] on the massing

process of faceted building facades), or address a single performance requirement, such as thermal [312], lighting [203, 314, 315, 320, 321], or direct solar heat gains [316]. Moreover, the range of optimization strategies allowed is often limited (for examples, [312] only provides Direct Search algorithms and [307, 313–318] Genetic Algorithms), not only narrowing the scope of application of the proposals, but also requiring architects to master an extensive range of tools to address different design problems and to know in advance which strategy best suits each one. Lastly, some of the proposals do not present a Graphical User Interface (GUI) displaying the resulting solutions [311–313, 317, 318], making their use insufficiently intuitive and insufficiently user-friendly, or do not directly communicate with the design tools architects use [311–313, 318]. The existing exceptions [203, 307, 314, 316, 320–322], however, are mostly based on the visual programming paradigm, thus sharing its limitations, particularly scalability [241].

5.3. The Role of Collaboration

Nowadays, environmental and social concerns play an important role in the design of buildings, requiring the integration of analysis and optimization processes from early design stages to ensure the different design requirements are met. This has, in turn, motivated the increasing adoption of CD approaches in the field which, together with the growing complexity of architectural design problems, have been causing several changes in design studios.

The adoption of collaborative design environments merging differently skilled professionals is one such example [323]. It allows design studios to benefit from diverse specialized knowledge in their design practice and thus more successfully respond to the growing design requirements [324]. Perkins+Will, White Architects, Foster+Partners, Woods Bagot, UNStudio, and SHoP are well-known examples of collaborative design practices. In this section, we present a set of collaboratively developed architectural projects that stand out at both representational and technological levels, benefiting from the latest CD methods and manufacturing strategies.

The first, and earliest, example is the Sydney Opera House (Figure 5.2 left) designed by Jørn Utzon. It was one of the first projects to challenge the means of architectural production of its time, explaining the long period between its design (1959) and its construction (1973). At the time, this project involved several experts to realize the unconventional roof shape, being a pioneer in resorting to computers to structurally analyze its different roof shells and guide the assembly of their arches.

Already in the 1990s, the International Terminal Waterloo (Figure 5.2 middle) designed by Nicholas Grimshaw and Partners and completed in 1993, is one of the first projects to technically apply Parametric Design (PD). The arches that originally composed its roof structure had 36 different

possible dimensions but similar geometric principles, allowing the team to create a parametric model based on the arches' underlying geometric rules [129] instead of modeling them separately. In this project, the use of PD enabled the team to save both time and effort in the design process, proving its applicability in a real context, while evidencing its advantages for architectural design.

Project ZED designed by Future Systems in 1995 is another example that stands out due to being one of the first projects to use computational fluid dynamics to guide the facade design process and make the building self-sufficient in terms of energy consumption [129]. The result was a high-performance facade design incorporating photovoltaic cells and optimized in terms of curvature to maximize the channeling of wind towards the building's wind turbines.

Another iconic example from this decade is Guggenheim Museum Bilbao (Figure 5.2 right), designed by Frank Gehry and constructed between 1993 and 1997; one of the most innovative projects of its time in terms of design exploration and materialization. Given the inability of the existing design tools to model its complex facade shape and fabricate its double curved panels, the team used instead an aerospace modeling software, namely CATIA. Using this tool, the team could not only overcome the design limitations found, but also make the project's construction viable, proving that the adoption of both knowledge and tools from other fields can bring advantages to architectural practice.

Figure 5.2. From left to right: Sydney Opera House (©author); the International Terminal Waterloo (©Grimshaw and Partners); Guggenheim Museum Bilbao (©Tony Hisgett).

Already in the 2000s, the Southern Cross Railway Station (Figure 5.3 left) designed by Grimshaw and Partners in association with Jackson Architecture and constructed between 2002 and 2006, is another relevant example resulting from a collaborative design process involving architects, structural engineers, and steel detailers. In this project, computer-generated analyses of both the wind and the natural extraction of stale air were used to guide the design of its roof structure, originating an organic undulating shape that was visually interesting and acted as a sun shading element that simultaneously

extracted stale air from the diesel trains. For its structural design, the team once again used CD, this time to perform iterative analysis evaluations applying different loads, structural elements, and roof configurations, among others, as well as optimize the size and connection complexity of the resulting structure [325].

Figure 5.3. From left to right: Southern Cross Railway Station (©2021 GRIMSHAW); Louvre Abu Dhabi (©author); the dome's cladding structure (©author).

Another relevant example is the Louvre Abu Dhabi (Figure 5.3 middle), designed by Ateliers Jean Nouvel in association with HW Architecture and constructed between 2009 and 2017. This project is characterized by its environmentally informed dome structure, whose design complexity made the team of architects collaborate with other teams of experts, such as Buro Happold, TransSolar, and Gehry Technologies, in a process relying on a web-based central model that allowed the different stakeholders to add their own design constraints and rules and share information in real-time. This collaboration was critical to converge toward a design solution that successfully integrated all design requirements [326], such as aesthetic, creating a visually pleasing covering structure made of different layers of differently scaled and rotated star-shaped elements (Figure 5.3 right); thermal comfort, shading and cooling down the inside spaces; natural lighting, regulating the perforation levels according to the functional area below; and structural, supporting the 165 meter long roof structure [327]. To make its construction viable, the team applied CD strategies to reduce the diversity of structural elements from 2497 to 44 [326], an almost one year long process requiring the use of 23 analysis models[51]; as well as to mass-customize its cladding elements, namely to collaboratively detail them in a parametric model, automatically convert them into construction elements [327], and automatically catalogue and engrave assembly information on them, which was critical for their subsequent installation.

[51] BuroHappold, *Buro Happold shortlisted for two IstructE awards* (2018). https://www.burohappold.com/news/burohappold-shortlisted-two-istructe-awards/ (Retrieved on <January 27th 2022>).

The next example is the Morpheus Hotel (Figure 5.4 left), designed by Zaha Hadid Architects and constructed between 2013 and 2018. The geometric complexity of this project, especially its highly irregular facade, made the use of traditional methods not viable in terms of time, effort, and efficiency, thus motivating the design studio to collaborate with other teams of specialists, e.g., Buro Happold, and to adopt an entirely CD approach combining AD and structural optimization. During this process, the team developed several algorithmic descriptions to connect the AD tool Grasshopper with other tools, such as MIDAS, Robot, and Excel, automating the structural analysis and optimization of the facade design and making its construction feasible. In this project, using AD was advantageous to both design exploration and construction processes [328] due to (1) saving both time and effort; (2) minimizing fabrication errors; (3) reducing manufacturing costs while considering the design intent; and (5) automatically generating accurate technical documentation (e.g., 2D drawings and 3D models) for the ensuing manufacturing and construction stages.

The last example is the Kuwait International Airport, designed by Foster+Partners (Figure 5.4 right) and still under construction[52], which is characterized by its continuous free-form roof structure (with around 320000m^2) performing both structural and aesthetic functions. Given the large scale and geometric complexity of this project, its design involved differently skilled teams of experts, as well as various specialized design methods, software platforms, and fabrication technologies. To ensure the efficiency and flexibility needed to overcome the lack of interoperability between most of the specialized tools used, the design studio adopted an integrated AD workflow combining geometric, performance, and engineering data in a single input source. The result was a central data model developed in Rhinoceros 3D that allowed the team to not only share information more efficiently and accurately, but also collaboratively perform several parametric variations and structural analyses within the short time available for it. It also allowed the team to automate several processes from design to fabrication, such as the detailing of construction elements and the extraction of technical documentation, as well as control the fabrication process by producing, in the end, the building elements in temporary factories near the construction site, which was critical to reach the high-level of precision and fabrication speed needed [329]. Even though the resulting workflow is not yet fully automated, since it still uses handmade drawings occasionally, it nevertheless proves the potential of collaborative CD approaches to deal with unconventional design outcomes and higher levels of information complexity from conceptual design exploration to fabrication [277].

[52] Estimated completion date in 2024.

Figure 5.4. Left: Morpheus Hotel (©Ivan Dupont); right: Kuwait Airport Terminal 2 (©Foster+Partners).

In sum, the selected projects prove, on the one hand, the need to adopt (1) collaborative design environments to better respond to the growing complexity of architectural design and (2) CD approaches to successfully coordinate its multiple constraints and strategies in a single workflow. In most examples, using CD allowed the differently skilled participants to dynamically contribute to the design process, making it easier to handle the complexity of the existing information schemes, as well as realize the resulting unconventional solutions, which challenged the design and manufacturing technologies available at the time.

On the other hand, the projects demonstrate the still existing limitations in collaborative design practices, especially the lack of interoperability between design tools. In some projects this was surpassed by converting the design model into an algorithmic equivalent, integrating both geometric and construction data. In others, this was partially solved through the creation of algorithmic extensions uniformizing the data produced by the different tools, improving the sharing of information between them. However, in current practices the adoption of such scenarios is often not trivial, the first barrier being the lack of programming experience of most architects, which hinders their collaboration in the design process as well as their understanding of the shared data. Another common barrier is the high level of abstraction typical of AD approaches, which makes both the understanding and sharing of design information between the team elements difficult.

Considering the current scenario, the need to increasingly adopt CD approaches and collaborative design teams involving differently skilled professionals is thus evident. However, it requires architects to acquire, at least, a basic understanding of AD to successfully communicate with AD specialists while intervening in the design process. In case they want to participate more actively

in AD-related tasks, architects have to learn new skills, such as programming, which still takes time and effort.

To smooth AD's learning curve and reduce the initial time investment required to learn it, architects should be provided with architectural-oriented methodologies and tools supporting the algorithmic development, analysis, optimization, and fabrication of their solutions. Although this might not be enough to completely reduce the complexity and effort associated with AD, it should nevertheless make AD more accessible to those architects who want to benefit from its advantages in their design practices. By reducing AD's abstraction barrier, while increasing its understandability, it is expected it will reduce the time and effort those architects with some AD experience spend in design development, not only facilitating the integration of multiple types of data and technologies, but also smoothing the transition between design exploration and fabrication stages.

6. MAKING DIGITAL REAL

There are two main concerns that drive architectural practice: the design and the construction of buildings [84]. Originally, the profession of architecture involved both tasks and, thus design decisions were highly dependent on the construction viability of the solutions [330]. It was the architect who made the relationship between design, structure, and materiality, as it is visible in ancient iconic buildings such as Egyptian pyramids, Greek temples, and Gothic cathedrals [331]. The separation of architecture and construction occurred during the Renaissance period, when architects started to differentiate themselves from master builders and craftsmen [330] by focusing more on the ideation and design development processes rather than on solving construction issues [332]. Nevertheless, the feasibility of the solutions never ceased to be a concern for architects, who continued to consider the available materials and construction techniques when conceiving their designs [141].

The desire for unconventional geometries that defied the laws of nature has always been present in architecture. Architects have always ambitioned to design and construct innovative shapes and structures that went beyond what had been done to date. Recent examples prior to the rise of digital tools include the reinforced concrete freeform shapes of the mid-twentieth century. This new material opened up new construction possibilities at the time, allowing the concretization of geometries that were not possible to build until then [333], as is the case of the works of Antoni Gaudí, Felix Candela, Eladio Dieste, Buckminster Fuller, Frei Otto, and Heinz Isler, among others. At the time, these architects/engineers used physical models to explore less conventional shapes and study their corresponding structural behavior [277]: e.g., Antoni Gaudí's system based on the mathematical description of hyperboloids to deal with ruled surfaces [102] and Felix Candela's structural models of hyperbolic paraboloid concrete shells.

With the emergence of digital tools and Computational Design (CD) approaches, architects were provided with higher levels of design freedom and efficiency, facilitating not only the geometric exploration of freeform shapes, but also the study of their structural and constructive viability and their subsequent manufacturing. The works of design studios such as Zaha Hadid Architects, Foster+Partners, Frank Gehry, and UNStudio, among others [277], are some examples of that.

However, the realization of the resulting solutions is often constrained by the construction methods available, which rarely match the flexibility allowed by CD tools. Digital Fabrication (DF) strategies are gradually changing this reality, despite still presenting some limitations [333]. Their integration with CD tools has been promoting a greater design flexibility and fluidity between design and fabrication processes, allowing architects to control the entire design process and thus reducing the distance between design thinking and making [84, 141, 334]. This chapter addresses contemporary fabrication methods, including the recent phenomenon of DF in architecture, and their main challenges and practical applications. Given the scope of this investigation, most examples are related to facade design strategies and solutions.

6.1. DIGITAL FABRICATION

The new digital means enabled the development of fabrication technologies capable of automating manufacturing processes and achieving higher levels of design complexity and accuracy. These technologies, known as DF, are generally based on Computer Numerical Control (CNC) machines to control the manufacturing of different building elements of varying shapes and materials, allowing architects to control the entire fabrication process and thus reach higher levels of precision.

DF techniques allow for the manufacturing of building's elements that otherwise would be unviable to produce [335]. Ideally, these technologies would also enable the conversion of traditional manufacturing processes, where only the mass-production and assembly of standard elements is economically viable [84], into new ones based on mass-customization strategies to produce multiple non-standard elements at affordable costs [336]. This scenario, however, remains a challenge in architecture due to the still existing limitations of the available fabrication technologies, which include their cost, machining time, scale and material limitations, material waste, and special spatial conditions, among others. Nevertheless, their gradual cost decrease is motivating their growing use in architecture [337]. Finally, DF make it possible to develop architectural projects in an entirely digital manner, where design data directly flows from design development stages to manufacturing and construction ones [84].

Currently, there is a wide variety of DF techniques available suiting architectural practices. In general, they vary in terms of (1) process used to shape the elements, which can be based on adding, removing, cutting, or deforming materials, among others; (2) materials supported, which include plastic, glass, concrete-based materials, etc.; (3) suitability to produce certain shapes and scales of elements; and (4) surface finishing allowed, which can be smooth, textured, printed, perforated, etc.

Some authors categorize the existing DF strategies in three groups, namely *additive*, *subtractive*, and *formative* [335, 337–340], whereas others classify them into four groups, the previous ones plus *cutting* [84, 341, 342], or even in five groups, the first three ones plus *joining* and *robotic* [343]. This research considers the three perspectives, which are further detailed in the following paragraphs.

Additive processes add layers of material to produce the desired shape [335]. These techniques are based on the translation of digital information into a sequence of two-dimensional layers [84]. 3D printing is the most popular additive process in architecture but there are other techniques available, such as stereolithography, fused deposition modeling, laser sintering, and digital light processing [335, 340]. Additive methods have the advantage of directly converting the digital model into a physical element without requiring any additional device. They also support geometric freedom, producing unique elements in a viable way [340]. Moreover, the available machines are often silent, produce reduced material waste, and do not require programming expertise [84]. Still, the production of large-scale elements is still an issue [337], as also is the resulting surface quality and the large production times [340].

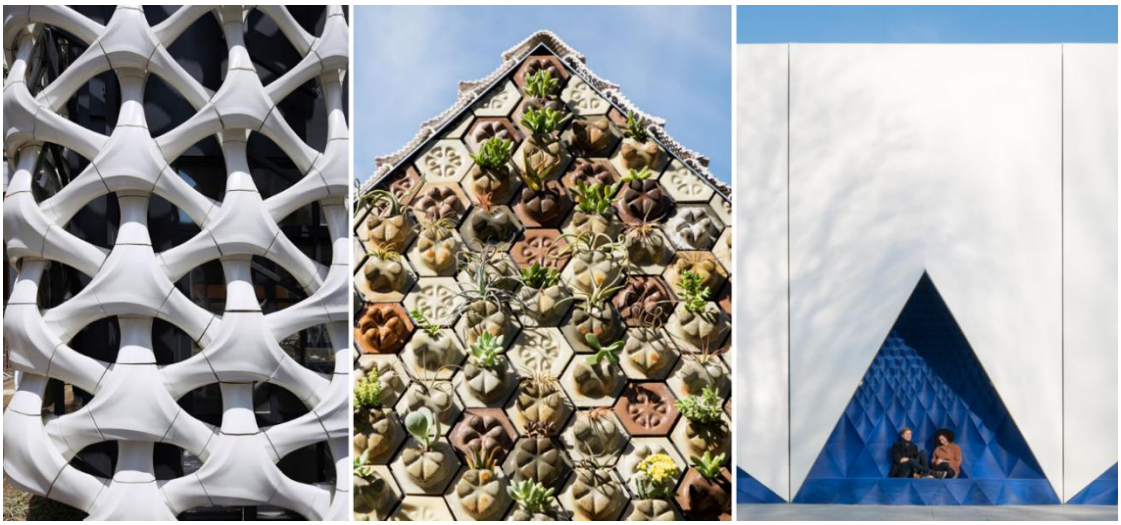


Figure 6.1. Additive manufacturing examples (from left to right): Arachne 3D printed facade by Lei Yu (©Architizer); House of 3D Printed Curiosities by Emerging Objects, 2018 (©Matthew Millman); EU Building 3D printed facade by DUS Architects, 2015 (©Ossip van Duivenbode).

In architecture, 3D printing has been often applied in the production of less conventional facade elements, some examples including the entirely 3D printed facade of the Arachne project in China (Figure 6.1 left); the facade of the Cabin of 3D Printed Curiosities in California (Figure 6.1 middle) made of hundreds of 3D printed ceramic tiles of different geometries; and the facade of the Europe

Building in Amsterdam (Figure 6.1 right) composed by large-scale unique 3D printed elements made of bioplastic and colored concrete[53].

Subtractive processes use electro-, chemically, or mechanically reduced techniques to remove or separate particles of raw material from an existing solid [335, 342] to achieve the desired shape [337]. CNC milling and routing processes are the most commonly applied techniques [337]. When compared with additive processes, the available subtractive technologies present several advantages in terms of (1) elements size, allowing producing from smaller to larger scale elements; (2) material diversity, enabling the use of a wider variety of materials; (3) precision, producing elements with finer details; and (4) production efficiency, requiring less time and material [84]. Nevertheless, these processes tend to produce considerable material waste [335].

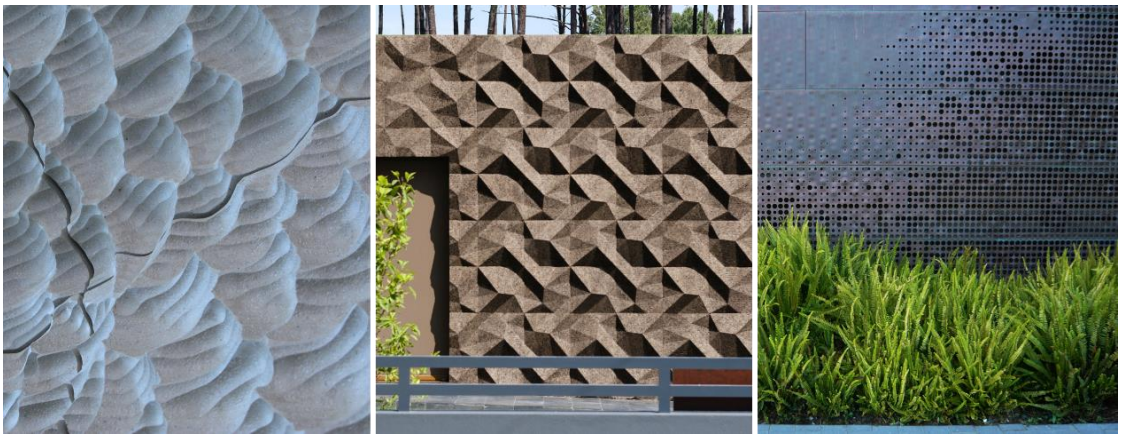


Figure 6.2. Subtractive manufacturing examples (from left to right): Elbphilharmonie's acoustic panels in Hamburg by Herzog & de Meuron, 2017 (©ONE TO ONE); CNC milled cork facade designed by GenCork, 2019 (©GenCork); De Young Museum by Herzog & de Meuron, 2005 (©david basulto via flickr).

In architectural practice, CNC milling, and routing (with 3 to 5 axis) have been applied in the fabrication of building elements in a process similar to carving, i.e., by removing material from a volume. The manufacturing of the acoustic panels of the Elbphilharmonie in Hamburg (Figure 6.2 left) and the cork facade panels of a house in Aroeira, Portugal (Figure 6.2 middle), are two examples of that. These technologies have also been used to produce either perforated or bumped/textured facade elements, such as the sheet facade panels of *de Young* Museum in San Francisco (Figure 6.2 right); or customized molds, as it happened in the Neuer Zollhof office buildings in Düsseldorf (Figure 6.3 left), whose facade panels were manufactured with 355 different CNC milled molds, and in the

[53] DUS. *Europe Building. House DUS* (2016). https://houseofdus.com/work/#project-europe-building (Retrieved on <July 12th 2022>)

MaoHaus facade in Beijing (Figure 6.3 right), which is made of self-supporting ultra-high performance concrete panels casted from CNC milled molds[54].

Figure 6.3. CNC milled molds examples (from left to right): the Neuer Zollhof office buildings in Düsseldorf by Frank Gehry, 1999 (©Mirco Wilhelm via flickr); MaoHaus facade by AntiStatics Architecture, 2017 (©Xia Zhi).

Cutting processes, also known as two-dimensional fabrication, are based on a two-axis motion of a cutting head to extract two-dimensional planar elements from surfaces or solids [335]. These processes follow a set of instructions provided by the digital model to produce flat components with the desired shape [337], often relying on laser-beam, plasma-arc, or waterjet technology. Cutting is a very popular and widely used strategy, probably the most used one [84, 342], and it has been frequently applied in the manufacturing of facade panels with complex patterns. These methods have the advantage of being precise, cheap, and quick, but are limited in terms of type and thickness of the material they can cut, while requiring the use of different technologies accordingly [84, 342]. Examples of cutting applications include the ceiling of the Trumpf Campus Gatehouse in Stuttgart [344] (Figure 6.4 left), the facade panels of the Megalithic Museum in Mora, Portugal (Figure 6.4 middle), and the facade of the Formstelle building in Töging am Inn (Figure 6.4 right).

[54] Say, Asli. *The MaoHaus by AntiStatics Architecture* (2018). https://parametric-architecture.com/the-maohaus-by-antistatics-architecture/ (Retrieved on <July 12th 2022>)

Figure 6.4. Cutting manufacturing examples (from left to right): Trumpf Campus Gatehouse by Barkow Leibinger Architects, 2009 (©David Franck); Megalithic Museum by CVDB Architects and P-06 Atelier, 2016 (©Fernando Guerra | FG+SG); Formstelle by Format Elf Architekten, 2013 (©Format Elf Architekten).

Formative processes use mechanical forces to deform materials into the desired shape [337]. These methods often resort to heating to make the material adapt to the new geometry and then to cooling to keep the new geometry stable [84]. CNC folding, CNC bending, CNC punching, hydro morphing, and welding are some examples [335]. In architecture, these methods have been mostly applied in the manufacturing of geometrically complex metal facade panels. Nevertheless, the existing methods are still expensive due to the price of both the machines and material used [335]. Architectural examples include the Prism Gallery in Los Angeles (Figure 6.5 top-left), whose facade panels were shaped through heat-forming techniques [345]; the facade of the Holt Renfrew flagship store in Vancouver (Figure 6.5 top-right), which is composed by heat-slumped glass panels that create a specular, translucent visual effect [346]; the undulating glass panels of the Elbphilharmonie in Hamburg (Figure 6.5 bottom-left), whose free form shape results from heat-bending processes; and the metal facade panels of the Experience Music Project in Seattle (Figure 6.5 bottom-right), which were produced through cold bending techniques [336].

Lastly, Robotic assembly processes involve the use of robotic arms or drones to accurately place elements in layers. Some of these methods make it possible to reduce or even eliminate the lack of accuracy typical of manual assembly processes, allowing a complete correspondence between the intended design and its final product [339]. Architectural examples resulting from robotic assembly strategies include the brick facade of the Chi She Gallery in Shanghai (Figure 6.6 right), which was assembled with the help of a robotic arm controlling the precise position of each stacked brick to obtain the desired visual effect; and the facade of the Winery Gantenbein in

Switzerland (Figure 6.6 left), which also used a robotic production method to accurately control the stacking angle of each brick and create the intended pictorial effect.

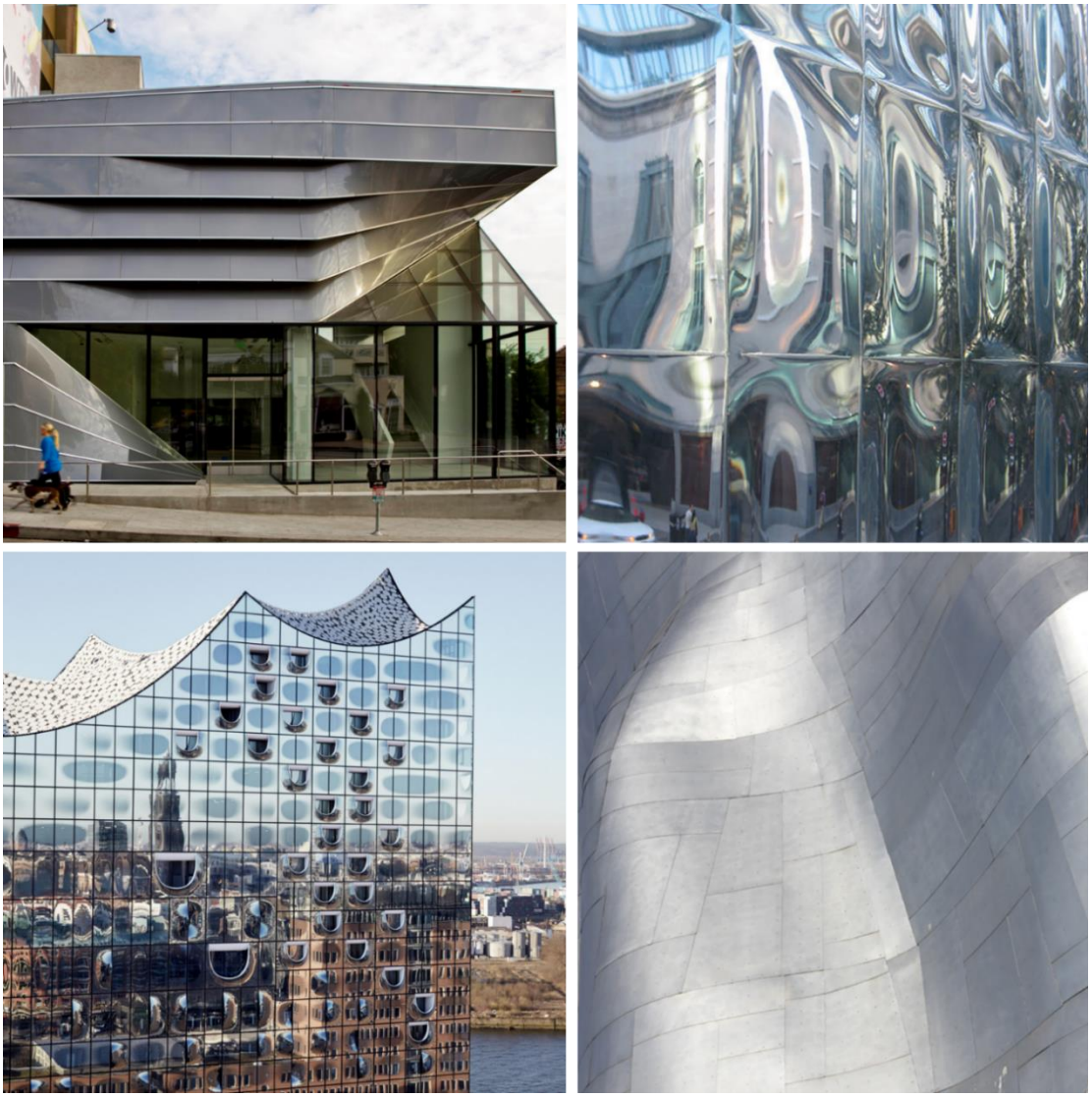


Figure 6.5. Formative manufacturing examples (from left to right): Prism Gallery by P-A-T-T-E-R-N-S, 2010 (©P-A-T-T-E-R-N-S); Holt Renfrew flagship store in Vancouver by Janson Goldstein Architects, 2008 (©Marc Simmons / Front Inc); Elbphilharmonie Hamburg by Herzog & de Meuron, 2017 (©Maxim Schulz); Experience Music Project by Frank Gehry, 2000 (©David via Blogger "Doodles Designs and Daydreams").

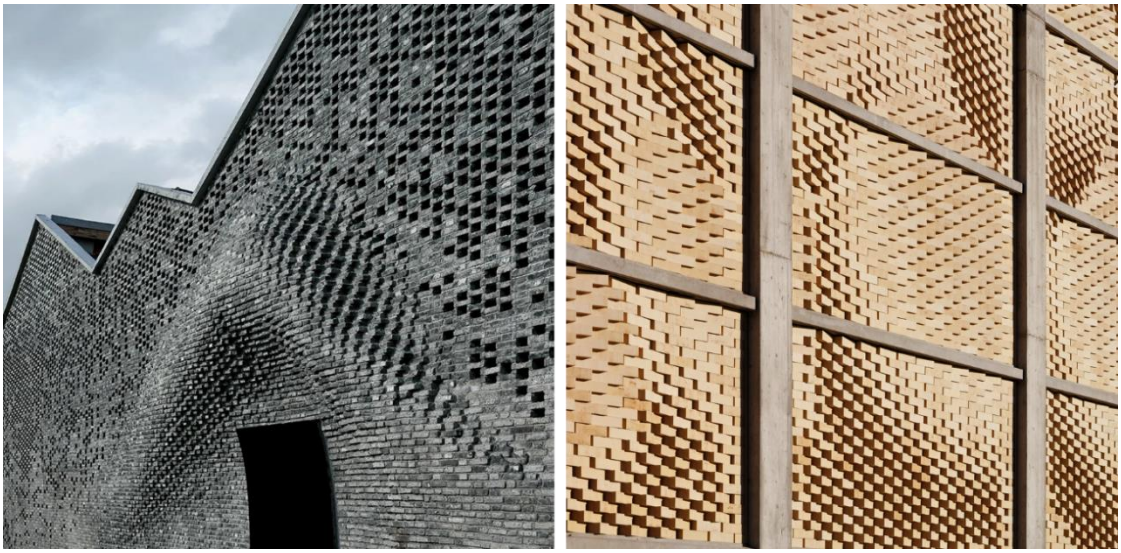


Figure 6.6. Robotic manufacturing examples (from left to right): Chi She Gallery by Archi-Union Architects, 2016 (©shengliang su); Winery Gantenbein by Bearth & Deplazes Architekten + Gramazio & Kohler, 2006 (©Ralph Feiner).

6.2. Design Strategies for Digital Fabrication

The way the different DF technologies are used in architecture, as well as the reason why they are used in a specific scenario, differs from case to case. It depends, among others, on the material used, the scale and shape of the elements to produce, and on the type of surface finishing and intended geometric effect. Based on this, Iwamoto [161] and Dunn [84] presented similar classifications for design strategies to apply DF, organizing them into five categories: *sectioning*, *tessellating/tiling*, *folding*, *contouring*, and *forming*.

Figure 6.7. Examples of sectioning (from left to right): Serpentine Pavilion 2005 by Álvaro Siza and Eduardo Souto Moura (©Sylvain Deleu); Damiani Holz & KO Office extension by MoDus Architects, 2013 (©MoDusArchitects); Metropol Parasol by Jürgen Mayer H. Architects, 2011 (©Javier Orive).

The first one, sectioning, is based on the use of a series of profiles to create either a surface or a structure. Despite its recency in architecture, this technique has been long used in other fields like shipbuilding and airplane industry [84]. The CNC milled waffle structure of the Serpentine Pavilion 2005 (Figure 6.7 left), the facade of the Damiani Holz & KO Office extension in Brixen (Figure 6.7 middle), and the Metropol Parasol in Sevilla (Figure 6.7 right) are some examples.

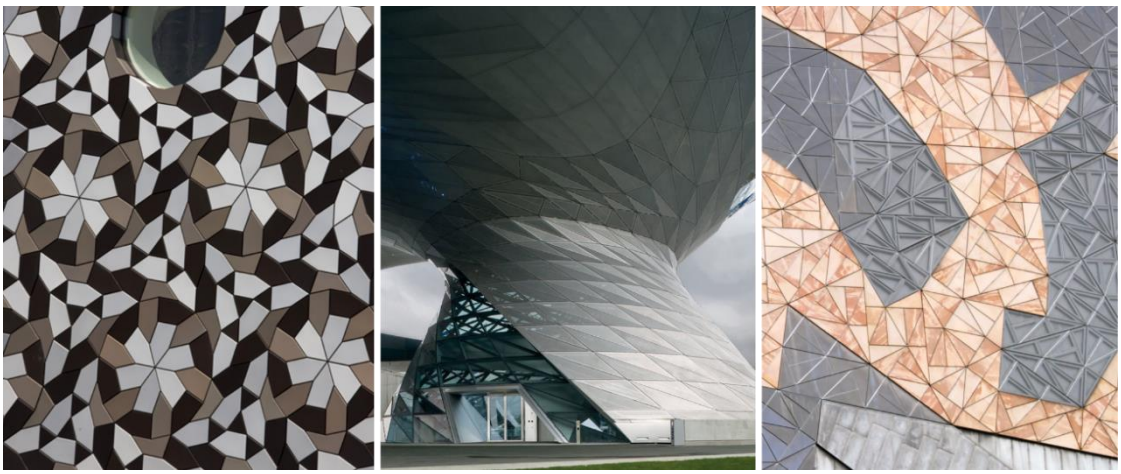


Figure 6.8. Examples of tessellating (from left to right): Ravensbourne College by Foreign Office Architects, 2010 (©Morley von Sternberg); BMW Welt building by Coop Himmelb(l)au, 2007 (©Duccio Malagamba); Federation Square by LAB Architecture Studio, 2002 (©Travis via flickr).

The second category, tessellating, results from the perfect fit of several pieces, creating a smooth surface without gaps. This technique has long been used in architecture, under the name of tile patterns or tiling: the ancient Rome mosaics and the Islamic tiles are two examples. Contemporary examples include (1) the use of tiles of different shapes and colors to produce complex facade patterns, e.g., Ravensbourne College tile facade in London (Figure 6.8 left); (2) the application of mass-produced unique panels of multiple sizes and curvatures, e.g., BMW Welt curved facade panels in Munich (Figure 6.8 middle); or (3) the composition of different facade panels to obtain an intricate geometric pattern, e.g., Federation Square triangular facade panels in Melbourne (Figure 6.8 right).

Regarding the folding category, it involves the conversion of flat surfaces into three-dimensional ones for geometric, structural, and aesthetical purposes [161]. This strategy is commonly applied in architecture because it is very effective and economic and allows for continuous and smooth surface finishing [84]. Architectural examples of folding include the facade of the National Aquatics Center in Beijing (Figure 6.9 left), whose translucent bubble-like effect results from a composition of two inflated layers of plastic film with different cellular organizations [347]; the facade of the Allianz Arena in Munich (Figure 6.9 middle), whose three-dimensional diamond-shape

ethylene tetrafluoroethylene cushions were produced through the injection of air pressure inside them [14]; and the facade the King Fahad National Library in Riyadh (Figure 6.9 right), which consists in a set of fabric membranes whose shape is controlled by a three-dimensional steel cable structure [337].

Figure 6.9. Examples of folding (from left to right): National Aquatics Center designed by PTW Architects, 2008 (©William via flickr); Allianz Arena by Herzog & de Meuron, 2005 (©Ulrich Rossman); King Fahad National Library by Gerber Architekten, 2013 (©Christian Richters).

The next category, contouring, involves the removal of successive layers of material from a flat surface to create a three-dimensional effect. This technique is similar to the long-established carving technique but, instead of using manual processes, it uses digitally controlled ones that are more precise, efficient, and capable of producing highly complex patterns [84]. Nevertheless, when compared to other techniques, these processes require larger amounts of time and often produce more material waste. Architectural applications of contouring include the Bone Wall, which is composed by CNC milled high-density foam cells (Figure 6.10 left); GenCork's wall panels made of CNC milled cork creating different geometric patterns (Figure 6.10 middle); and Anoma's wall prototypes made of CNC milled stone panels with nature-inspired geometric patterns (Figure 6.10 right).

Lastly, forming is an economic and widely applied process that uses molds to mass produce elements. This technique has been long used in architecture, mainly in the mass-production of facade panels, whose application typically resulted in conventional solutions with repetitive patterns. More recently, DF technologies brought new possibilities to conceive customized molds that allow producing less conventional solutions; a strategy widely applied in the production of facade elements.

Figure 6.10. Examples of contouring (from left to right): Bone Wall by Urban A&O, 2006 (©Joe McDonald/Urban A&O Architecture LLC); interior cork panel at Cork Museum WOW in Oporto, Portugal (©GenCork); stone facade panels (©Anoma).

Architectural applications of forming include the Deep Facade project, designed by a group of students at ETH Zurich in 2018, who used 3D printed molds to create a metal facade with a highly complex geometric pattern (Figure 6.11 top-left); the facade of the IBS Building at Minho University in Guimarães (Figure 6.11 top-middle), whose prefabricated cement-based panels were shaped with different customized molds; the three-dimensional facade of the San Francisco Museum of Modern Art extension (SFMoMA), which is composed by unique fiber-reinforced polymer panels produced with personalized CNC milled molds (Figure 6.11 top-right); the facade of the Filigrane building in Tourcoing (Figure 6.11 bottom-left), whose concrete panels inspired by the architectural ornamentation technique *guilloche*[55] were produced with customized CNC milled molds; the facade of the Palace of Justice in Cordoba (Figure 6.11 bottom-middle) whose panels were produced with custom-made CNC milled molds; and, lastly, the copper-cladded structure of the Louisiana State Museum and Sports Hall of Fame (Figure 6.11 bottom-right), which is composed by more than 1000 unique cast-stone panels produced with robotically fabricated molds [348].

[55] Home World design. *The Filigrane Project / D'Houndt + Bajart Architects & Associès*. https://homeworlddesign.com/filigrane-project-dhoundt-bajart-architects/ (Retrieved on <July 12th 2022>)

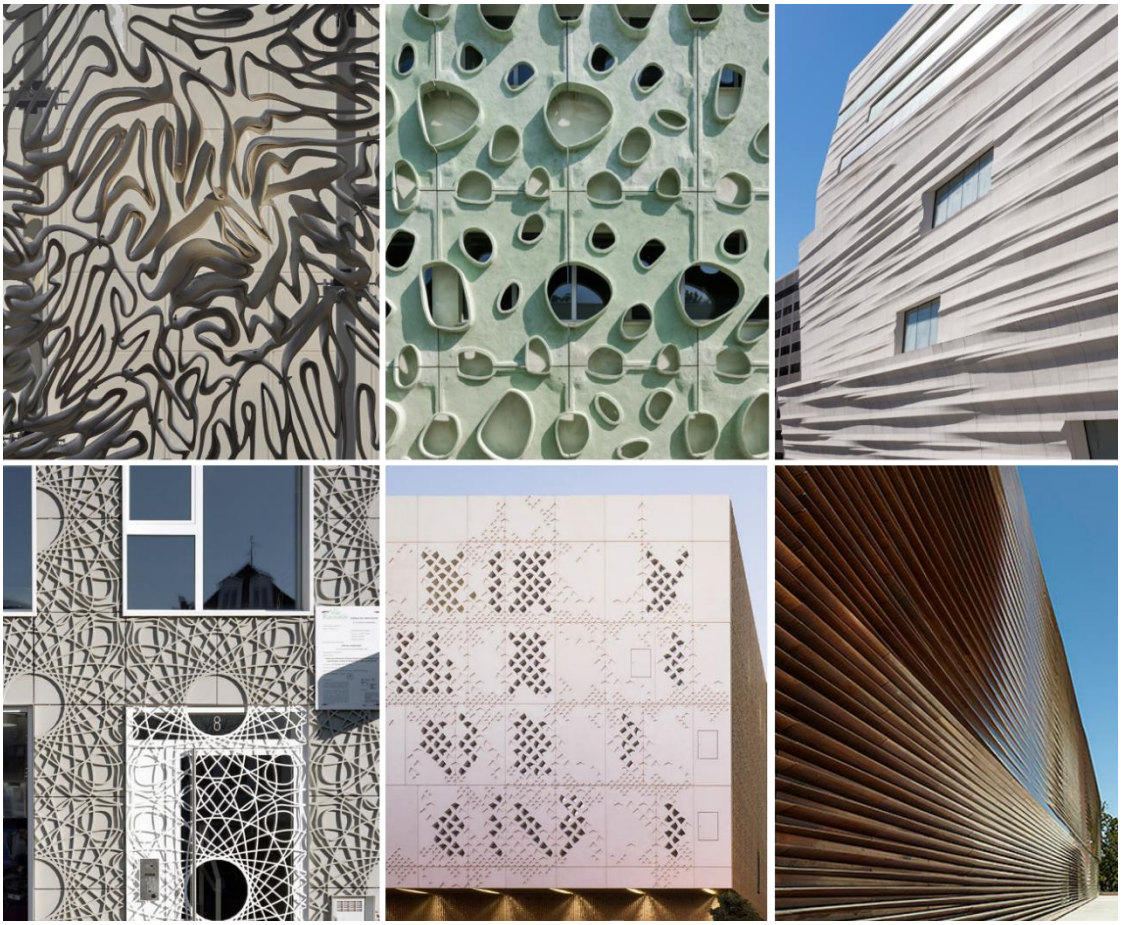


Figure 6.11. Examples of forming strategies (from top-left to bottom-right: Deep Facade project, 2018 (©Jetana (Jet) Ruangjun); IBS Building by Claudio Vilarinho, 2015 (©João Morgado); SFMoMA extension by Snøhetta, 2016 (©Henrik Kam); Filigrane building by D'HOUNDT+BAJART architects & associates, 2017 (©Maxime Delvaux); Palacio of Justice in Cordoba by Mecanoo, 2017 (©Fernando Alda); Louisiana State Museum and Sports Hall of Fame by Trahan Architects, 2012 (©Tim Hursley).

6.3. Geometric Optimization Strategies

In the last decades, the design freedom allowed by most CD technologies has been motivating the design of more complex freeform shapes [69], as well as the creation of intricate facade design patterns. Unfortunately, their production is often challenging and expensive and, in many cases, the complexity achieved can be hardly conceived through traditional construction methods [331, 335]. As a result, architects have to spend a lot of time and effort in solving manufacturing and economic issues [331] and it is often the case that their creative intent is neglected in favor to these.

Despite the currently available mass-production techniques capable of producing non-conventional elements at low cost, none of them is entirely suitable to deal with the geometric diversity of architectural design, which usually requires the manufacturing of hundreds or thousands

of non-standard elements [336] that are often project-specific. To make the construction of free-form shapes and complex facade patterns possible, architects have been increasingly adopting geometric optimization techniques [349] in their design practice. These strategies allow them to gain more insight and control over the designed solutions [69], facilitating the latter's gradual adaptation until reaching the desired feasibility [331]. In architecture, popular examples of geometric optimization strategies include *design rationalization* and *surface paneling*.

6.3.1. Design Rationalization

Design rationalization is a type of geometric optimization that focuses on subtly adjusting the building elements that are expensive to produce until meeting the desired feasibility, without compromising the design intent [168, 336]. The Sydney Opera House (1973) is one of the first examples to combine design rationalization and computational technologies to gradually change the original roof shape until its construction became viable [331]. Still, it was only during the 90s that this combination gained popularity, mainly with the works of Gehry Partners, which were characterized by complex free-form shapes [331]: the Guggenheim Museum Bilbao, for instance, was one of the first buildings to have a free-form shape made of single-curved panels that could be unfolded without stretching or tearing [168].

Based on the literature [69, 331, 350–352], design rationalization can vary in terms of temporal application in the design process and target of the rationalization process. Regarding the former, it can be classified in (1) pre-rationalization, when it is conducted before the design development stage; (2) co-rationalization, when it is used during the design development stage; and (3) post-rationalization, when it is applied after the design development stage. According to Austern et al. [331], pre- and post-rationalization are the most and least addressed strategies in the academia, and co- and post-rationalization the most and least applied ones in architectural practice.

Figure 6.12. Pre-rationalization examples (from top-left to bottom-right): the International Terminal Waterloo by Grimshaw Architects, 1993 (©Grimshaw and Partners); the Eden Project by Grimshaw Architects and Anthony Hunt Associate, 2001 (©Grimshaw and Partners); London City Hall by Foster+Partners, 2002 (©author); the 30 St Mary Axe building by Foster+Partners, 2004 (©Foster+Partners).

When adopting pre-rationalization, the design process is constrained by construction requirements since the beginning, which means the design exploration is controlled by such constraints. Examples of pre-rationalization include the International Terminal Waterloo Station (Figure 6.12 top-left), the Eden Project (Figure 6.12 top-right), the London City Hall (Figure 6.12 bottom-left), and the 30 St.Mary Axe (Figure 6.12 bottom-right). When using co-rationalization, the design development process often combines principles from both pre- and post-rationalization methods [331], thus coordinating material properties, fabrication and assembly constraints, and aesthetic principles in a flexible way [353]. Examples of its application include the Galaxy SOHO (Figure 6.13 top-left), the Wangjing SOHO (Figure 6.13 top-right), the Bangkok Central Embassy (Figure 6.13 bottom-left), and the Leadenhall Building (Figure 6.13 bottom-right). Lastly, when adopting post-rationalization, the construction requirements do not control the design development process and are instead fitted onto the design's

final solution [69]. This strategy was adopted, for instance, in the Guggenheim Museum Bilbao (Figure 6.14 top-left), the Walt Disney Concert Hall (Figure 6.14 top-right), the Nordpark Railway Stations (Figure 6.14 bottom-left), and the Heydar Aliyev Center (Figure 6.14 bottom-right).

Figure 6.13. Co-rationalization examples (from top-left to bottom right): Galaxy SOHO by Zaha Hadid Architects, 2012 (©Iwan Baan); Wangjing SOHO by Zaha Hadid Architects, 2014 (©Virgile Simon Bertrand); Central Embassy in Bangkok by AL_A, 2017 (©Hufton+Crow); The Leadenhall Building by Rogers Stirk Harbour+Partners, 2014 (©Richard Bryant).

Figure 6.14. Post-rationalization examples (from top-left to bottom-right): Guggenheim Bilbao Museum by Gehry Partners, 1997 (©Gehry Partners); Disney Concert Hall by Gehry Partners, 2003 (©Gehry Partners); Nordpark Railway Stations by Zaha Hadid Architects, 2007 (©Hafelekar via Wikimedia); Heydar Aliyev Center by Zaha Hadid Architects, 2012 (©Hufton+Crow).

Regarding the target of design rationalization processes, these methods have been mostly applied in the design of building facades to make their manufacturing viable: to either (1) reduce the number of different facade elements, e.g., window frames, facade panels, wall tiles, and shading devices, etc., without neglecting the initial design intent, (2) search for geometric similarities that potentially facilitate their production, or (3) simplify their free-form surfaces into feasible solutions.

Examples of the first scenario include the facade of Museo Soumaya, in which design rationalization was used to minimize the number of different hexagonal aluminum panels (Figure 6.15 top-left); the facade of the Federation Square in Melbourne (Figure 6.8 right), to reduce the

geometric complexity of the predefined basic units composing it [354]; and the facade of MAAT Museum in Lisbon (Figure 6.15 top-right), to simplify its hexagonal geometric pattern.[56]

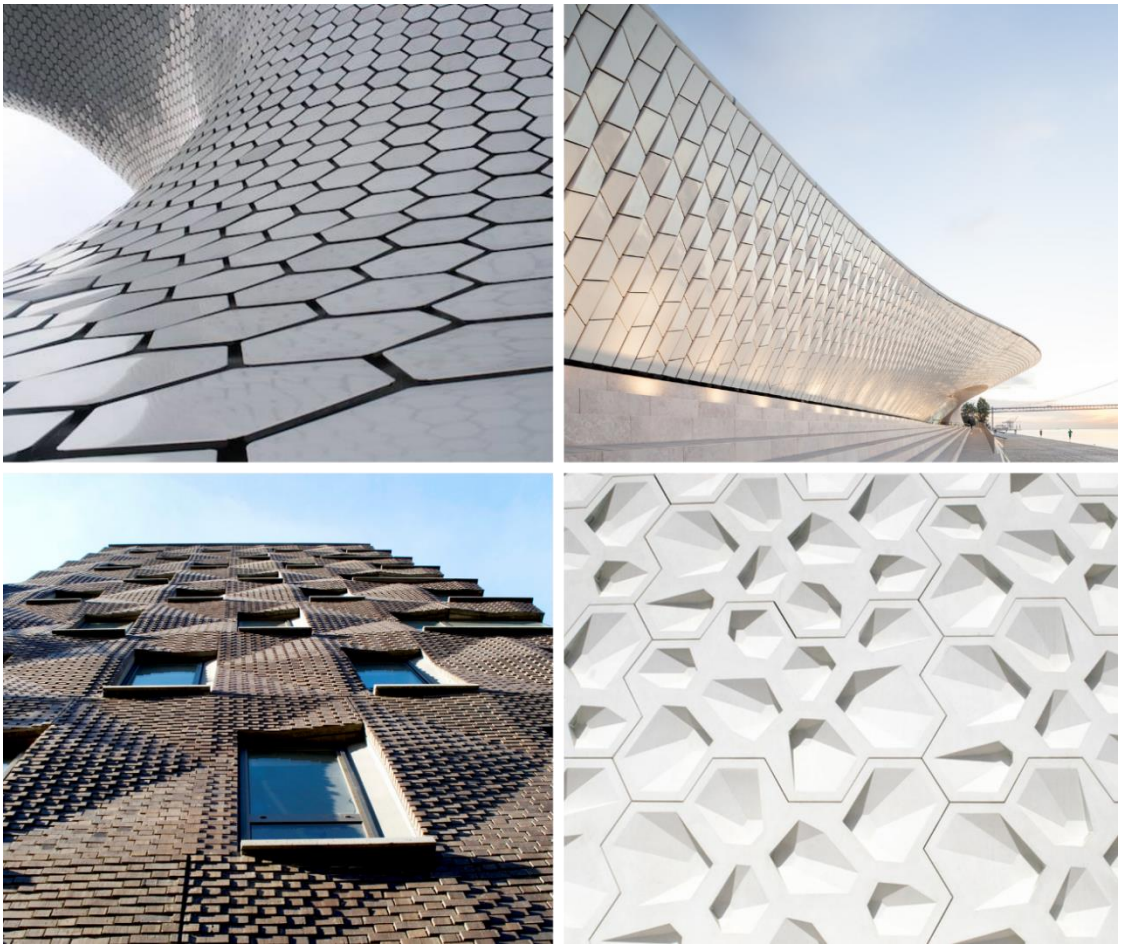


Figure 6.15. Geometric simplification examples (from top-left to bottom-right): Museo Soumaya by Fernando Romero, 2011 (©Geometrica, Inc.); MAAT Museum by AL_A, 2016 (©Francisco Nogueira); 290 Mulberry Street building by SHoP Architects, 2013 (©SHoP Architects); Centre for Contemporary Art by Nieto Sobejano Arquitectos, 2013 (©realites: united).

The second scenario occurred, for instance, in the manufacturing of the brick facade panels of the 290 Mulberry Street building (Figure 6.15 bottom-left) [355] and in the production of the hexagonal facade tiles of the Centre for Contemporary Art in Córdoba (Figure 6.15 bottom-right) [356].

Regarding the third scenario, different strategies have been adopted to make free-form surfaces feasible [336]. One of them consists in the simplification of the original shape into one made of planar panels only approximating it, as it happens in Renzo Piano's Peek & Cloppenburg

[56] CeramicArchitectures. *Museum Art Architecture and Technology*. https://www.ceramicarchitectures.com/obras/museum-art-architecture-technology/ (Retrieved on <July 12th 2022>)

Department Store in Cologne (2005). Another one is the production of a faithful, smoother approximation of the shape by using smaller discrete elements of different curvatures, as it happens in Zaha Hadid Architects' Nordpark Railway Stations in Innsbruck (2007) [157]. In either case, the resulting solutions often resort to different surface paneling techniques.

6.3.2. SURFACE PANELING

Panelization, or *paneling*, is a geometric optimization strategy that focuses on dividing a large surface into smaller panels of constructable size and acceptable cost. This strategy involves two dependent tasks: the segmentation of the original shape into smaller pieces and the approximation of each smaller piece into a shape that can be manufactured at a reasonable cost, while preserving the design intent [171, 336, 352].

Dividing a surface into smaller planar surfaces of different polygonal shapes, like triangular, quadrilateral, and polygonal, is the cheapest paneling strategy. This technique has been long used in architecture and the dome structures designed before the 20th century are an example of its application: their construction typically used flat quadrilateral panels due to aesthetical and economic reasons [352]. Triangular panels emerged in the late 1920s thanks to the Carl Zeiss Company and were later adopted by several architects due to having more structural stability and allowing for greater geometric flexibility[57]. Their production, however, often creates more waste, and each vertex is typically more complex than those of quadrilateral panels due to connecting six edges instead of four. Moreover, since more panels are needed for covering the same surface area, the resulting structure is also usually heavier [352]. Even so, triangular panels are the most popular paneling solution in contemporary architecture, some examples being the roof of the Great Court at British Museum (Figure 6.16 left), the facade of the 30 St. Mary Axe building, both in London (Figure 6.12 right), the roof of Zlote Tarasy in Warszawa (Figure 6.16 mid-left), the roof of the Islamic Art Exhibition in the Louvre Museum in Paris (Figure 6.16 mid-right), and the roof of the Incheon International Airport (Figure 6.16 right).

[57] It is easier to change the vertices of triangular panels to accommodate different design requirements and keep their faces planar.

Figure 6.16. Triangular paneling examples (from left to right): British Museum Great Court roof by Foster+Partners, 2000 (©author); Zlote Tarasy roof structure by The Jerde Partnership, 2007 (©JERDE); Islamic Art Exhibition roof by Mario Bellini Architects and Rudy Ricciotti, 2012 (©Raffaele Cipolletta); Incheon International Airport by HEERIM Architects & Planners, MooYoung Architects and Gensler Architects, 2018 (©Gensler).

Regarding quadrilateral panels, these are used, for instance, in the facade of the Jerusalem Museum of Tolerance (Figure 6.17 left) and in the roof of the Tokyo Midtown Plaza (Figure 6.17 middle). Finally, hexagonal panels were applied in the bubbly domes of the Eden Project (Figure 6.12 middle), the curvilinear facade of the Museo Soumaya (Figure 6.15 top-left), and the double-dome of the Landesgartenschau Exhibition Hall (Figure 6.17 right).

Figure 6.17. Quadrangular and hexagonal paneling examples (from left to right): Museum of Tolerance Jerusalem by Chyutin Architects (©ChyutinArchitects); Tokyo Midtown Plaza roof by Buro Happold, 2005 (©SOM); the Landesgartenschau Exhibition Hall by ICD/ITKE/IIGS University of Stuttgart, 2014 (©Roland Halbe).

Another paneling strategy used in architecture involves the division of the original surface into smoothly bent stripes, also known as single-curved panels or developable surfaces, that can be

produced by simply bending a flat piece of sheet metal; a technique that has been showing gradual improvements over time due to the availability of more advanced tools [352]. Despite being more often applied to metal surfaces, this strategy can be used with other materials, such as glass or wood, albeit often leading to higher fabrication costs and requiring the use of specialized machinery [352]. Examples of glass single-curved panels include the IAC building in New York (Figure 6.18 left) and Foundation Louis Vuitton in Paris (Figure 6.18 middle). In general, these panels are more affordable than double-curved ones but also often lead to surfaces with less precision [336]. Walt Disney Concert Hall (Figure 6.14 top-right) is one such example, where the originally double-curved facade panels were converted into flat and single-curved ones to make their manufacturing viable, resulting in a surface presenting a visible discontinuity between panels. Another example is the Mercedes-Benz Museum (Figure 6.18 right) where, contrarily to the initial design intent, only a few facade areas are composed by double-curved panels, the remaining ones presenting a lower precision due to the discontinuity of the curvatures [336].

Figure 6.18. Single-curved panels examples (from left to right): IAC building in New York designed by Frank Gehry, 2007 (©Chuck Choi/Arcaid); Foundation Louis Vuitton in Paris by Frank Gehry, 2014 (©ToddEberle); Mercedes-Benz Museum by UNStudio, 2006 (©UNStudio).

One last paneling strategy focuses on dividing the surface into double-curved perfectly fitting panels, resulting in smooth curved surfaces with a high finishing quality. This technique has been applied on concrete surfaces, e.g., the Rolex Learning Center at EPFL in Lausanne (Figure 6.19 top-left) and the Meiso No Mori Municipal Funeral Hall in Japan (Figure 6.19 top-middle); metal facades, e.g., the Dongdaemun Design Plaza building in Seoul (Figure 6.19 top-right); acrylic glass facades, e.g., the BMW Bubble project in Frankfurt (Figure 6.19 bottom-left); wood surfaces, e.g., the Kamppi Chapel in Helsinki (Figure 6.19 bottom-middle); and, lastly, fiber-reinforced plastic facades, e.g., the temporary building Chanel Mobile Art (Figure 6.19 bottom-right).

Figure 6.19. Double-curved panels examples (from top-left to bottom-right): Rolex Learning Center by SANAA, 2010 (©Davide Galli); Meiso No Mori Municipal Funeral Hall by Toyo Ito, 2006 (©Dennis Gilstad/Construction Matters); Dongdaemun Design Plaza by Zaha Hadid Architects, 2014 (©Virgile Simon Bertrand); BMW Bubble building by Franken Architekten GMBH, 1999 (©Fritz Busam); Kamppi Chapel by K2S Architects, 2012; Chanel Mobile Art by Zaha Hadid Architects, 2008-2010 (©Roland Halbe).

Compared with the previous paneling strategies, this is the most precise but also the most expensive one as it often requires the production of several customized molds [171]. Based on the idea that the larger the number of molds needed, the higher the manufacturing cost, several strategies based on mold reuse have been adopted to reduce the solutions' final cost [171]. One of them is the identification of surface areas with similar geometries that can be produced with the same molds; a strategy adopted in, for instance, the L'Atoll retail park in Angers (Figure 6.20 left) and the Investcorp Building for the Middle East Centre at Oxford University (Figure 6.20 middle). Another one is the adaptation of the original surface shape to decrease its level of geometric variation; a technique applied, for instance, in the roof structure of the Sydney Opera House (Figure 6.20 right).

Despite the existing technologies and architectural examples, the production of large-scale free-form structures is still regarded as a challenging task [169–174]. Moreover, the existing literature, as well as practical examples, mostly focus on simple patterning techniques, i.e., the use of triangular, quadrangular, and hexagonal panels, rarely considering other more complex geometric patterns, which lately have been widely applied in building envelope design.

Figure 6.20. Surface simplification examples (from left to right): L'Atoll retail park by Antonio Virga Architecte+AAVP Architecture, 2012 (©Luc Boegly); Investcorp Building by Zaha Hadid Architects, 2015 (©Luke Hayes); Sydney Opera House by Jørn Utzon, 1973 (©Roybb95 via Wikimedia Commons).

6.4. MATERIALIZING ARCHITECTURAL CREATIVITY

This section further develops some of the previous examples, describing the adopted rationalization strategies, as well as the fabrication means used. The aim is to illustrate the potential of different fabrication and rationalization techniques in achieving design solutions that simultaneously meet the design intent, the available resources, and the established deadlines.

The first example is the Museo Soumaya in Mexico City (Figure 6.15 top-left), designed by FR-EE (Fernando Romero Enterprise) and inaugurated in 2011. In this project, the unique double-curved facade was first produced manually by the architects, the resulting physical model being then scanned to obtain the corresponding digital model. The result was a solution with 16 000 unique hexagonal panels whose shape adapted to the facade's curvature (Figure 6.21 left). To make their production viable in terms of cost and resources, the design was then submitted to a rationalization process based on Algorithmic Design (AD) strategies [357] to reduce the number of different panels. This process required the coordination of different specialized teams and the use of different computational tools, resulting in a solution with only 49 unique panels whose production was already viable [358].

The Spanish Pavilion at the 2005 World Expo in Aichi, designed by Foreign Office Architects, is another example resulting from a post-rationalization strategy. Inspired by both Islamic and Gothic architectures, this building facade was initially made of 72 unique irregular hexagonal tiles [359] that varied between six possible shapes and colors and that could be either solid or perforated (Figure

6.21 middle). As their manufacturing required the use of customized molds, the team searched for a strategy that minimized the number of different molds to make their production viable. The result was a solution made of 72 unique tiles that could be produced with only six molds, whose different combinations allowed the team to create the intended visual effect [359].

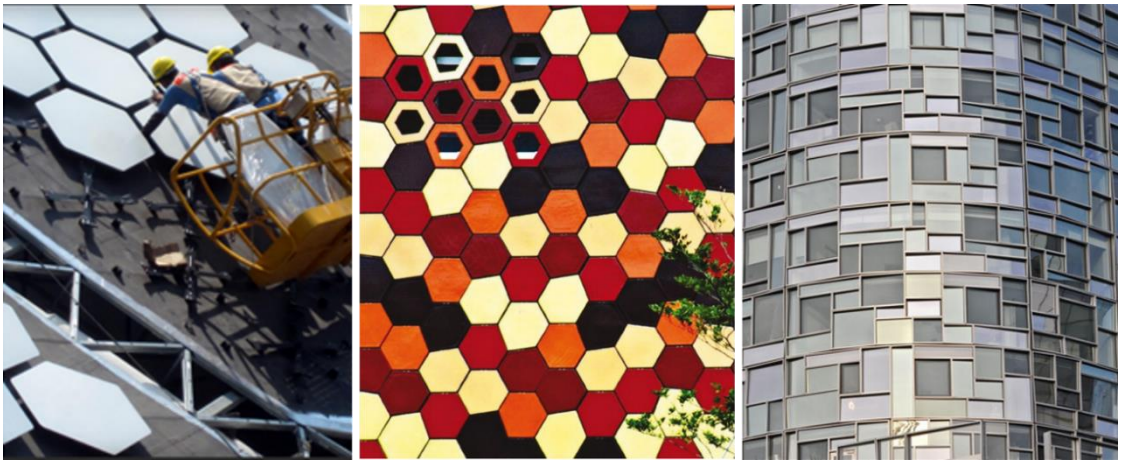


Figure 6.21. From left to right: the assembly of the customized hexagonal panels on the facade of Museo Soumaya[58]; the facade tiles of the Spanish Pavilion 2005 in Japan [360]; the 100 11th Avenue project facade (©Philippe Ruault via Dezeen).

A last example of post-rationalization is the 100 11th Avenue project (2010) in New York (Figure 6.21 right), designed by Ateliers Jean Nouvel. In this project, the design team aimed at creating a non-conventional glass facade that visually communicated the vibrancy, density, and variability of the surrounding city. The result was a patterned building facade made of glass panes of varying sizes, shapes, and materials, whose design complexity posed some challenges in terms of manufacturing. The design team addressed these challenges by iteratively rearranging the facade geometric system until achieving a feasible solution [361]. As a result, the glass panes were organized into 87 unique larger panels, whose dimensions varied according to the inside spaces' function [346]. Still, the geometry and structural integrity of the resulting framing system posed another challenge, which was solved with the replacement of the originally intended slim profiles with steel mullions of varying sections [361]. Finally, to reduce the still too high manufacturing costs of the solution, the design team collaborated with two fabrication companies [346], manufacturing the facade panels in a Chinese steel factory and then shipping them to New York City to assemble on site [361].

[58] Gehry Technologies. *Museo Soumaya: Facade Design to Fabrication* (2013). https://issuu.com/gehrytech/docs/sou_06_issuu_version/24 (Retrieved on <July 12th 2022>)

Regarding the application of co-rationalization strategies, the Federation Square in Melbourne (Figure 6.8 right), designed by LAB Architecture Studio and inaugurated in 2002, is one such example. In this project, the simplicity of the facade design system and its fabrication constraints were considered since the beginning of the design process together with the architects' design intent. The use of a tiling system inspired on Joseph Conway's triangular tiling [354] was due to its ability to constantly shift and easily create an intricate and apparently rule-less geometric effect that was easy to produce. Therefore, despite being visually complex, the resulting facade design system was quite simple (Figure 6.22 left): it used five triangles of the same material to create a larger triangular panel and then combined five triangular panels of the same or different materials to create an even larger panel [354]. As three different materials were used in the final design (zinc, sandstone, and glass), three types of facade panels were produced. In practice, all panels were manufactured in the factory and then transported to the project's site to be assembled.

Another example is the Lane 189 project in Shanghai, designed by UNStudio and inaugurated in 2017. In this project, the facade design is composed by a hexagonal grid with diamond-shaped panels of two sizes that are apparently randomly arranged, creating a complex visual effect of voids and solids (Figure 6.22 middle). The aim was to create an articulated geometric facade pattern that responded to different design constraints, such as perfectly fitting the building's curvature, providing high-quality views, and adapting its transparency level according to the inside spaces' function. The result was a set of different facade panels that varied in terms of thickness (ranging from one to three layers), material, transparency, and reflectivity. By using rationalization strategies to balance the design intent and its feasibility, the team could reach a design solution made of only 20 unique facade panels, whose strategic arrangement created the desired visual effect.

A last example of co-rationalization is the Museum of Art, Architecture and Technology (MAAT) in Lisbon, designed by Amanda Levete Architects (AL_A) and inaugurated in 2016, whose facade design considered aesthetic, manufacturing, and assembly constraints since early design stages. In this project, the final free-form facade made of different three-dimensional hexagonal ceramic tiles (Figure 6.22 right) was the result of balancing the architects' design intent, who aimed at creating complex sunlight reflections that change during the day, with both manufacturing and budget constraints[59]. As the ceramic tiles had a non-standard shape, they had to be produced using

[59] CeramicArchitectures. *Museum Art Architecture and Technology*. https://www.ceramicarchitectures.com/obras/museum-art-architecture-technology/ (Retrieved on <July 12th 2022>).

customized molds. Therefore, to minimize the facade design production cost, the team focused on reducing the number of different tiles composing it, while considering their adaptability to its curvature. In the end, the team adopted a strategy based on splitting the hexagonal grid into a trapezoidal one due to its ability to correctly fit both single and double curvature facade areas. This solution allowed the team to reach a solution made of only three types of tiles that were continuously rotated 180 degrees along the surface to create the desired dynamic undulating effect[59].

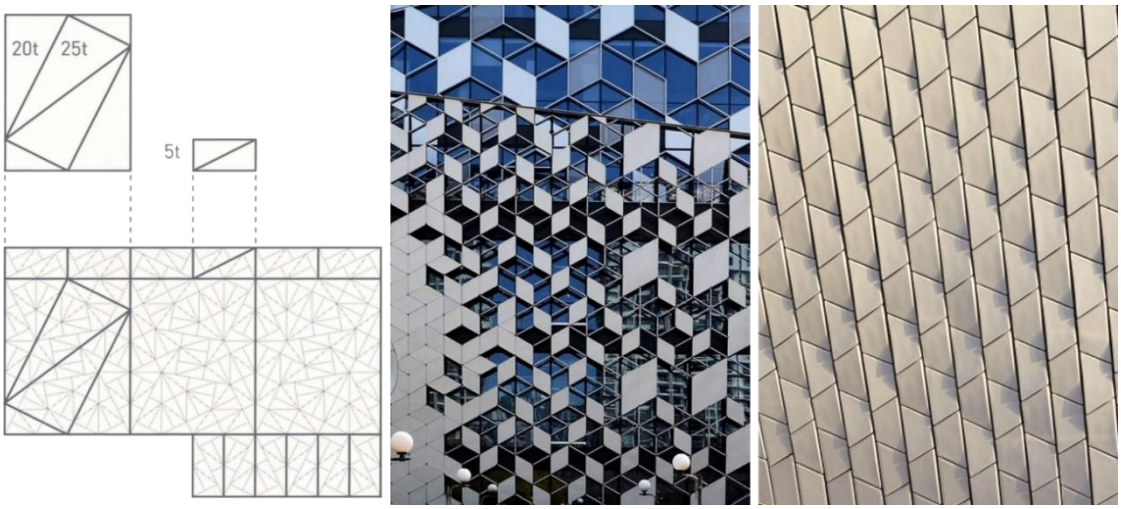


Figure 6.22. From left to right: the triangular tilling rule of the Federation Square facade (edited from [47]); the diamond-shaped facade of Lane 189 (©Eric Jap); MAAT facade tiles (©HUFTON+CROW / AL_A).

Regarding pre-rationalization, the 290 Mulberry Street building (2013), designed by SHoP Architects (Figure 6.15 bottom-left), is an example where both manufacturing and assembly constraints were considered early on and used to guide geometric exploration processes. Given the site regulations' preference for masonry facades, the design team decided to explore the potential of the available construction technologies to develop a non-standard ripple motif made of differently positioned bricks [355]. As this solution required the use of customized precast concrete panels with the bricks embedded, the architects considered the information provided by the brick-panel fabricators since early design stages to guide their decision-making process. Similarly, the constraints resulting from the panels' transportation, assembly on site [355], and manufacturing costs were also integrated in the design development process to minimize the amount of molds needed for their manufacturing, while maximizing the range of panel geometries produced by each one [355]. In the end, a single CNC cut mold was used to produce all facade panels (Figure 6.23 left), which could be blocked out in different ways to create the facade panels composing the final solution [359].

Another example of pre-rationalization is the DIY For Architects project by Sstudiomm (2016). Due to budget constraints, the architects established from the very beginning that its facade design would only use standard bricks and be built using traditional construction means. During the design exploration process, AD was used to control the stacking angle of the facade bricks, resulting in a unique brickwork pattern[60] made of differently rotated standard elements that created a dynamic visual effect (Figure 6.23 middle). Despite being visually complex, the brick facade was constructed with local resources and crafts: the bricks were manually assembled on site with the support of metal profiles, whose section shape guided the stacking angle of each brick. Even though the precision of the constructed solution was lower than its digital representation, the visual result was acceptable for the human eye resolution.

A last example of pre-rationalization is the South Asian Human Rights Documentation Centre in New Delhi (Figure 6.23 right), designed by Anagram Architects (2005). In this project, the modest budget available made the architects search for a solution that met their design intent of creating a visually complex brick facade with different levels of porosity and, simultaneously, used standard elements and traditional construction means since early on. During the design exploration stage, CD was used to control the rotation angle of each facade brick, the result being a self-supporting solution made of modules of one to six standard bricks that were strategically stacked to create the desired visual effect, while meeting the established natural ventilation and daylight illumination levels. Given the limited space available for its construction and the need to use traditional fabrication strategies, the team adopted a five-week collaborative process between masons and architects that involved the manual placement of the brick modules on site. To promote an accurate stacking process, while minimizing potential human errors, the team followed a brick assembly strategy guided by technical drawings informing about each module stacking angle and using triangular wooden wedges that ensured the stacking angles were respected[61]. The result was a self-supporting brick structure that successfully met both aesthetic and performance requirements, demonstrating the possibility of building less conventional solutions through traditional construction means.

[60] Chatel M. *DIY For Architects* (2016). https://www.archdaily.com/791588/diy-for-architects-this-parametric-brick-facade-was-built-using-traditional-craft-techniques?ad_medium=gallery (Retrieved on <July 12th 2022>).

[61] ArchDaily. *South Asian Human Rights Documentation Centre / Anagram Architects* (2010). https://www.archdaily.com/58519/south-asian-human-rights-documentation-centre-anagram-architects (Retrieved on <July 12th 2022>)

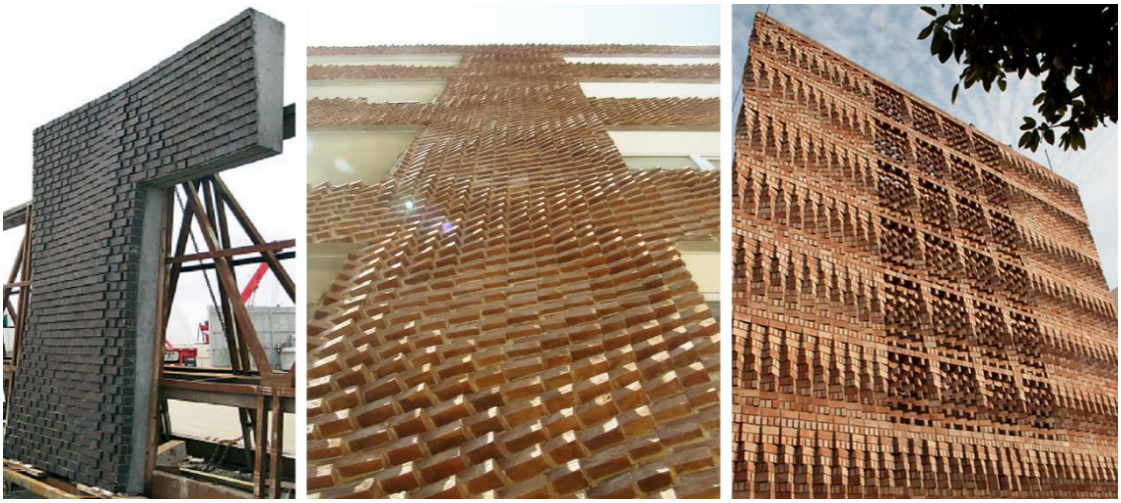


Figure 6.23. From left to right: a brick panel of the 290 Mulberry Street building facade produced in a factory and then transported to the project's site to be assembled [355]; DIY For Architects brick facade (©Sstudiomm); South Asian Human Rights Documentation Centre (©AnagramArchitects).

Finally, the Dongdaemun Design Plaza in Seoul (Figure 6.19 top-right), inaugurated in 2014, is an example where no rationalization strategy was used, evidencing the potential of DF technologies to produce highly curvilinear, freeform facades in reasonable time and with acceptable costs. Designed by Zaha Hadid Architects in collaboration with other specialized teams, including facade and BIM consultants, this building facade was originally composed of 45 133 unique aluminum panels, varying in terms of shape, color, perforation pattern, and degree of curvature[62]. Given the project's short deadline and limited budget, using the construction technology available at the time was not a possible solution, motivating the team to develop an entirely new fabrication method that made the manufacturing of thousands of different panels viable. This fabrication method is based on the use of a custom-build multipoint molding press machine to stretch and press the metal panels until reaching the desired shape, and a robotically controlled laser cutter to punch and cut the panels. As a result, the team could manufacture highly accurate unique panels that were automatically catalogued according to their location, color, and perforation pattern [336], while considerably reducing their fabrication time and cost. This project is a relevant example of how CD and DF techniques bring new construction opportunities and lead to innovative solutions whose production was nearly impossible using conventional construction techniques.

[62] Samsung C&T. *The Technology Behind Seoul's Landmark Dongdaemun Design Plaza* (2017). http://news.samsungcnt.com/the-technology-behind-seouls-landmark-dongdaemun-design-plaza/(Retrieved on <July 12th 2022>)

III

Framework

7. Mathematical Representation of Facade Designs

This investigation aims to extend the existing design knowledge by supporting architects that intend to use Algorithmic Design (AD) in solving large-scale and unconventional facade design problems involving multiple requirements, such as design intent, performance (e.g., structural, lighting, etc.), and feasibility. To that end, it provides a mathematics-based facade design methodology and an algorithmic framework that scales with the complexity of the design problems addressed while also interoperating with the various tools and techniques used.

In the past, the author addressed the limitations of facade-oriented AD frameworks in the handling of more intricate design problems. The result was a classification of facades that facilitated the geometric exploration of facade design solutions, not only helping with the identification of the best AD strategies according to the design scenario [79, 362], but also providing a set of predefined algorithms that could be easily combined in the development of new solutions. However, after using the framework for more than one year, the need to improve its structure and extend its functionalities became evident. The desire for a more flexible facade-oriented framework supporting a wider range of design scenarios and other relevant design processes beyond geometric exploration, such as analysis, optimization, and fabrication, triggered the current investigation.

7.1. Framework for Algorithmic Facades

This thesis proposes the algorithmic framework DrAFT 2.0, systematizing facade design processes to support the development of new design solutions. The preference for an algorithmic approach over a grammar-based one is due to its greater expressiveness and flexibility, as well as the higher level of control allowed. This is critical to support the intended procedural and deterministic design processes where architects fully master the course of the project and its expected results. This is also important to enhance both creative and critical thinking processes throughout the project and

thus enable a conscious and informed design space exploration and the achievement of useful results.

Taking previous work as starting point [79, 362], this investigation goes further by proposing an enhanced mathematical framework aiming at systematizing facade design processes based on AD from early to late design stages, not only integrating different geometric exploration strategies, but also the analysis, optimization, and fabrication processes typical of architectural design practice. Given the complexity and technicality of the latter, their incorporation in the framework is not straightforward, requiring profound transformations in the way the proposed mathematical principles are organized and described. Moreover, they must also respond to the variability and unpredictability typical of architectural practice and the diversity and context-specificity of its different criteria.

As the scope of this thesis is building facades, the first stage involved the collection and analysis of a large corpus of contemporary facade designs to identify recurrent design problems and relevant design strategies. It also entailed the study of existing classifications of facades [45, 47, 74, 75, 127], which are further detailed in section 2.4. Although none of the previous classifications focus on assisting architects with the algorithmic description of facade design solutions nor address their improvement in terms of performance and feasibility, they inspired the organization of the collected design knowledge in a facade-oriented way.

In the proposed framework, the mathematical principles are organized according to their type and role in facade design processes, originating a multidimensional classification that facilitates the selection of the most suitable strategies for different creative intents and design problems, while guiding their subsequent combination in the development of new solutions. When using this framework, the architect matches the design intent with the existing categorical dimensions, receiving in turn the most appropriate strategies for (1) generating the idealized facade design solution; (2) analyzing and improving its performance regarding one or more fitness criteria; (3) increasing its feasibility in terms of cost and resources; and (4) proceeding with its manufacturing. The aim is to resolve most of the design limitations found in facade design processes, especially when solving more intricate, larger-scale design problems, while adapting to the context-specificity and diversity of architectural design practice.

Since facade design processes are ruled by multiple requirements that can be generic or context-specific, straightforward or abstract, fixed or evolving, it is not reasonable to expect that this matching process yields a complete mathematical solution. Instead, the proposed framework assumes that the architect is responsible for (1) dividing the whole design into parts, (2) establishing

the dependencies between them, (3) instantiating and combining the different strategies dealing with each part, (4) implementing additional strategies that might be needed to handle the specific circumstances of the design brief, and (5) evaluating the results. Even so, its use simplifies the algorithmic description of new solutions, leaving more time for both design exploration and critical reflection and thus increasing the likelihood of finding better solutions.

7.2. CATEGORICAL 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 traditional and computational design approaches, including AD. Since this investigation addresses the second case, it must deal with the variability that typically exists in architectural design in a perspective that is simultaneously understood by a computer.

Computational tools work by following a set of imposed instructions, transmitted to computers via programming languages, whose variety is currently numerous. Even though the first computers resorted to very low-level programming languages, which greatly hindered users' understanding, with time, these have improved to become closer to human-based languages, particularly, the universally understood language of mathematics. Considering this, the investigation considers the formalism of mathematics in (1) organizing the collected design knowledge, (2) defining the ready-to-use principles, and (3) structuring the framework specialized in facade design. This allows the proposal to be understood by both architects and computers, while ensuring its universal application by enabling the framework to be integrated in diverse design briefs. Moreover, it also allows the proposal to be continuously improved with additional design knowledge as the need arises.

To overcome the limitations architects face when using AD, which are elaborated in chapter 4 of this thesis, the proposed framework must have:

- Flexibility, to support the variability typically of architectural design processes.
- Diversity, to address a large range of design problems.
- Multiplicity of criteria, to accurately evaluate different design scenarios.
- Coherency, to correctly link different types of data and processes in a single workflow.

Obviously, covering all possible design scenarios would require the framework to integrate an infinite set of strategies and solutions within its structure, which is impossible. Nevertheless, by

considering generic solutions to recurrent design problems, as those encapsulated by the existing modular programming and design pattern techniques [7], the framework can not only reduce the initial investment required by AD, but also embrace a wider range of design scenarios and requirements. Additionally, by having these techniques available since initial design stages, architects spend much less time and effort with algorithmic tasks due to not having to rewrite all the algorithms from scratch every time they start a new design. As a result, the framework does not consider all possible design scenarios and problems, instead focusing in the most common ones and, more importantly, on being adaptable to more specific design briefs. The latter cases can then be incorporated in it whenever the need arises.

Regarding the framework's structure, it is organized in a modular way into six main categories of principles (Figure 7.1), each addressing a specific task of facade design:

- Geometry: to define the building's overall shape.
- Distribution: to distribute its facade elements.
- Pattern: to generate facade elements of varying shapes.
- Optimization: to adapt the design according to one or more fitness goals.
- Rationalization: to control the design's feasibility.
- Fabrication: to prepare the design for manufacturing.

This modular structure was carefully designed to allow including additional dimensions (i.e., categories and/or subcategories) and the combination of novel design strategies and knowledge with the already existing ones.

The following sections detail each category in terms of mathematical formulation and combination, while illustrating the ability of the proposed principles to simplify the replication of prior design knowledge through an algorithmic perspective. Then, in chapter 8, these principles are evaluated in terms of their ability to generate new design knowledge while reducing the algorithmic complexity in the development of novel facade design solutions.

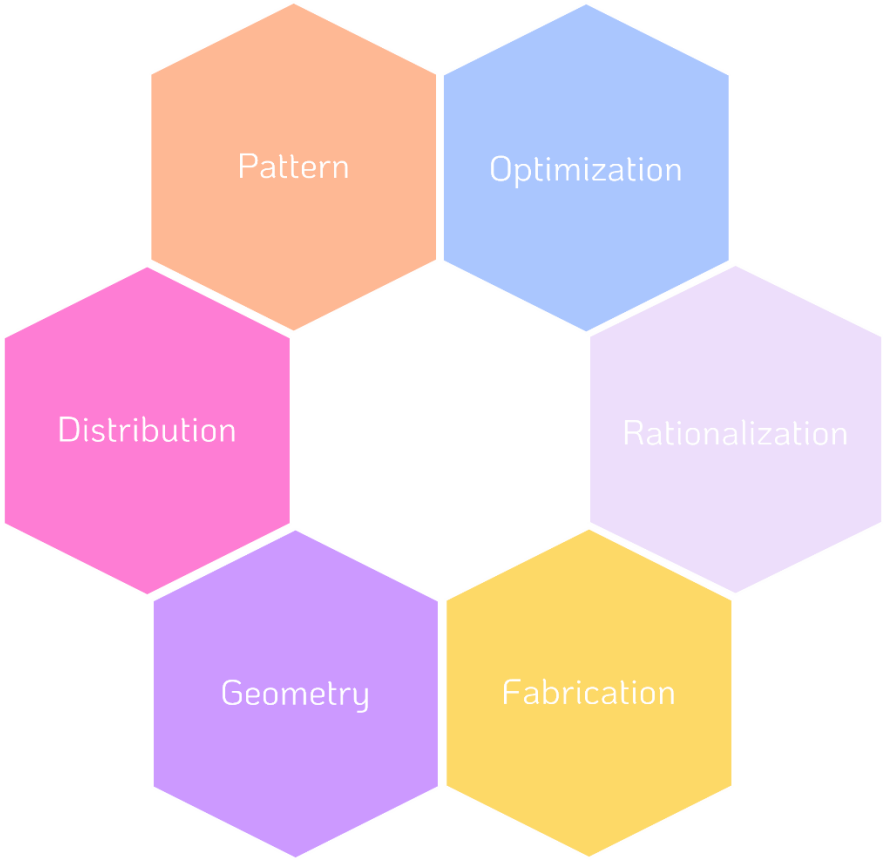


Figure 7.1. DrAFT 2.0 modular structure composed of six main categories.

7.3. MATHEMATICAL FORMULATION

This thesis seeks to systematize and simplify algorithmic-based facade design processes, proposing a mathematical framework synthetizing prior design knowledge into ready-to-use strategies that can be combined in the generation of new design knowledge. The latter are organized according to their type and role in the facade design process in the following categories: *Geometry*, *Distribution*, *Pattern*, *Optimization*, *Rationalization*, and *Fabrication*. For each one, the framework provides $\mathbb{R} \rightarrow \mathbb{R}$, $\mathbb{R} \rightarrow \mathbb{R}^2$, $\mathbb{R}^2 \rightarrow \mathbb{R}^2$, and $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ functions, among others, that can be then combined through function composition.

Before proceeding to the different categories, it is important to first explain a set of fundamental concepts that are used to describe the framework's mathematical principles and their combination. The first one is the *anonymous* function concept introduced by the mathematician Alonzo Church in the 1930s when inventing the lambda calculus (or λ-calculus) notation. Differently from traditional mathematical functions, such as the sinus or the square root, anonymous functions have no name, being often used as arguments to other functions or to allow these to return functions as result. This leads us to the second concept of *higher-order functions* (HOFs), which represent functions that receive other functions as arguments and/or compute other functions as results [363]: e.g., the function composition ∘ is a HOF that receives two functions as input and returns a function as result, such as $(f \circ g)(x) = g\big(f(x)\big)$.

The next concept regards the way the framework deals with *surface sampling*, which is particularly relevant to then understand the mapping and transformation processes explained in sections 7.3.2 and 7.3.3. The framework adopts a matrix-based approach because of (1) its ability to systematize and simplify the representation of different surface subdomains and (2) its combinatorial potential. In practice, upon receiving the mathematical description of the surface, the framework produces a matrix of points belonging to the surface. Figure 7.2 illustrates this with two mathematical descriptions of a torus surface, namely, the parametric and the implicit ones (left and middle examples, respectively), and the matrix-based representation of a sub-domain of that same surface.

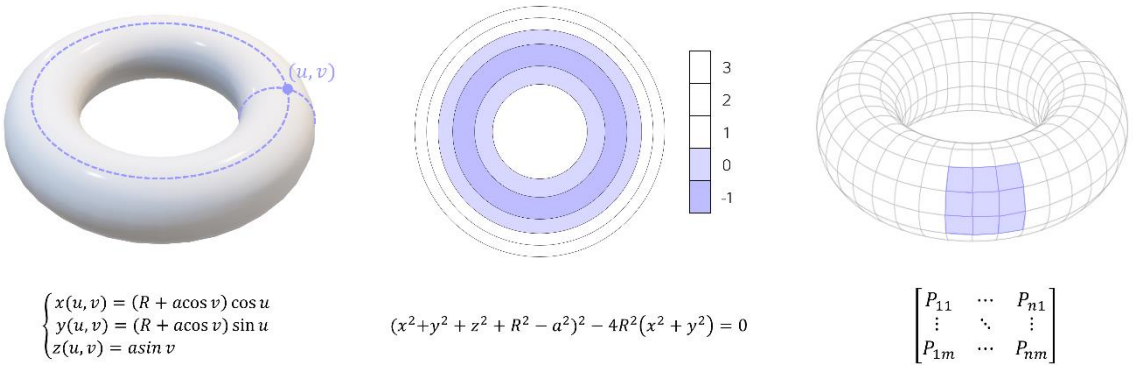


Figure 7.2. Surface representation methods conceptual illustration: from left to right, describing the surface of a torus by using a parametric, implicit, and matrix-based representation.

Finally, in general, the framework handles all surface-related functions $S(u, v)$ within the domain $0 \leq u \leq 1, 0 \leq v \leq 1$ and provides HOF operators that can be arbitrarily combined, namely the one-dimensional linear variation function $linear(a, b) = \lambda(t).\, a + (b - a)t$ and the (paradoxical) constant "variation" function $constant(c) = \lambda(t \ldots).\, c$. Here, it is employed the λ-calculus notation for an anonymous function with parameter t [363], which means the result of the function $constant$ is 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.

The next sections explain the mathematical formulation of the six categories.

7.3.1. GEOMETRY

Given that designers want their models to be flexible, when they define the underlying principle of a shape, they want to easily control and change it, so that many design instances can be generated within the same geometric principle. This idea guides the first category *Geometry*, which contains algorithms to define the building facade's overall shape. For each different geometry, the framework provides a $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ parametric function that describes a point on the facade surface and that when mapped along two ranges of values, returns the corresponding surface shape. For example, $f(u, v) = XYZ(u \times 5, 0, v \times 5)$, where XYZ is the Cartesian coordinate function, represents a five-by-five square on the XZ plane.

Naturally, other coordinate systems can be used, such as the *Cylindrical*, represented by the function CYL, and the *Spherical*, represented by the function SPH. Moreover, each coordinate system supports different types of transformations (e.g., translation, rotation, etc.), which are related to a spatial location of reference, capable of codifying the transformed referential, which, for brevity, we will omit.

As an example, consider the completely planar facade of the Formstelle Building (Figure 7.3 left). Depending on a width w and height h values, the facade surface can be defined as $straight(w, h) =$

$\lambda(u,v).XYZ(u \times w, 0, v \times h)$. Note that the algorithm $straight$ is a HOF that returns an anonymous parametric function representing a delimited region on the XZ plane. Although it is perfectly possible to explicitly define algorithms such as $straight$, the framework goes deeper than that by providing a set of more fundamental functional operators that can be arbitrarily combined, such as the already mentioned algorithms $linear$ and $constant$. As the algorithms of the *Geometry* category are often two-dimensional, we need to extend the one-dimensional variations' domain into $\mathbb{R}^2$. We use HOFs to define two functions, $dim_u(f) = \lambda(u,v).f(u)$ and $dim_v(f) = \lambda(u,v).f(v)$, that make a function f vary only in one dimension, i.e., u or v accordingly. To generalize function composition operations, the framework provides the operator

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

and, to simplify the notation used, it defines $u \otimes lim = dim_u\big(linear(0, lim)\big)$ and $v \otimes lim = dim_v\big(linear(0, lim)\big)$, wherein lim is the domain's upper limit. The framework also treats all numbers n in a function context as $constant(n)$ and any first-order function f being used with functional arguments $g_1, \cdots, g_n$ as $\circ(f, g_1, \cdots, g_n)$, thus being $f \times g$ the same as $\circ(\times, f, g)$. Moreover, it also considers that any function applied to fewer arguments than the required ones is curried [364].

Figure 7.3. Formestelle Building, Töging am Inn, Germany (©Format Elf Architekten); Suzhou SND District Urban Planning Exhibition Hall by BDP architects, Jiangsu, China (©MarcoJacobs); SMG apartment building by Mejiro Studio, Tokyo, Japan (©koichi torimura).

With HOFs we can move from the numeric space into the functional space and combine functions using functional operators rather than simply combining numbers using numeric operators. In a functional space, the algorithm $straight$ has the equivalent representation $straight(w,h) = XYZ(u \otimes w, 0, v \otimes h)$. Similarly, cylindrical facades such as that of Figure 7.3 (middle) are described by the function $cylindrical(r,h) = CYL(r, u \otimes 2\pi, v \otimes h)$, where r is the radius size of the building and h its height. Regarding undulating facades, the framework represents the most common sinusoidal movement with the HOF $sinusoid(a, \omega, \phi) = \lambda(x).a \times sin\,(2\pi\omega x + \phi)$, where a is the amplitude of the sinusoid, ω

is the angular frequency, i.e., the number of cycles per unit length, and ϕ is its phase. When the undulation movement occurs, for instance, in the XY plane, producing a horizontal wave as that of Figure 7.3 (right), the surface is described as:

$$sinusoidal_u(w, h, a, \omega, \phi) = XYZ(u \otimes w, u \otimes sinusoid(a, \omega, \phi), v \otimes h)$$

When the sinusoidal effect occurs in both XY and YZ planes, creating horizontal and vertical waves such as those of Figure 7.4 (left), it is described as:

$$sinusoidal_{uv}(w, h, a, \omega, \phi) = XYZ(u \otimes w, u \otimes sinusoid(a, \omega, \phi) \times v \otimes sinusoid(a, \omega, \phi + \pi/2), v \otimes h)$$

Figure 7.4. Boiler House at Guy's Hospital, London, UK (©heatherwick); Selfridges Building, Birmingham, UK (©Bs0u10e0).

Finally, the framework also supports the generation of more unconventional or organic facade shapes as that of Figure 7.4 (right), which are often hard to describe mathematically but easy to create manually. For such cases, it provides the algorithm $freeform$, which is a type of visual input mechanism (VIM): it allows the architect to manually model the desired shape in the design tool and then use it as input to the available algorithms. Other examples of VIMs are introduced throughout this chapter.

In practice, when selecting a manually created surface, the algorithm $freeform$ converts it into a format suiting the remaining algorithms (i.e., into an algorithm reproducing that same surface), while allowing the user to decide whether the dependency between the algorithms and the selected shape is preserved. In the affirmative case, the algorithms are automatically regenerated every time the input shape is changed in the design tool. Otherwise, they only consider these changes when the shape is re-selected, thus becoming independent from it: not only do they no longer react to the changes made to the original shape in the design tool, but they also can store its algorithmic description to then

reproduce it either in the same or another design tool. This enables the framework to have a general application, not restricting the available functionalities to a specific tool, while permitting to take advantage of shapes produced in multiple modeling environments.

Having the algorithmic equivalent of the manually generated shape then allows the architect to reproduce and manipulate it whenever necessary, as well as combine it with the algorithms of the remaining categories and create, for instance, different point distributions and geometric patterns on it. Nevertheless, as the resulting algorithm is non-parametric, it only represents and reproduces the original shape and not variations of it.

The next sections explain how the algorithms available in the *Geometry* category can be combined with those of *Pattern* and *Distribution* categories in exploring different facade design patterns.

7.3.2. DISTRIBUTION

While the previous category is concerned with the description of different surface shapes, this one focuses on converting the latter descriptions into actual points in space. To that end, it contains algorithms that, given the parametric representation of a surface shape and the interval of values to assign to its variables, create different surface points configurations within that range.

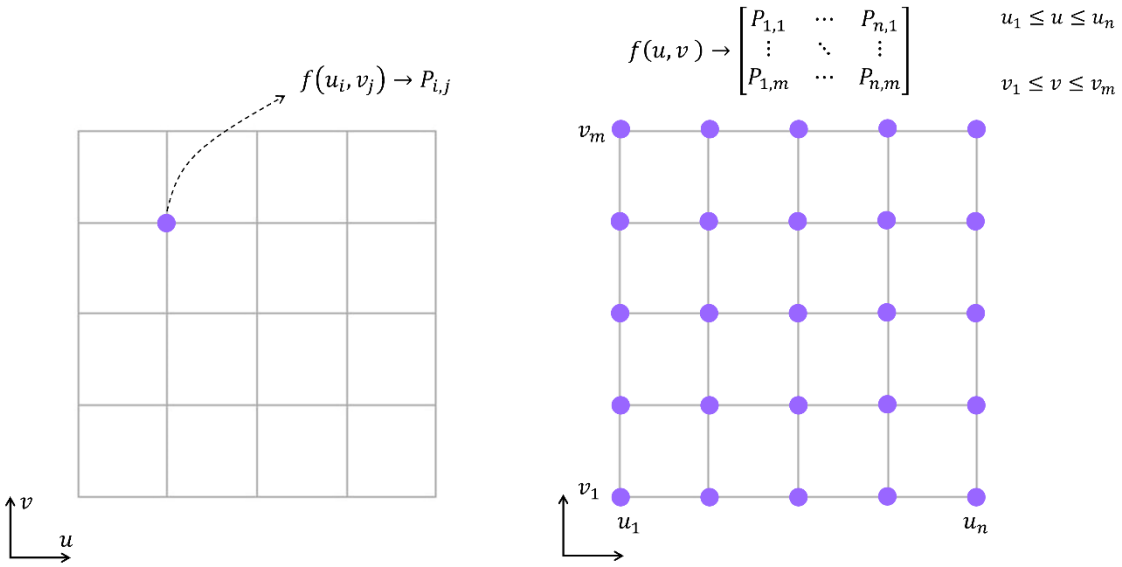


Figure 7.5. On the left, a point resulting from the parametric representation of a surface f and, on the right, the set of surface point sampling resulting from its application over two intervals, i.e., $[u_1, u_n]$ and $[v_1, v_m]$, composed by n and m values respectively.

In practice, these algorithms receive a matrix of surface point samples to rearrange ($ptss$), which can be produced by iterating the functions available in the *Geometry* category along two domains, i.e., the

u and v dimensions of the surface sampling (see Figure 7.5). They then return another matrix with the same surface points but rearranged in different configurations.

As an example, given the same set of surface points $ptss$, the algorithm $grid_{rectangles}$ rearranges them in sets of four points describing rectangular areas on the surface (Figure 7.6-A); the algorithm $grid_{triangles}$ organizes them in sets of three points representing triangular areas (Figure 7.6-B); and the algorithm $grid_{rhombus}$ in sets of four points creating diamond-shaped areas (Figure 7.6-C).

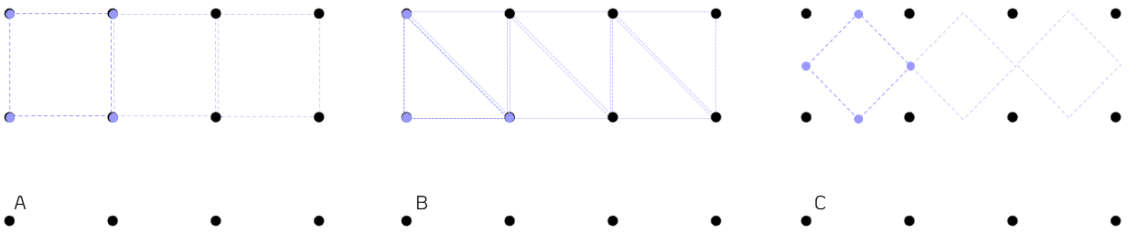


Figure 7.6. Conceptual representation of the points rearrangement: the black dots represent the surface point samples and the purple dots their reorganization in sets of four points (A and C) and three points (B).

The way the given surface point sampling is rearranged in the new matrix therefore originates different grid configurations, as those illustrated in Figure 7.7: for instance, we can mathematically describe example A as $grid_{squares}(ptss)$ and example F as $grid_{rhombus}(ptss)$, wherein $ptss$ is the result of mapping the $straight$ algorithm along two domains, i.e., $[u_1, u_n]$ and $[v_1, v_m]$, with n and m subdivisions in both u and v directions (in this case $n = m = 10$).

Note that some *Distribution* algorithms rearrange the original surface point sampling ($ptss$) into sets of points with the same size but with different organizations. As an example, consider the algorithms $grid_{rectangles}$ and $grid_{rhombus}$, which correspond to the examples A and C of Figure 7.6, respectively: both rearrange the given set of points ($ptss$) in sets of four points but, while the former directly groups the given points to create a rectangular grid configuration, the latter calculates their intermediate points to then group them into sets of four as well but originating a diamond-shaped configuration.

By combining these algorithms with those creating different facade elements, such as those available in the *Pattern* category, we can produce several element distributions that in turn originate different surface patterns.

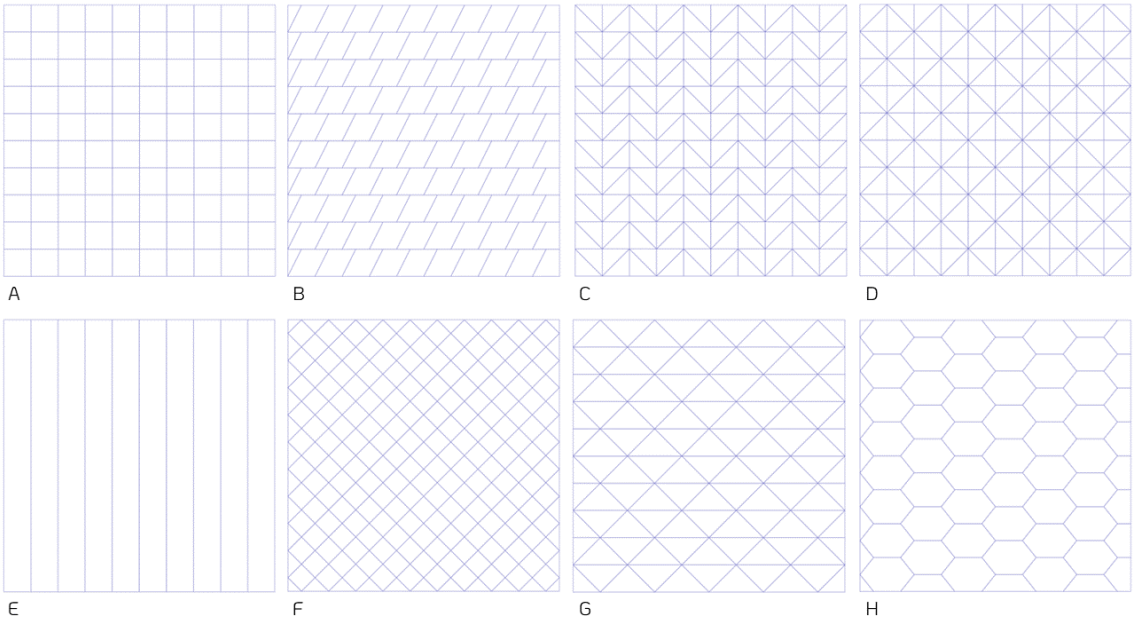


Figure 7.7. Conceptual representation of eight *Distribution* algorithms: $grid_{rectangles}$ (A), $grid_{sweeked}$ (B), $grid_{trianglesX}$ (C), $grid_{trianglesXY}$ (D), $grid_{stripes}$ (E), $grid_{rhombus}$ (F), $grid_{trianglesHexagon}$ (G), and $grid_{hexagons}$ (H).

7.3.3. PATTERN

This category contains algorithms to create different facade elements. When combined with those of the previous categories, these algorithms produce several facade design patterns. This thesis categorizes as *discrete* those patterns resulting from repeating the same element along both $\boldsymbol{u}$ and $\boldsymbol{v}$ dimensions (Figure 7.8 left) and as *continuous* those repeating the same element along only one of the dimensions (Figure 7.8 right). In both cases, the repeated element can be kept unchanged along the facade's domain or can suffer some transformations regarding its shape, size, etc. Therefore, this category provides algorithms to generate different geometric shapes, as well as to apply different geometric transformations to them. These are organized in two groups, namely *Shape* and *Transformation*, each one containing algorithms to handle both discrete and continuous patterns.

Mathematically, the framework represents discrete and continuous patterns differently in terms of how the elements are shaped and geometrically transformed: while the former case deals with polygonal geometries (e.g., regular polygons, star-shapes, rosettes, etc.) and geometric solids (e.g., spheres, cones, etc.), among others, the latter handles stripe-based elements such as pillars, bars, tubes, etc., which requires the geometric transformations to be different. The next sections explain the algorithms addressing both types of shapes and geometric transformations.

Figure 7.8. Lisbon Aquarium Extension by Campos Costa Arquitetos (©Daniel Malhão); ONS Incek Residence Showroom and Sales Office by Yazgan Design Architecture (©Yunus Özkazanç).

SHAPE

Before describing the shapes and geometric transformations available in the framework, it is important to present the HOF $\boldsymbol{pattern}$, which is responsible for combining the algorithms of both *Shape* and *Transformation* groups and produce the overall facade pattern. Mathematically, the framework represents this algorithm as $\boldsymbol{pattern}(f, ptss; args\ldots)$, where f is either the function or the composition of functions shaping each facade element, $ptss$ is the set of points describing the facade surface, and $args\ldots$ the supported optional arguments. In practical terms, the algorithm $\boldsymbol{pattern}$ provides the optional arguments ($args\ldots$) as input for the received function(s) (f), which is/are then mapped along the received surface points ($ptss$). Regarding the f argument, it can be a single function simply describing a geometric shape, resulting in a surface pattern with non-varying geometric elements, or it can be a composition of different *Shape* and/or *Transformation* algorithms, producing a geometric pattern either with different or identical elements that either vary or not their shape along the facade surface.

The next subsections present, first, some of the *Shape* algorithms addressing discrete patterns, which create different 2D and 3D shapes, and then those targeting continuous patterns, which produce stripe-based shapes.

TWO-DIMENSIONAL ELEMENTS

This investigation considers a two-dimensional shape as a geometric entity defined by a set of points and lines connecting those points in a closed chain and the points contained within it. When bounded by curves, these shapes are named as curved shapes, and when bounded by straight lines, they are named as polygons. In either case, the available algorithms receive the set of points where to center

the geometric shape ($\boldsymbol{pts}$), an information provided by the *Distribution* algorithms, being the remaining parameters dependent on the characteristics of each geometric shape: $\boldsymbol{shape(pts; args\ \ldots)}$. Figure 7.9 illustrates six *Shape* algorithms and their corresponding parameters.

The first group, curved shapes, includes, among others, circular, elliptic, and super-elliptic geometries, providing for the first case the algorithm $\boldsymbol{shape_{circle}}$, which receives the set of points ($\boldsymbol{pts}$) where to center the circle (Figure 7.9-A) and the radius size ($\boldsymbol{r}$); for the second case the algorithm $\boldsymbol{shape_{ellipse}}$, which receives the same set of points, two radius sizes ($\boldsymbol{r_u}$ and $\boldsymbol{r_v}$), and a placement angle ($\boldsymbol{\alpha}$); and for the last case the algorithm $\boldsymbol{shape_{superellipse}}$, which receives the previous parameters plus the degree of the curve ($\boldsymbol{n}$):

$\boldsymbol{shape_{circle}(pts, r)}$ $\boldsymbol{shape_{ellipse}(pts, r_u, r_v, \alpha)}$ $\boldsymbol{shape_{superellipse}(pts, r_u, r_v, \alpha, n)}$

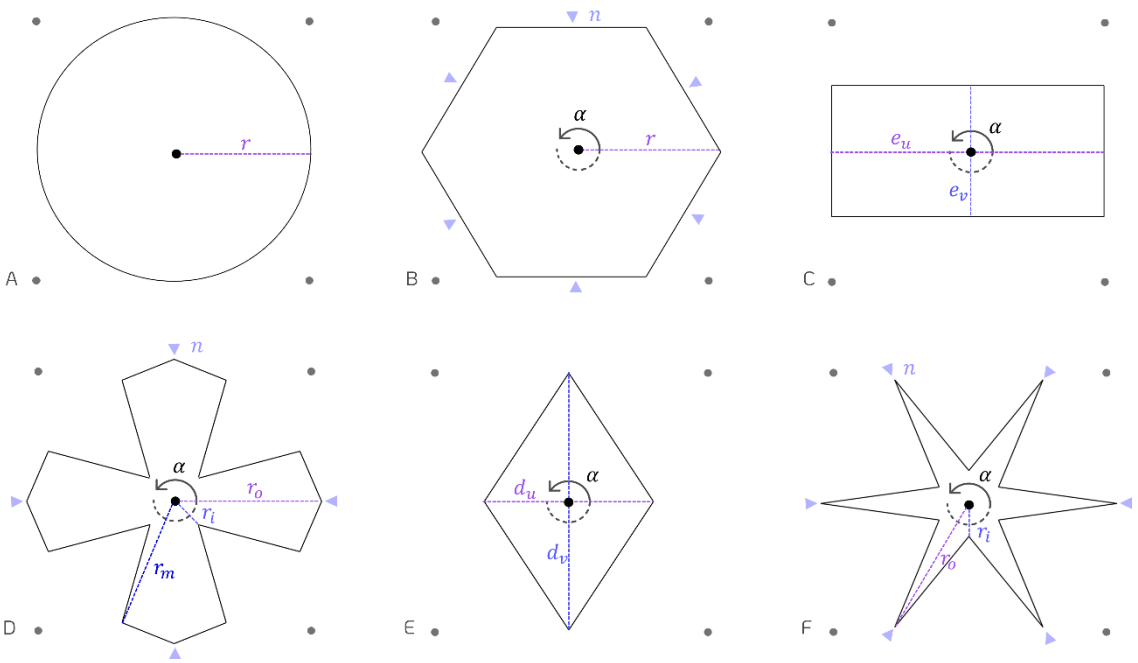


Figure 7.9. Parameters of six *Shape* algorithms: besides the set of points $\boldsymbol{pts}$ represented by the black dots, algorithm (A) receives a radius; (B) receives a radius, an angle, and a number of sides; (C) receives a length, a width, and an angle; (D) receives three radii, an angle, and a number of vertices; (E) receives two diagonals and an angle; and (F) receives two radii, an angle, and a number of vertices.

The second group, polygons, includes planar figures bounded by a finite chain of straight-line segments, called edges or sides, closing in a loop. The points where two edges meet are named vertices. When the polygon boundary intersects itself, it creates self-intersecting polygons like star polygons and rosette polygons, among others; otherwise, it creates regular and irregular polygons like squares, hexagons, rectangles, rhombuses, parallelograms, non-regular triangles (isosceles and scalene), among

others. The way different polygonal shapes are organized in the framework is inspired by the existing classifications of polygons, which differentiate them according to, for instance, the number of sides, the equality of the edges and/or angles, and their convexity. Given the high number of algorithms implemented, this section only illustrates some of them.

As a first example, consider the algorithm that creates regular polygons, whose practical application in architecture is illustrated in Figure 7.10. It receives the set of points where to center the element ($\boldsymbol{pts}$), the number of sides ($\boldsymbol{n_{sides}}$), the radius ($\boldsymbol{r}$), and the placement angle ($\boldsymbol{\alpha}$). Similarly, the algorithms creating rectangles and rhombuses receive the same set of points and placement angle and, in the first case (Figure 7.9-C), the length of its horizontal and vertical edges ($\boldsymbol{e_u, e_v}$), and, in the second case (Figure 7.9-E), the length of both its inner diagonals ($\boldsymbol{d_u, d_v,}$):

$$shape_{regularpolygon}(pts, n_{sides}, r, \alpha) \quad shape_{rectangle}(pts, e_u, e_v, \alpha) \qquad shape_{rhombus}(pts, d_u, d_v, \alpha)$$

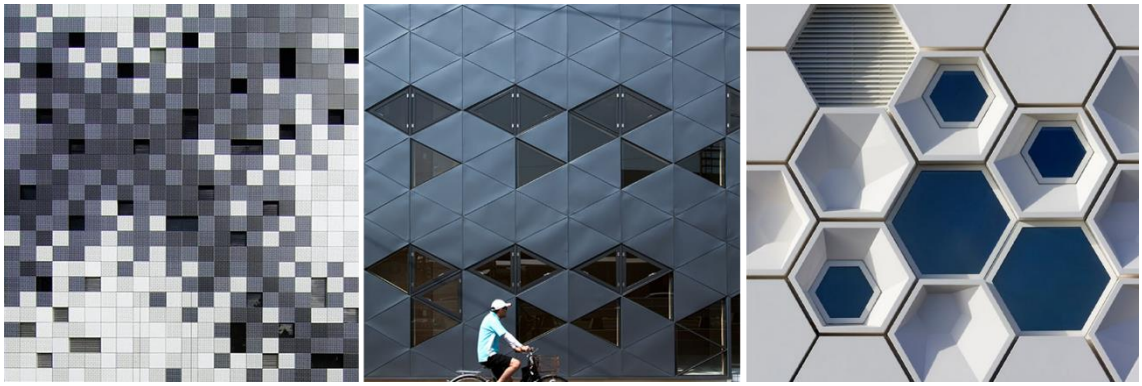


Figure 7.10. Frog Queen building by SPLITTERWERK, Graz, Austria (©Nikolaos Zachariadis); commercial building in Fukuoka Prefecture by Junichiro Ikeura, Japan (©Satoshi Ikuma); The Afsluitdijk Wadden Center by GEAR Architectencoöperatie, Netherlands (©Gerard van Beek).

Regarding star polygons, these shapes result from connecting non-adjacent vertices of a regular polygon until the first vertex is reached again [365]. They are often classified as self-intersecting, equilateral, and equiangular shapes, and when the intersecting lines are removed, they are also classified as *isotoxal* [366] due to their vertices alternating between two radii values. The use of these polygons in architecture has a long tradition, mainly in North-African and Arabic cultures. Lately, they have also been widely applied in contemporary building facades, as illustrated in Figure 7.11.

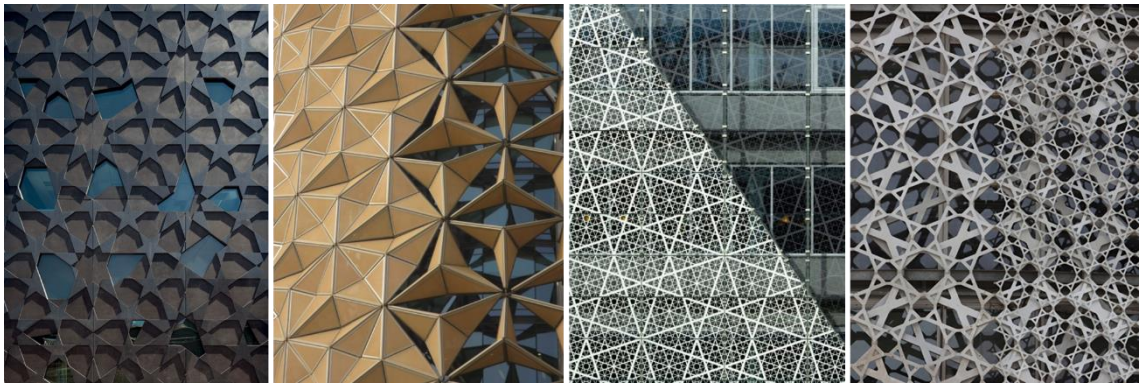


Figure 7.11. Yardmasters building by McBride Charles Ryan, Australia (©John Gollings); The Al Bahar Towers by Aedas (©WIKIARQUITECTURA); Stadshuis Nieuwegein by 3XN architects (©Adam M¢rk); Doha Tower by Jean Nouvel (©Ateliers Jean Nouvel).

The algorithm representing such shape, $shape_{star}$, takes as input the set of points received by all *Shape* algorithms (pts) plus the number of vertices ($n_{vertices}$) (Figure 7.12-A), two radii sizes (r_{convex} and $r_{concave}$), i.e., the distances between the star's center and either its *convex* or *concave*[63] vertices (Figure 7.12-B), and the placement angle (α) (Figure 7.12-C). To further increase its flexibility, an additional parameter was added, the factor $k_{apperture}$ ranging from 0 to 1, that allows apertures of different sizes and orientations (Figure 7.12-D): when $k_{apperture} = 0$ the result is a totally opaque star polygon; when $k_{apperture} = 1$ the result is a star-shaped polygonal line.

$$shape_{star}(pts, n_{vertices}, r_{convex}, r_{concave}, \alpha, k_{apperture})$$

As a last example, consider those polygons whose shape is inspired on the Islamic rosettes. According to some authors [367, 368], these geometries are a type of star polygons since they result from lines connecting vertices alternating, in this case, between three radii sizes. The framework names these elements rosette polygons, providing the algorithm $shape_{rosette}$ to generate them. In addition to the set of points (pts) and placement angle (α), this algorithm receives the number of petals (n_{petals}) and three radius sizes (r_{inner}, r_{middle}, and r_{outer}) corresponding to the inner, middle, and outer points (Figure 7.13) of the rosette:

$$shape_{rosette}(ptss, n_{petals}, r_{inner}, r_{middle}, r_{outer}, \alpha)$$

[63] When the internal angle of the polygon is less or more that π radians, respectively.

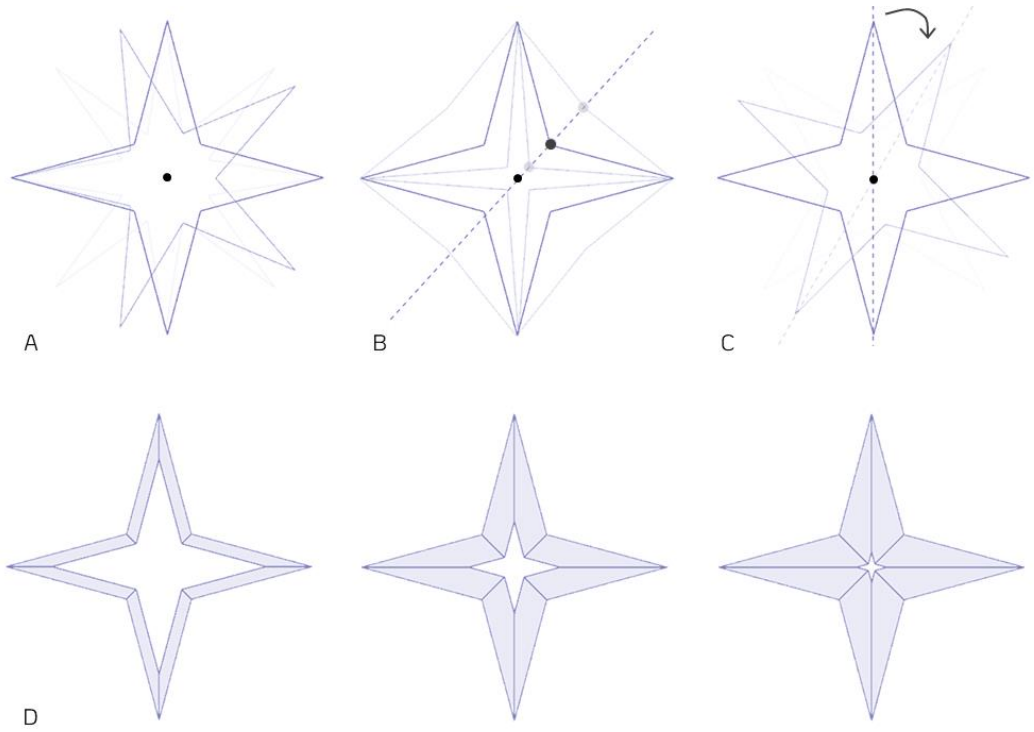


Figure 7.12. Conceptual representation of the algorithm $\boldsymbol{shape_{star}}$: (A) number of sides; (B) distance between the star center and its concave vertices; (C) placement angle; (D) apertures resulting from different factors.

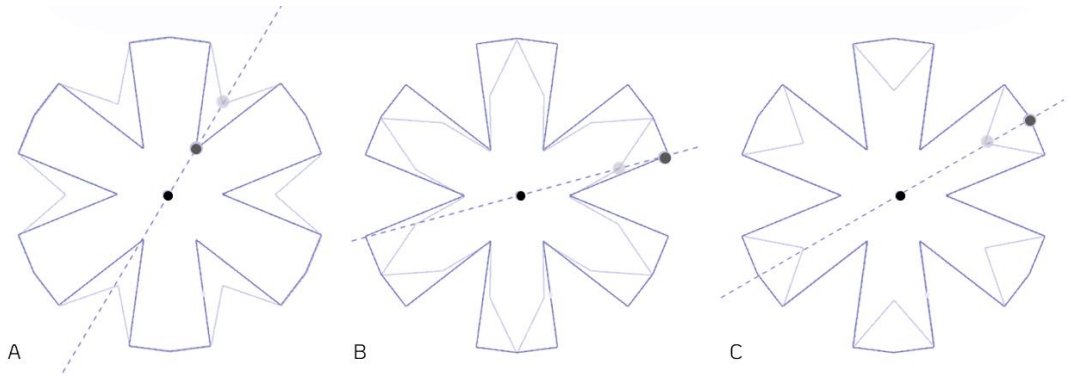


Figure 7.13. Algorithm $\boldsymbol{shape_{rosette}}$ conceptual representation: inner (A), middle (B), and outer (C) radii.

Before proceeding to the *Transformation* group, the combination of the *Shape* algorithms with both *Geometry* and *Distribution* ones is illustrated. As a first example, consider the creation of a geometric pattern based on star polygons. Besides selecting the algorithm $\boldsymbol{shape_{star}}$, we need to define (1) the surface where to apply the star polygons and (2) their type of distribution. Imagine that, for the former case, we choose a straight surface ($\boldsymbol{straight}$) and, for the latter, we select a rhombus grid distribution ($\boldsymbol{grid_{rhombus}}$). Then, we combine the three algorithms in the process illustrated in Figure 7.14.

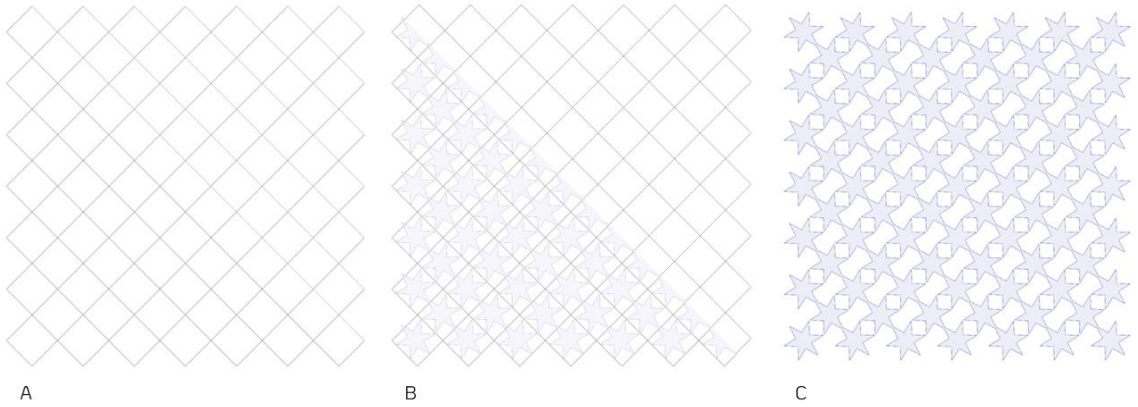


Figure 7.14. Conceptual representation of the composition of (A) $straight$ and $grid_{rhombus}$ algorithms, originating a rhombus grid; and (B) the previous algorithms plus the $shape_{star}$ one, distributing star-shape elements along the previous grid (the result is illustrated in C).

We can simplify the composition of both $straight$ and $grid_{rhombus}$ algorithms by writing $grid_{rhombus}(straight(w,h))$ and, to facilitate the mathematical representation of algorithms dealing with matrices, we can take advantage of *broadcasting*[64] and apply a function f to an array of elements with the same or different number of dimensions from the other received arguments. Broadcasting is represented by the *dot syntax* $f.(args\ldots)$ and it can be applied in single or nested calls $f.(g.(args\ldots))$. This means we can simplify the function composition illustrated in Figure 7.14 into

$$shape_{star}.\begin{pmatrix} grid_{rhombus}(straight(w,h)), \\ n_{vertices}, \\ r_{convex}, \\ r_{concave}, \\ \alpha, \\ k_{apperture} \end{pmatrix}$$

where $grid_{rhombus}(straight(w,h))$ returns a matrix of surface points arranged in a rhombus grid configuration and the broadcasting operation maps the $shape_{star}$ algorithm onto it.

It is important to note that the resulting mathematical structure supports a greater design flexibility, offering control over the various parameters differently and independently and thus producing a wider range of geometric patterns. While mapping the algorithm $shape_{star}$ along a squared grid produced by the algorithm $grid_{squares}$, we can make the parameter $n_{vertices}$ vary, for instance, randomly within a range of options by using the *Transformation* algorithm T_{random}, which is further explained in the Transformation section; a scenario illustrated in Figure 7.15-A with a star pattern

[64] The mapping of a function over an array or a matrix.

whose number of vertices randomly vary between 3 and 10. This is valid for all *Shape* algorithms, some of which are illustrated in Figure 7.15.

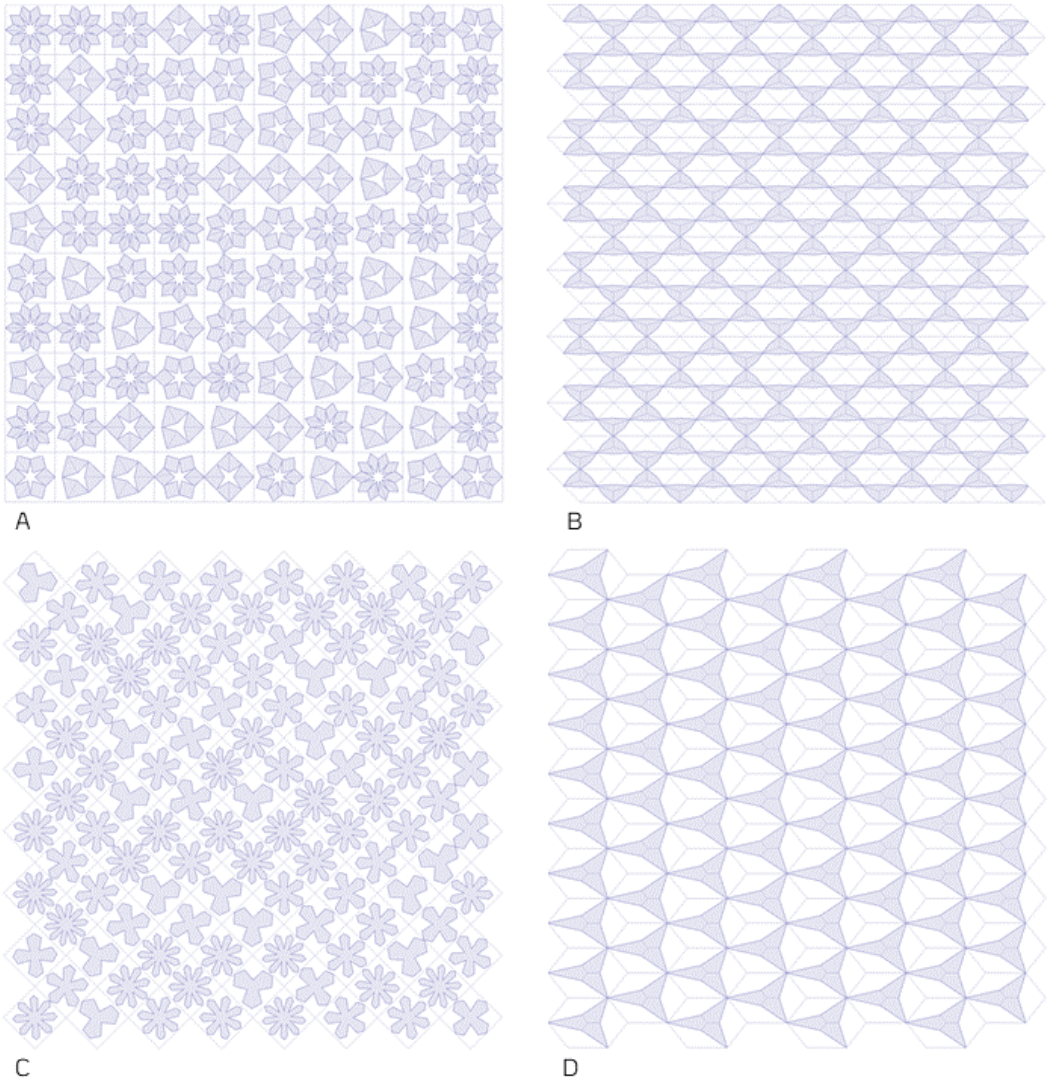


Figure 7.15. Examples of *Shape* and *Distribution* algorithms combinations: (A) $grid_{squares}$ and $shape_{star}$ with a random number of vertices and radius size; (B) $grid_{trianglesHexagon}$ and $shape_{rhombille}$; (C) $grid_{rhombus}$ and $shape_{rosette}$ with a random number of petals; (D) $grid_{hexagons}$ and $shape_{rhombitrihexagonal}$.

THREE-DIMENSIONAL ELEMENTS

The framework also includes algorithms to produce different three-dimensional shapes, including simple geometric solids, such as prisms, boxes, cylinders, cones, and pyramids, and composite solids[65]. Figure 7.16 presents some facade design examples using three-dimensional elements.

[65] A solid composed of two or more other solids.

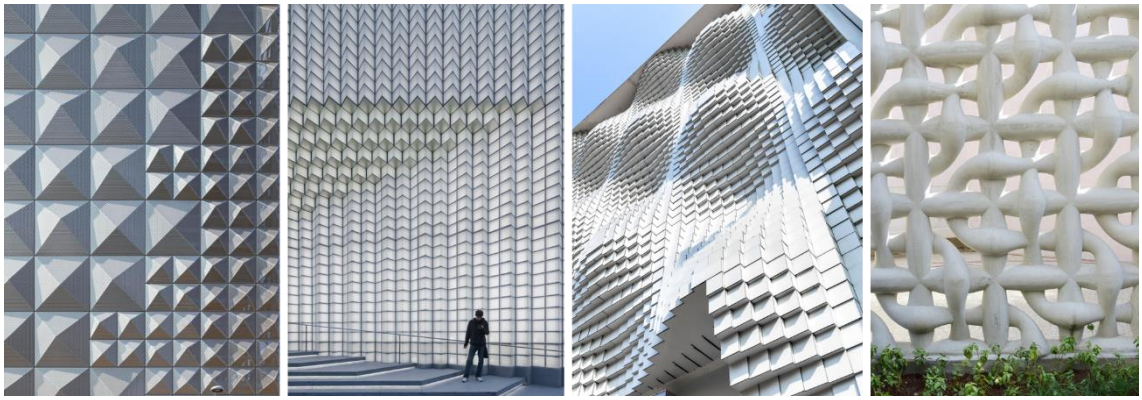


Figure 7.16. Three-dimensional facades: Community Car Park A1 by XVW architectuur, Amsterdam (©Stijn Brakkee, Isabelle Nabuurs); Shangai boutique design by UUfie studio (©Shengliang Su); Regal Shoes shop by Nudes, India (©Nudes); Consulate of Portugal by Campos Costa Arquitetos and DAU/ PUC-Rio students, Brazil (©Henrique Delarue).

Like two-dimensional shapes, these algorithms all receive the set of points where to center the shape ($\boldsymbol{pts}$), the placement angle ($\boldsymbol{\alpha}$), except the $\boldsymbol{shape_{sphere}}$ algorithm, and a set of additional parameters depending on the type of element to create: for instance, the algorithm $\boldsymbol{shape_{cube}}$ receives an edge size ($\boldsymbol{e}$), the algorithm $\boldsymbol{shape_{sphere}}$ receives a radius size ($\boldsymbol{r}$), and the algorithms $\boldsymbol{shape_{cylinder}}$ and $\boldsymbol{shape_{cone}}$ receive a radius size ($\boldsymbol{r}$) and a height ($\boldsymbol{h}$).

$$shape_{cube}(pts, e) \qquad shape_{sphere}(pts, r) \qquad shape_{cylinder}(pts, r, h) \qquad shape_{cone}(pts, r, h)$$

In addition to these shapes, the framework provides algorithms to create other types of three-dimensional facade elements, such as bricks, ceramic elements, trusses, among others. These algorithms receive the same set of points ($\boldsymbol{pts}$) plus a set of parameters specific to each element: for instance, the algorithm $\boldsymbol{shape_{cobogo}}$, which creates a type of *cobogó* brick[66], receives the thickness and width of the brick's outer frame, the radius size of its inner elements, and the pattern rule to create (Figure 7.17):

$$shape_{cobogo}(pts, thickness, width, r_{inner}, rule)$$

[66] A hollow ceramic block that allows ventilation and daylight control.

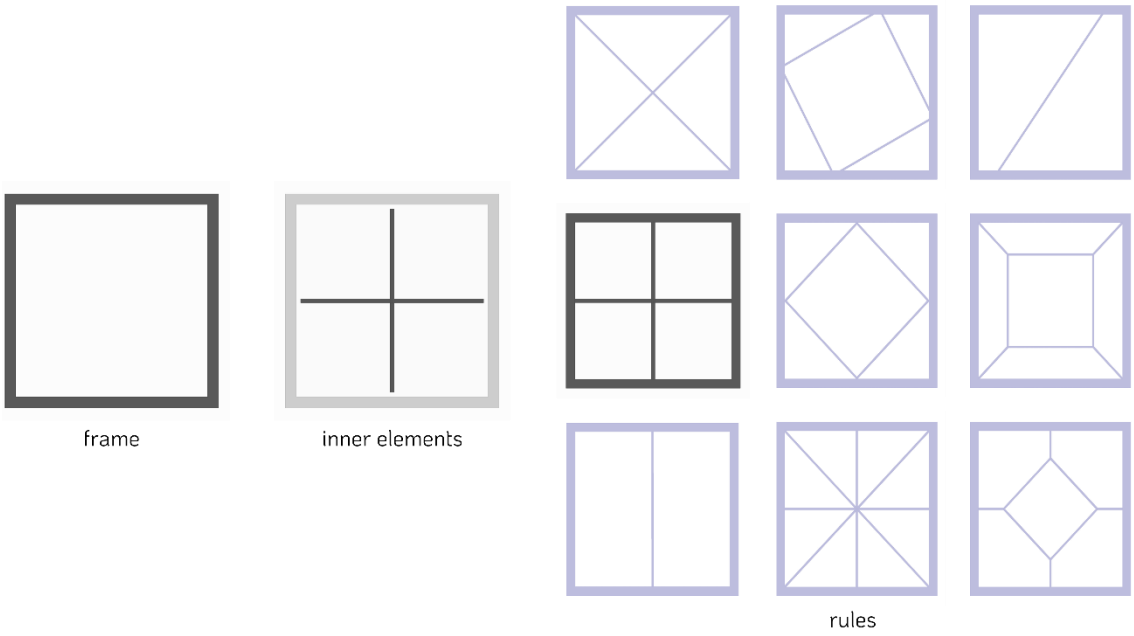


Figure 7.17. Conceptual representation of the algorithm $\boldsymbol{shape_{cobogo}}$.

As already mentioned, it is possible to manipulate the algorithms' parameters in different and independent ways. In the case of the algorithm $\boldsymbol{shape_{cobogo}}$, this allows for the application of multiple geometric rules in the same function composition and thus create a facade pattern with different cobogó elements. This scenario is illustrated in Figure 7.18 with four examples resulting from the same function composition: in all cases, a randomly selected geometric rule from a set of possible rules ($\boldsymbol{rules}$) is applied to each cobogó element, alternating between two rules, in the top-left example, between five rules in the top-right one, and four rules in the bottom ones. In the latter case, a random opaqueness level is also applied to each selected rule, which in this case ranges between five possible factors.

$$shape_{cobogo}.\left(grid_{squares}\left(straight(w,h)\right), thickness, width, r_{inner}, rules\right)$$

Note that, while in the top examples the surface points are uniformly distributed, originating equally sized *cobogó* elements, in the bottom ones their distribution is controlled by a random translation factor, thus producing elements of varying shapes and sizes.

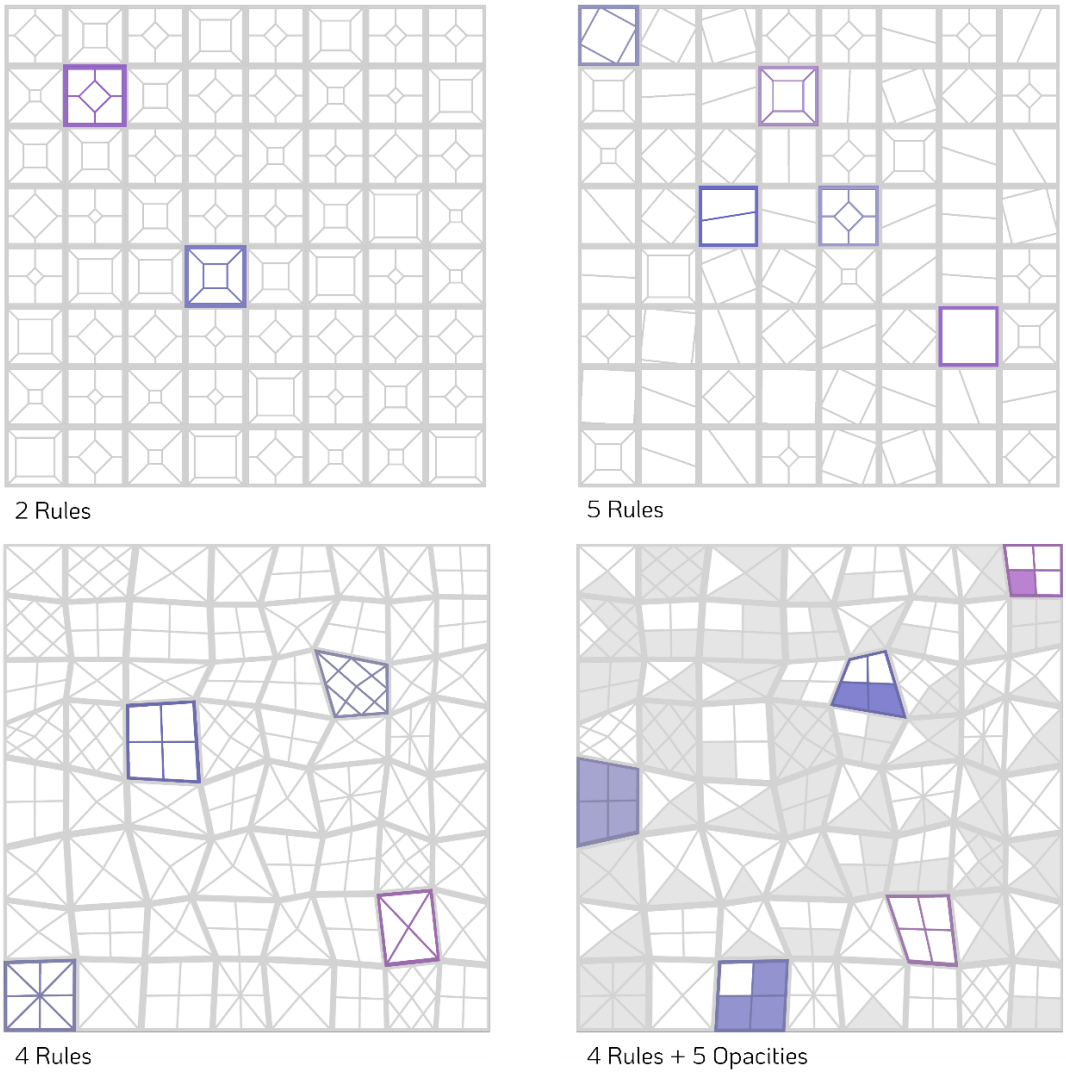


Figure 7.18. Four facade patterns resulting from the same algorithms but from different point distributions and geometric rules: in the top examples the surface points are regularly distributed and either two or five rules are used; in the bottom examples the surface points are randomly translated, and four rules are used with either the same or different opacity levels.

STRIPE-BASED ELEMENTS

There are several examples of facades whose pattern results from elements discrete in one of the surface's dimensions and continuous in the other (Figure 7.19). The framework names these geometries stripe-based, and it describes them mathematically through the HOF $shape_{stripe}$. This function takes as input an algorithm describing the stripe's central axis (pts) and another one defining its section shape ($shape_{section}$). As an example, when the section shape is a polygon, the result is an element similar to a polygonal bar or pillar (Figure 7.19 right). When it is, for instance, a line-like shape, the element created resembles a strip (Figure 7.19 left), and so on. Besides the previous arguments, the algorithm also

receives different geometric transformations (see section Transformation), which are mathematically represented as optional arguments ($args$ …):

$$shape_{stripe}(pts, shape_{section}, args\ \ldots)$$

Figure 7.19. Stripe-based facade elements: Community Library in La Molina by Gonzalez Moix Arquitectura, Peru (©Ramiro Del Carpio Fotografía); Aspen Art Museum by Shigeru Ban Architects, USA(©Michael Moran/OTTO); Striped Living by Group8asia, Switzerland (©Régis Golay); Nebuta-no-ie Warasse by Molo, d/dt, Frank La Riviere Architects, Japan (©Frank La Riviere).

To better understand this algorithm, consider the example of Figure 7.20. It starts with the selection of the algorithm $straight$ to obtain a matrix of $n{\times}m$ points describing a planar surface (example A). On this matrix we can then create either horizontally oriented stripes, if we provide the algorithm $shape_{stripe}$ with the matrix' rows, by using the algorithm $grid_{rows}$; or vertically oriented ones, if we use instead the matrix' columns (example B), by selecting the algorithm $grid_{columns}$. In both scenarios, we can simplify the mathematical representation of the resulting function composition as $shape_{stripe}.(ptss, shape_{section}, args\ \ldots)$, where $ptss$ are the result of the composition of the algorithm $straight$ with either the algorithm $grid_{rows}$ or $grid_{columns}$, and $args$ … are the supported optional

arguments regarding the geometric transformations allowed. Imagine we opt for the vertical stripes (example C). We must combine the algorithms $\boldsymbol{straight}$, $\boldsymbol{grid_{columns}}$, and $\boldsymbol{shape_{stripe}}$ in the following composition:

$$shape_{stripe}.\left(grid_{columns}\left(straight(w,h)\right), shape_{section}\right)$$

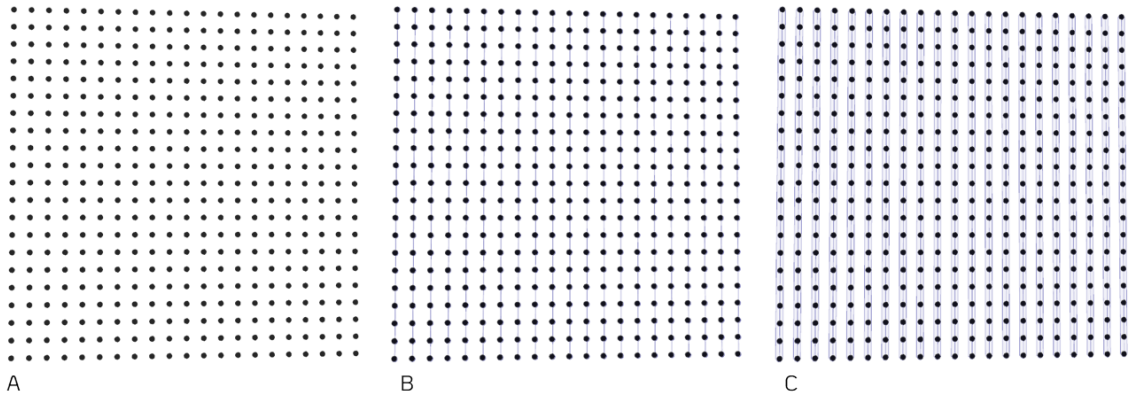


Figure 7.20. Continuous pattern conceptual representation: (A) surface points matrix ($\boldsymbol{ptss}$) and (B) its organization into $\boldsymbol{n}$ columns of points to then (C) generate the stripe-based elements on each one.

In this composition, the number of stripes to create is controlled by the output of the algorithm $\boldsymbol{grid_{columns}}$ since it depends on the number of columns on which the surface points ($\boldsymbol{ptss}$) were rearranged: if the latter are organized, for instance, in five sets of points vertically aligned, the result is a five-stripe pattern; if they are instead rearranged in 21 sets of points, as illustrated in Figure 7.20, the result is a 21-stripe pattern.

Similarly, the stripes' curvature also depends on the output of the algorithm $\boldsymbol{straight}$, i.e., the surface geometry on which they are created, resulting, in this case, in a set of $\boldsymbol{n}$ stripes without curvature. If we had selected instead a non-straight surface, the stripes' curvature would have been automatically adapted to it as illustrated in Figure 7.21 with a set of horizontal stripes.

There are more *Shape* algorithms than those presented in this section, some of them being illustrated in chapter 8. The next section elaborates on the functionalities available in the *Transformation* group.

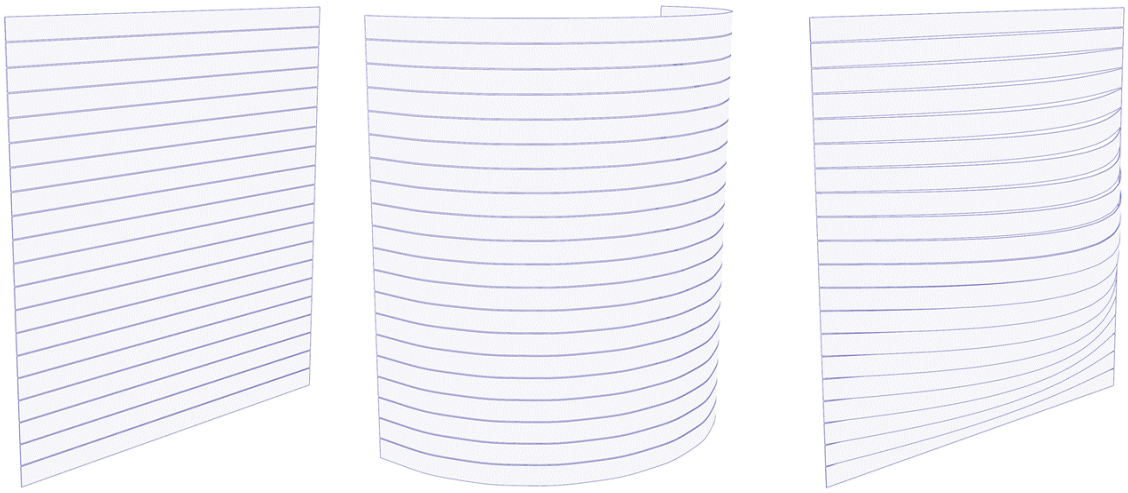


Figure 7.21. Three stripe-based patterns resulting from the algorithm $shape_{stripes}$ and different surface shapes.

TRANSFORMATION

Like the *Shape* group, this group also provides algorithms for both discrete and continuous patterns, which are organized into three main subgroups: *affine transformations*, *rule-based transformations*, and *continuous* transformations. The next sections elaborate on the algorithms available in each subgroup and how these can be combined with the other algorithms. They demonstrate that, despite containing a finite set of *Transformation* algorithms, the framework provides a high level of design flexibility, supporting the generation of a wide range of patterns resulting from the application of one or more geometric transformations to the same *Shape* element.

AFFINE TRANSFORMATIONS

This group contains algorithms inspired on affine transformations, namely *identity*, *scaling*, *rotation shear*, *reflection*, and *translation*. In general, these transformations map an affine space onto itself while preserving collinearity and the size relations of its points, straight lines, planes, and sets of parallel line segments. The angles between lines or distances between points can be changed, but the ratios of distances between points lying on a straight line cannot. In architecture, there are several building facades whose design results from different affine transformations (Figures 7.22-25).

Figure 7.22. Scaling examples: Louis Vuitton Store in Shenzhen, China (©Sérgio Gottsfritz pinterest); Quality Hotel Friends by Karolina Keyzer + Wingårdhs, Solna, Sweden (©Tord-Rickard Söderström); Cube Tube by SAKO Architects, Jinhua, China (©Misae HIROMATSU).

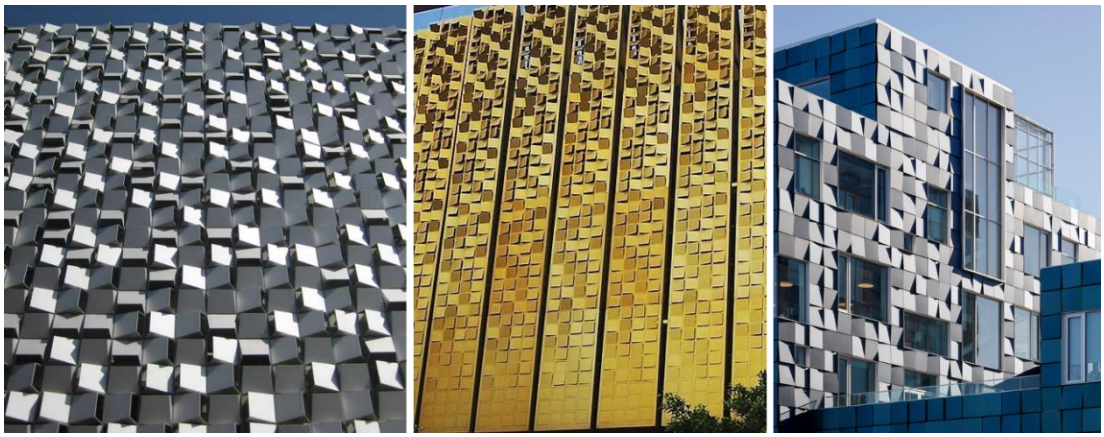


Figure 7.23. Rotation examples: Charles Street Multi Storey Car Park in Sheffield (©Adrian Welch); Nolan building in Melbourne (©Tim Dickson); Copenhagen International School Nordhavn by C.F. Møller (©Adam Mørk).

Figure 7.24. Shear examples: Victoria Gate by ACME, UK (©Jack Hobhouse); 700 Bourke Street, Melbourbe (©António Leitão); Ginza Place by Klein Dytham architecture + TAISEI DESIGN Planners Architects & Engineers, Japan (©Nacasa & Partners).

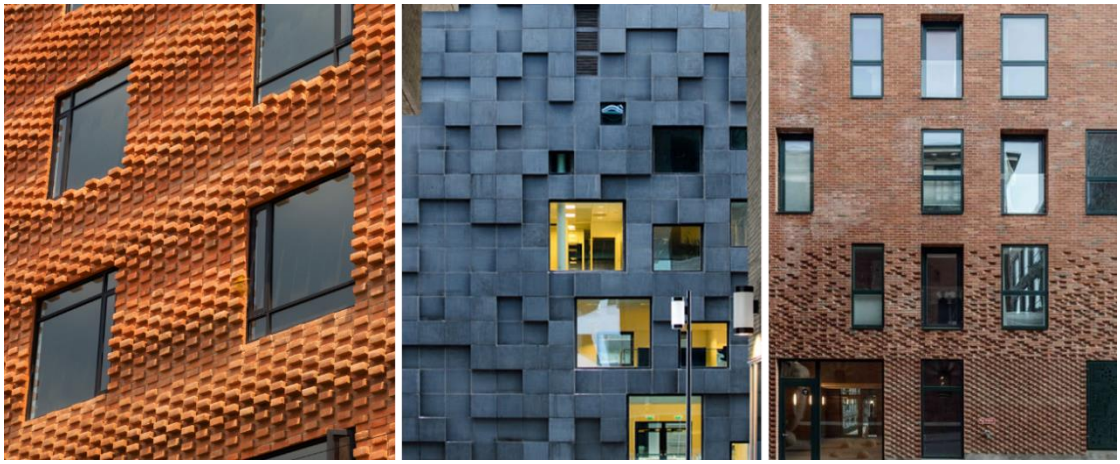


Figure 7.25. Translation examples: Z53 Social Housing by Michan Architecture + Grupo Nodus, Mexico (©Rafael Gamo); The Building C by Dark Arkitekter, Oslo (©Cavazos&Associates); Theodora House by ADEPT, Copenhagen (©Rasmus-Hjortshoj).

The algorithms available in this group include:

- identity ($\boldsymbol{T_{id}}$), to keep the original shape unchanged.
- scaling ($\boldsymbol{T_{scale}}$), to change distances between points according to a constant factor $\boldsymbol{k}$, compressing or enlarging shapes if $\boldsymbol{k < 1}$ or $\boldsymbol{k > 1}$, correspondingly.
- reflection ($\boldsymbol{T_{mirror}}$), to mirror shapes with respect to an axis or point.
- rotation ($\boldsymbol{T_{rotate}}$), to rotate shapes around an axis.
- shearing ($\boldsymbol{T_{shear}}$), to distort shapes parallel to an axis or plane.
- translation ($\boldsymbol{T_{translate}}$), to move shapes according to a displacement vector.

All these algorithms receive as input the surface points ($\boldsymbol{ptss}$), a specific position on the surface ($\boldsymbol{pt}$), and, except for $\boldsymbol{T_{id}}$, a factor controlling the transformation effect intensity ($\boldsymbol{k}$), being 0 and 1 the null and maximum effects, correspondingly. The remaining arguments depend on the transformation to apply.

The algorithm $\boldsymbol{T_{id}}$ represents the non-transformation of a shape. To understand its functioning, the mathematical notion of the identity function ($\boldsymbol{id}$) is introduced, which is a function that always returns the value used as argument: $\boldsymbol{id(x) = x}$. Similarly, when receiving a set of geometric entities as input, this algorithm returns the exact same entities with all their spatial relations and dimensions preserved. Although it seems to have little applicability, this functionality is actually very useful to deal with more complex combinations of algorithms, which will be further explained in the next section.

The algorithm $\boldsymbol{T_{scale}}$ scales a geometric entity according to (1) its position on the surface; (2) its distance to one or more *attractor* points or curves, which can be algorithmically or manually described

in the design tool by using, for instance, the algorithms $attractor_{points}$ and $attractor_{curves}$; (3) the point-in-polygon problem[67], where the set of surface areas can be described by using algorithms from the *Shape* category or manually by using the algorithm $affected_{areas}$; and (4) random rule(s), among others. For each case, T_{scale} receives the information needed to perform the transformation, which might be, respectively:

1. the direction(s) of the effect to produce, causing the scaling factor to either increase or decrease along them.
2. a set of attractors, which might be points, curves, etc., making the scaling factor vary according to their relative position to each transformed shape.
3. a set of surface areas, the scaling factor changing depending on whether the shape is or not contained within one of such areas.
4. random values or rules, making the scaling factor vary in a random way.

Mathematically, the framework represents this algorithm as $T_{scale}(ptss, pt, k, args\ ...)$, being $args\ ...$ the supported optional arguments. The same logic applies to the remaining algorithms of this group.

To better understand the applicability of these algorithms, consider the example of Figure 7.26, which results from the combination of the algorithms $shape_{regularPolygon}$, the one creating each polygonal shape (in this case, a squared shape, i.e., $n_{sides} = 4$), and $grid_{squares}$, the one arranging the surface positions in a squared grid. In this composition, each set of four points resulting from $grid_{squares}(ptss)$ is assigned to the algorithm $shape_{regularPolygon}$ by using broadcasting.

$$shape_{regularPolygon}\left(\begin{matrix} grid_{squares}(ptss), \\ n_{sides}, \\ r_{outer}, \\ \alpha \end{matrix}\right)$$

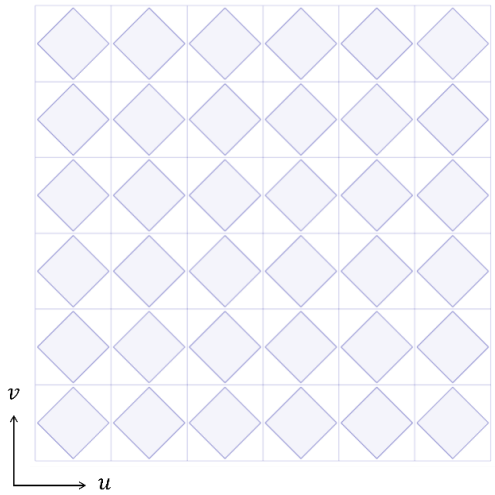


Figure 7.26. A pattern of squares distributed in a squared grid.

[67] Testing if a given point in the plane is inside, outside, or on the boundary of a surface area.

Imagine we want to increase the polygons' rotation in the u direction as illustrated in Figure 7.27. We combine the previous algorithms ($shape_{regularPolygon}$ and $grid_{squares}$) with T_{rotate}, which, in this case, returns a factor that increases with the surface length. As we make the latter algorithm specifically affect parameter α of the algorithm $shape_{regularPolygon}$, the squares' rotation angle increases in the u dimension. This is illustrated in the function composition of Figure 7.27 where $\overrightarrow{v_u}$ represents the direction of the transformation effect.

Note that, in this composition, $grid_{squares}$ and T_{rotate} inform the two parameters pts and α of the algorithm $shape_{regularPolygon}$, respectively: the former provides the position of each polygon and the latter changes their angle according to a factor. In turn, both algorithms are informed by the algorithm $straight$ but with a small difference: while $grid_{squares}$ takes all surface points at once, T_{rotate} receives a surface point at a time, corresponding to the position of the element to rotate.

$$shape_{regularPolygon}\cdot\begin{pmatrix} grid_{squares}(ptss), \\ n_{sides}, \\ r_{outer}, \\ \alpha \times T_{rotate}(ptss, pt, k, \overrightarrow{v_u}) \end{pmatrix}$$

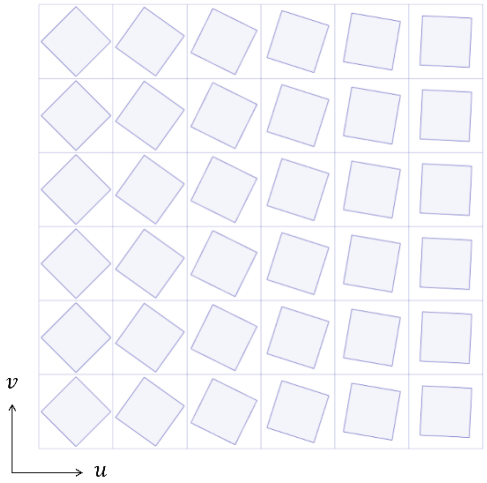


Figure 7.27. A pattern of squares distributed in a squared grid whose angle varies horizontally.

Following this logic, we can apply multiple transformations to the same pattern by combining more *Transformation* algorithms in function compositions. As an example, we can add a random scale variation to the previous rotation transformation, obtaining pattern A of Figure 7.28. For such, we select the algorithms (1) T_{scale} to control the parameter r_{outer} of $shape_{regularPolygon}$, and (2) T_{random} (see section Ruled-based Transformations) to make such factor vary according to a random rule. In the same way, we can continue applying more geometric transformations to the pattern and add, for instance, a translation variation to randomly move the squares' position as illustrated in Figure 7.28-B. To that end, we combine the algorithms $T_{translate}$ and T_{random} with the previous ones, manipulating the parameter pts of the algorithm $shape_{regularPolygon}$ in a random way.

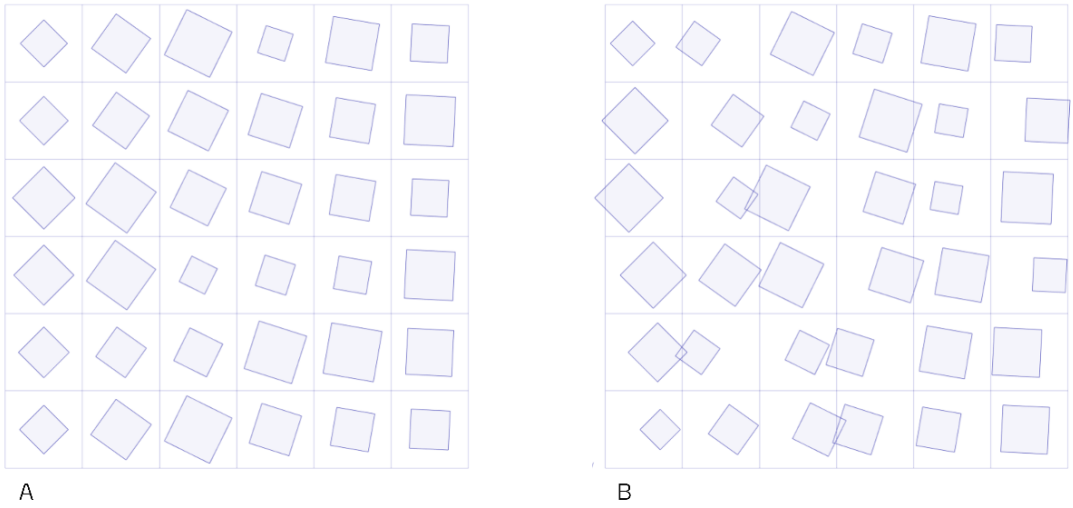


Figure 7.28. A pattern of squares distributed in a squared grid, whose angle varies horizontally and whose radius size (A) and center position (B) changes randomly.

Figure 7.29 presents some examples resulting from different combinations of *Transformation* and *Shape* algorithms. In examples A and B, for instance, we combine the algorithms $\boldsymbol{grid_{rhombus}}$, $\boldsymbol{shape_{star}}$, and $\boldsymbol{T_{scale}}$, obtaining a star-based pattern with decreasing radius sizes in the $\boldsymbol{u}$ dimension, in the first case, and random sizes, in the second. In either case, the algorithm $\boldsymbol{T_{scale}}$ manipulates two parameters of the algorithm $\boldsymbol{shape_{star}}$ proportionally, namely $\boldsymbol{r_{convex}}$ and $\boldsymbol{r_{concave}}$, creating star polygons of varying sizes but with the same size ratio between both radii. In example C, we use the same set of algorithms but, this time, we keep the parameter $\boldsymbol{r_{convex}}$ fixed, making the algorithm $\boldsymbol{T_{scale}}$ manipulate the parameters $\boldsymbol{r_{concave}}$ and $\boldsymbol{k_{apperture}}$ (aperture factor) of the algorithm $\boldsymbol{shape_{star}}$. In both cases, the scaling factor varies within a range of random values that either increase along the $\boldsymbol{u}$ dimension, in the former case, or are fixed, in the latter case.

Regarding the remaining examples (from D to F), we combine the algorithm $\boldsymbol{T_{scale}}$ with other *Shape* algorithms, namely $\boldsymbol{shape_{rosette}}$ (D), $\boldsymbol{shape_{rhombille}}$ (E), and $\boldsymbol{shape_{rhombitrihexagonal}}$ (F), and *Distribution* algorithms, namely $\boldsymbol{grid_{rhombus}}$, $\boldsymbol{grid_{trianglesHexagon}}$, and $\boldsymbol{grid_{hexagons}}$, correspondingly. In either case, the algorithm $\boldsymbol{T_{scale}}$ affects a size-related parameter of the shapes, making them gradually increase in either the $\boldsymbol{u}$ dimension (examples D and F) or $\boldsymbol{v}$ dimension (example E).

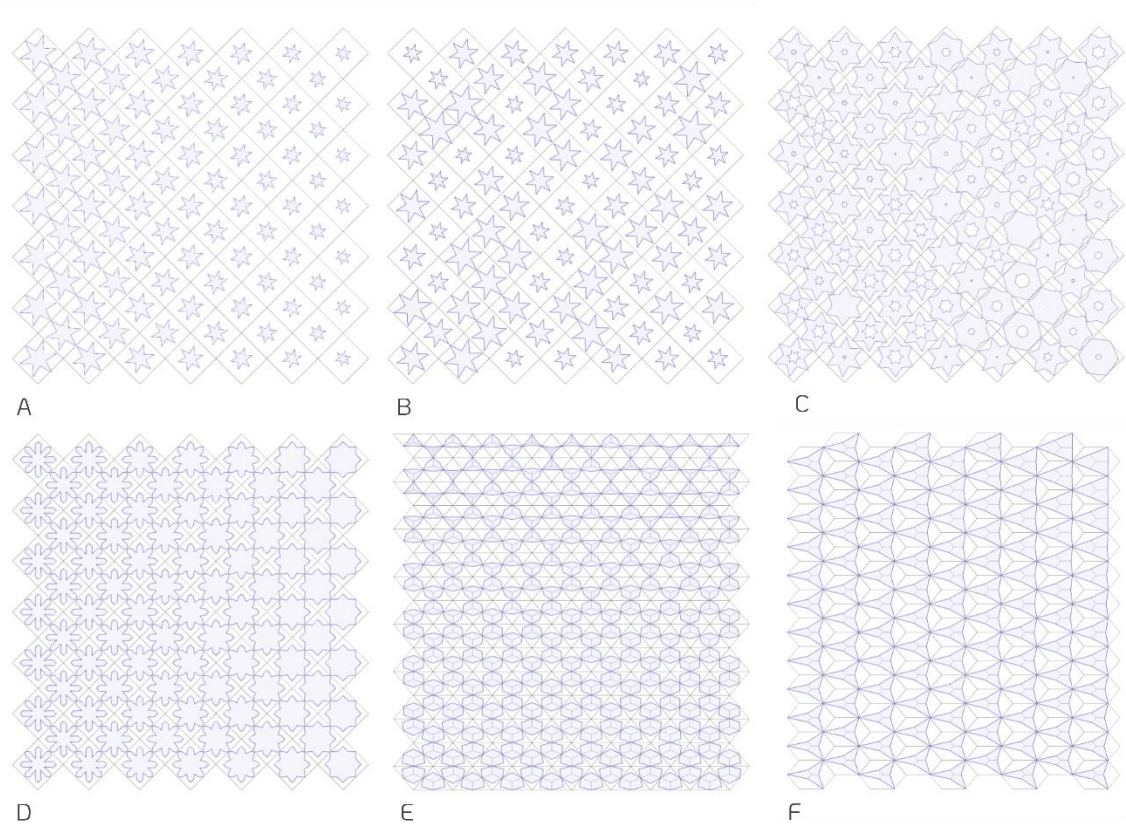


Figure 7.29. Conceptual representation of the algorithm T_{scale} combined with different *Distribution* and *Shape* algorithms: A to C combine $grid_{rhombus}$ and $shape_{star}$; D uses $grid_{rhombus}$ and $shape_{rosette}$; E applies $grid_{trianglesHexagon}$ and $shape_{rhombille}$; and F combines $grid_{hexagons}$ and $shape_{rhombitrihexagonal}$.

RULE-BASED TRANSFORMATIONS

This group contains algorithms to transform shapes according to different geometric rules found in real architectural examples (Figures 7.30-34), such as:

- Creating patterns combining multiple shapes – algorithm shape combination (T_{shapes}).
- Applying different colors or materials to the shapes – algorithm color application (T_{color}).
- Transforming shapes according to an image – algorithm pictorial effect ($T_{pictorial}$).
- Subdividing shapes according to one or more geometric rules – algorithm recursive subdivision ($T_{subdivide}$).
- Transforming the edges of a shape – algorithm edge deformation (T_{edge}).
- Adding randomness – algorithm randomness (T_{random}).

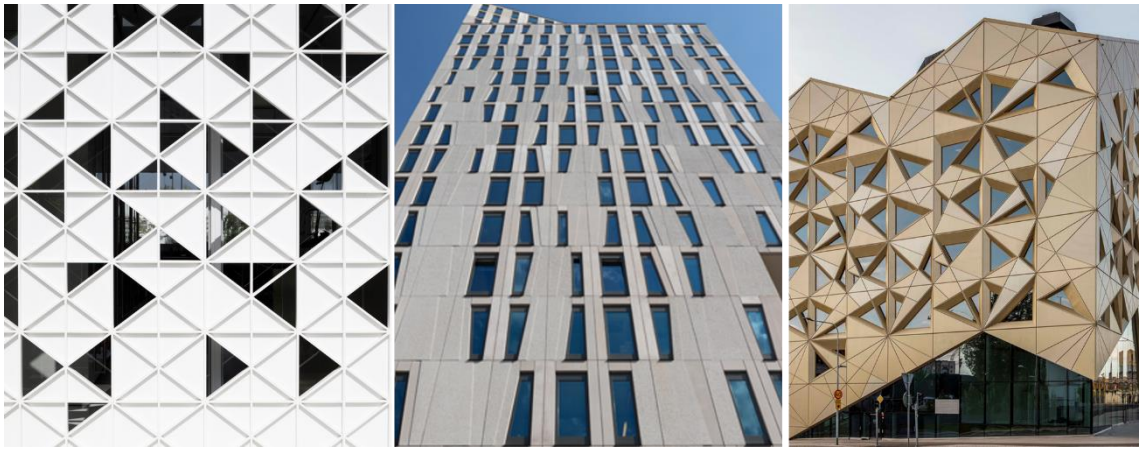


Figure 7.30. Shape combination examples: D3 House by Pitsou Kedem Architects, Israel (©Amit Geron); Offices Blaak 8 by GROUP A, Rotterdam (©Ossip van Duivenbode); Juvelen by Utopia Arkitekter AB (©Utopia).

Figure 7.31. Color application examples: Schuppen House by brandt+simon architekten, Berlin (©Michael Nast); Martinet School by Mestura Arquitectes, Spain (©Pedro Pegenaute); ABC Museum, Illustration and Design Center by Aranguren & Gallegos Architects, Spain (©Jesús Granada).

Figure 7.32. Picture-based examples: HELLO house by OOF! Architects, Melbourne (©Nic Granleese); Eskenazi Hospital parking structure by Rob Ley Studio (©Serge Hoeltschi); Administrative Building Textilverband / Behet Bondzio Lin Architekten, Germany (©Thomas Wrede).

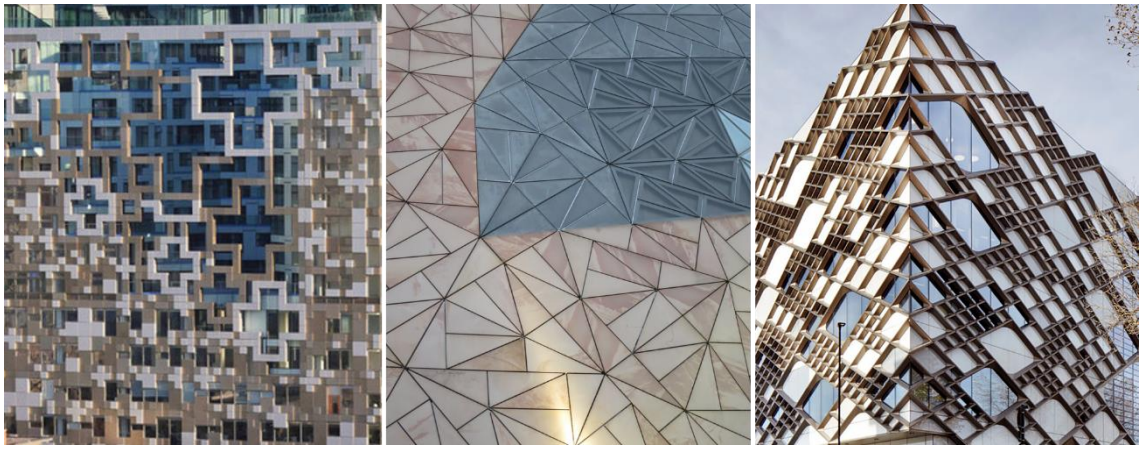


Figure 7.33. Recursive subdivision examples: the Cube by Make Architects, UK; Federation Square by LAB architecture studio, Melbourne (©António Leitão); Diamond building at the University of Sheffield by Twelve Architects, UK (©Jack Hobhouse).

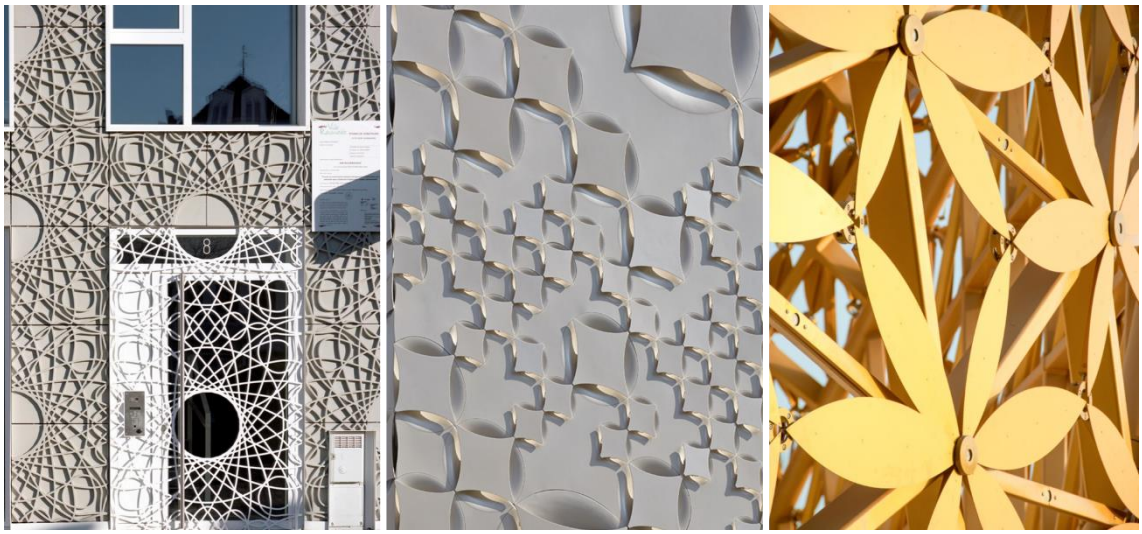


Figure 7.34. Edge deformation examples: Louis Vuitton's Tokyo store facade by Jun Aoki (©Daici Ano); The Filigrane by D'HOUNDT+BAJART architectes&associés (©Maxime Delvaux); Butterfly Pavilion by 3deluxe, UAE (©JoaquínBusch).

Compared to the previous group, these algorithms are mathematically more complex, requiring a different algorithmic combination from what we have seen so far. Therefore, before proceeding with their description, this section introduces an important mathematical concept for understanding their future function composition, namely the matrix of functions (MF), i.e., a matrix containing different functions. In practice, it allows us to apply more than one *Shape* or *Transformation* algorithm on the same geometric pattern, while controlling their order of application. Mathematically, the framework represents it as:

$$MF = \begin{bmatrix} f1 & f2 \\ g1 & g2 \end{bmatrix}, \text{ where } \begin{vmatrix} f_1 \in \mathcal{L}(U,U), f_2 \in \mathcal{L}(U,V), \\ g_1 \in \mathcal{L}(V,U), g_2 \in \mathcal{L}(V,V). \end{vmatrix}$$

In general, MFs are useful to iterate along two-dimensional domains, which is the case of our surface-related algorithms: both *Geometry* and *Distribution* algorithms produce two-dimensional matrices of points (MP). To deal with possible size differences between matrices, the available *rule-based* algorithms map the smallest size matrices along the largest size ones, iteratively applying the former to a submatrix of the latter of the same size. This process is illustrated in Figure 7.35 with three MFs of different sizes and whose colored squares represent different functions: when mapped along the MP below, the 2×2 matrix A affects 2×2 submatrices of the MP at a time, producing pattern 1; the same logic applies to both examples B and C. Note that, the framework supports MFs of any size, including non-squared matrices, row matrices, columns matrices, unit matrices, or even matrices covering the entire surface at once[68]. Having this knowledge, we can now focus on these algorithms' application.

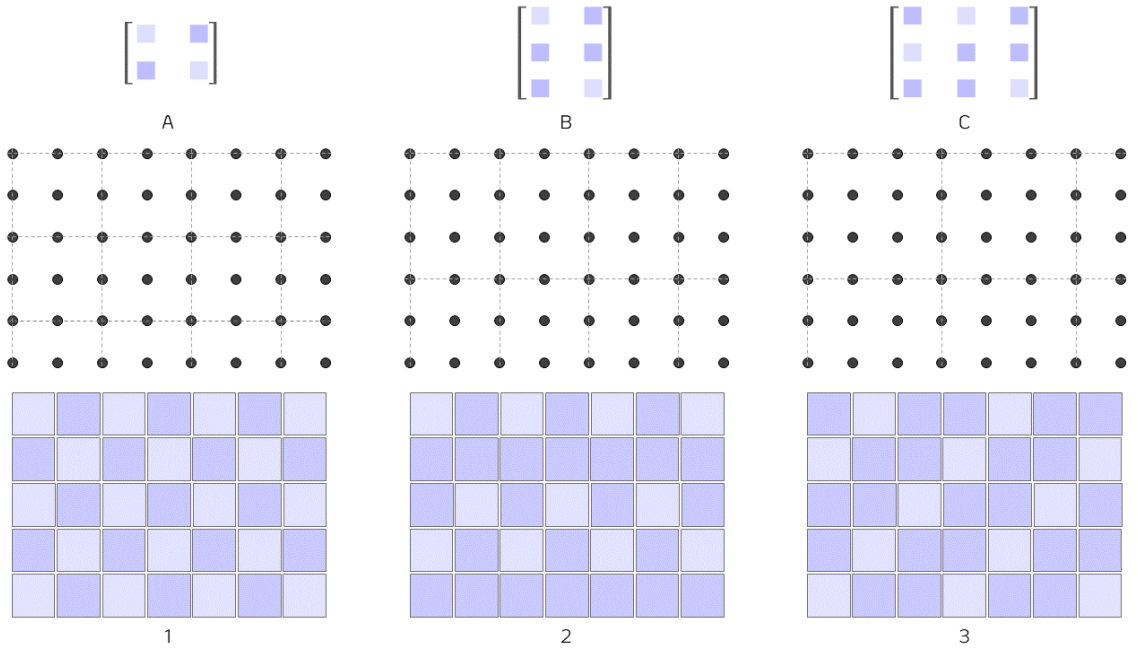


Figure 7.35. Conceptual representation of three MFs (A-C) applied to the same MP (black dots): mapping matrices A to C, whose sizes are 2×2, 2×3, and 3×3, on the same MP result in patterns 1 to 3, respectively.

In general, all *rule-based* algorithms receive three matrices: one containing the surface points ($\boldsymbol{ptss}$), another with one or more algorithms to apply ($\boldsymbol{M_{rules}}$), and a last one describing their order of application ($\boldsymbol{M_{pattern}}$):

[68] A common scenario when creating pictorial effects where the size of the matrix often matches that of the picture.

$$T_{ruleBased}(ptss, M_{rules}, M_{pattern})$$

Note that these three matrices are mathematically different, the first one ($ptss$) being a MP, the second (M_{rules}) a MF, and the last one ($M_{pattern}$) an integer matrix[69] (MI). The result is a new matrix with the algorithms of M_{rules} organized according to the sequence set by $M_{pattern}$: its integers identify which algorithm of M_{rules} should be applied at each position. Also note that this assignment of the integers contained in one matrix to the indexes of the other is represented here by the symbol ∘.

As an example, consider the matrices M_{rules} and $M_{pattern}$ illustrated in Figure 7.36: as the integers of $M_{pattern}$ (matrix 1) correspond to the indexes of the algorithms of M_{rules} (matrix 2), integer 1 therefore matches the algorithm f and integer 2 the algorithm g, resulting in the matrix 3. As the latter's size often differs from that of $ptss$ (matrix 4), it is then resized to become a matrix of the same size as $ptss$, containing the algorithms of M_{rules} arranged according to the pattern set by $M_{pattern}$ (matrix 5). Note that the framework supports M_{rules} containing from a single algorithm, creating the same shape on each surface position, to the number matching the size of the matrix $ptss$, applying at its extreme a different algorithm to each surface position.

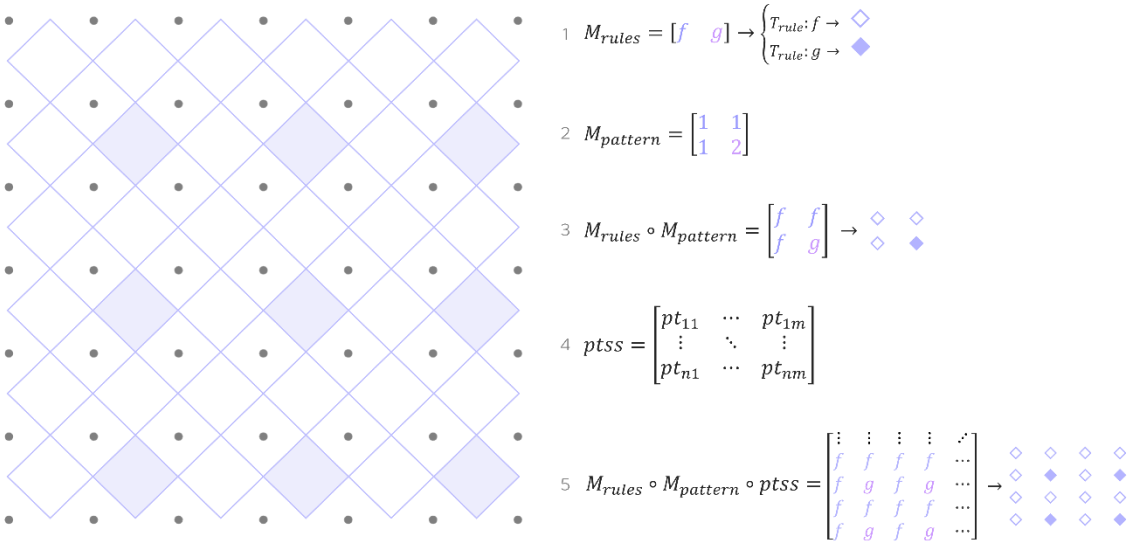


Figure 7.36. Conceptual representation of the three input matrices $ptss$, M_{rules}, and $M_{pattern}$ and their combination (right) and the resulting pattern (left).

To further illustrate their practical application, this section presents, in a step-by-step process, several examples combining *rule-based* algorithms with those of the previous sections. Let us start with a simple pattern made of several vertically and horizontally aligned squares (Figure 7.37-A). Imagine we want to

[69] A matrix containing only integers, e.g., binary matrices, the identity matrix, the matrix of ones, etc.

replace some squares with circles as illustrated in the example B of Figure 7.37. We select the algorithm T_{shapes} and we provide it with three matrices:

1. One containing the surface points ($ptss$), which in this case is a squared grid of points represented by the algorithm $grid_{squares}$.
2. Another including the two shapes (squares and circles) composing the pattern (M_{shapes}), which correspond to the algorithms $shape_{regularPolygon}$ and $shape_{circle}$.
3. A last one describing their order of application ($M_{pattern}$), which corresponds to a MI where integer 1 represents the algorithm $shape_{regularPolygon}$ and integer 2 the algorithm $shape_{circle}$.

$$T_{shapes}\left(\begin{array}{c} grid_{squares}\big(straight(w,h)\big), \\ [shape_{regularPolygon} \quad shape_{circle}], \\ \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \end{array}\right)$$

The result is a bigger matrix with the size of $ptss$ and containing the shapes in M_{shapes}, which are illustrated bellow with ■ and ●, organized according to the instructions of $M_{pattern}$:

$$\begin{bmatrix} ■ & ■ & ■ & ■ & \cdots \\ ● & ■ & ● & ■ & \cdots \\ ■ & ■ & ■ & ■ & \cdots \\ ■ & ● & ■ & ● & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

To increase its complexity, we can now apply different geometric transformations to the algorithms of M_{shapes} and obtain a pattern similar to that of Figure 7.37-C, if we combine the algorithms T_{scale} and T_{random} with the algorithm $shape_{regularPolygon}$, obtaining a pattern made of randomly sized squares but fixed sized circles; and to that of Figure 7.37-D, if we combine the previous algorithms with both $shape_{regularPolygon}$ and $shape_{circle}$, making the size of both shapes vary randomly.

This ability to keep some shapes unchanged, while transforming the others, is a good example of the T_{id} application. That is, when appearing in a MF containing different *Transformation* algorithms, the shapes to which this algorithm is applied remain unchanged because it returns the exact same shape: $T_{id}(shape) = shape$. Figure 7.37-C is an example of that: while we combine the algorithm $shape_{regularPolygon}$ with both T_{scale} and T_{random} to scale the squares in a random way, we combine the algorithm $shape_{circle}$ with T_{id} to keep the circles' size unchanged.

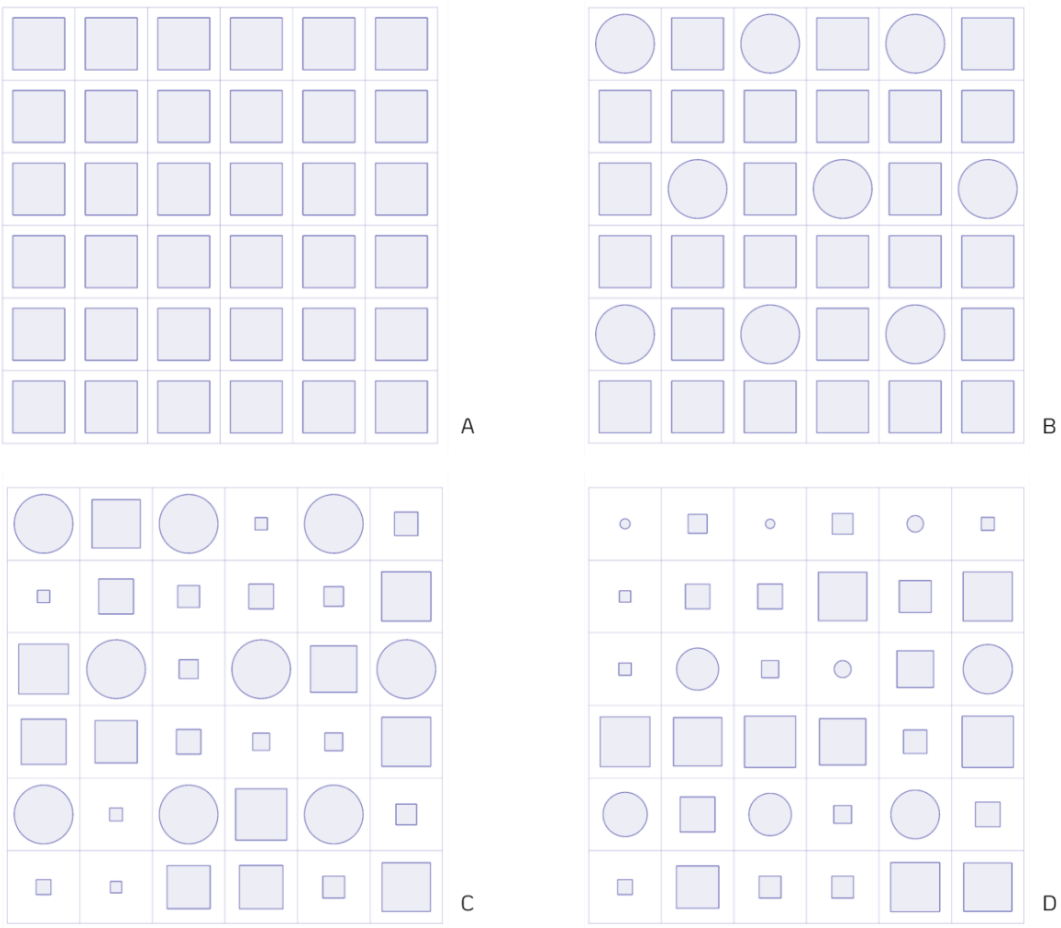


Figure 7.37. Conceptual representation of the algorithm T_{shapes}: A. original pattern; B. replacement of some squares by circles; C. application of a random scale variation to the squares; D. application of a random scale variation to both squares and circles.

Besides differently combining various *Shape* algorithms, the algorithm T_{shapes} also allows the creation of voids instead of shapes or simply creating no shape. In both cases, it is the matrix $M_{pattern}$ that informs the algorithm T_{shapes} about which option to execute, distinguishing shape subtraction from shape creation with a negative integer and the absence of shapes with zeros. As an example, consider Figure 7.38, where A is the pattern developed in the previous figure, and B and C are that same pattern but with punctual squared holes rather than shapes, in the first case, and no shapes, in the second. To obtain pattern B, we combine the previous $M_{pattern}$ with one informing about the existence of holes at every two diagonals, which the framework mathematically represents as:

$$\begin{bmatrix} +1 & +1 & -1 \\ +1 & -1 & +1 \\ -1 & +1 & +1 \end{bmatrix}$$

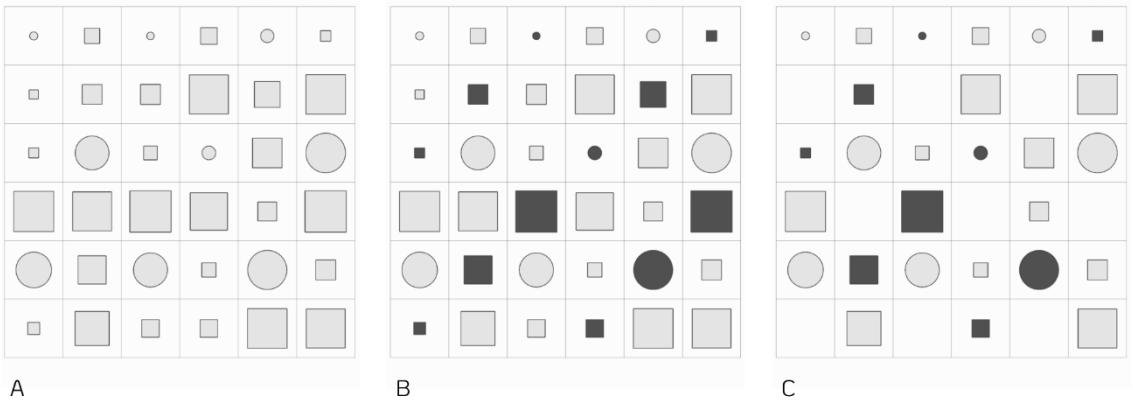


Figure 7.38. Conceptual representation of the algorithm T_{shapes}: original pattern (A) and the same pattern with punctual voids/holes (black shapes) (B) and with both punctual voids/holes and absent elements (C).

The result of their combination is the following function composition where integer -1 means subtracting the algorithm of index 1 in M_{shapes}, i.e., $shape_{regularPolygon}$, and integer -2 the algorithm of index 2, namely $shape_{circle}$:

$$\begin{bmatrix} +1 & +1 & -1 \\ +1 & -1 & +1 \\ -1 & +1 & +1 \end{bmatrix} \circ \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 2 & 1 \\ 1 & 1 \end{bmatrix} \circ \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ p_{n1} & \cdots & \cdots & p_{nm} \end{bmatrix} \rightarrow \begin{bmatrix} -1 & 1 & 1 & -1 & \cdots \\ 2 & -1 & 2 & 1 & \cdots \\ 1 & 1 & -1 & 1 & \cdots \\ -1 & 2 & 1 & -2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

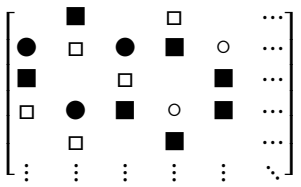
By providing the T_{shapes} with the previous matrix, we obtain a bigger one with the shapes of M_{shapes} organized according to $M_{pattern}$ together with information about their creation or subtraction, which are represented with ■/● and □/○, respectively:

$$T_{shapes}\left(\begin{matrix} grid_{squares}(straight(w,h)), \\ [shape_{regularPolygon} \quad shape_{circle}], \\ \begin{bmatrix} -1 & 1 & 1 & -1 & \cdots \\ 2 & -1 & 2 & 1 & \cdots \\ 1 & 1 & -1 & 1 & \cdots \\ -1 & 2 & 1 & -2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \end{matrix} \right) \rightarrow \begin{bmatrix} \square & \blacksquare & \blacksquare & \square & \cdots \\ \bullet & \square & \bullet & \blacksquare & \cdots \\ \blacksquare & \blacksquare & \square & \blacksquare & \cdots \\ \square & \bullet & \blacksquare & \circ & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

To obtain example C, we provide the algorithm T_{shapes} with a different $M_{pattern}$ where $\mathbf{0}$ represents the creation of no shape:

$$M_{pattern} = \begin{bmatrix} +1 & +1 & -1 & +1 & +1 & -1 \\ 0 & -1 & 0 & +1 & 0 & +1 \\ -1 & +1 & +1 & -1 & +1 & +1 \\ +1 & 0 & -1 & 0 & +1 & 0 \\ +1 & -1 & +1 & +1 & -1 & +1 \\ 0 & +1 & 0 & -1 & 0 & +1 \end{bmatrix}$$

As a result, nothing is created in the locations matching the zeros:



The same logic applies to the remaining *rule-based* algorithms, which are illustrated in the following set of examples.

Imagine we want to apply different colors to the pattern A of Figure 7.38. We select the algorithm T_{color} and we provide it with the same matrix $ptss$ plus two new matrices, one with the colors to apply (M_{colors}) and another with their pattern of application ($M_{pattern}$):

$$T_{color}(ptss, M_{colors}, M_{pattern})$$

Then, we combine this algorithm with T_{shapes}, obtaining a new matrix of the same size as $ptss$ merging the information of the following matrices, where the first two are the inputs of T_{shapes} and the latter two the inputs of T_{colors}:

$$M_{shapes} = [\square \quad \bigcirc] \qquad M_{pattern} = \begin{bmatrix} 1 & 1 & 1 & 2 \\ 1 & 2 & 1 & 1 \end{bmatrix} \qquad M_{colors} = [\blacksquare \ \blacksquare] \qquad M_{pattern} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix}$$

Regarding the first two matrices, as integer 1 matches symbol $\square$ and integer 2 the symbol $\bigcirc$, the result of their combination is:

$$M_{shapes} \circ M_{pattern} \rightarrow \begin{bmatrix} \square & \square & \square & \bigcirc \\ \square & \bigcirc & \square & \square \end{bmatrix}$$

The same happens with the other two matrices, where integer 1 matches color lilac and integer 2 color grey, the result of their composition being:

$$M_{colors} \circ M_{pattern} \rightarrow \begin{bmatrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{bmatrix}$$

By providing the algorithms T_{shapes} and T_{colors} with these two matrices, respectively, while combining them in a function composition, we obtain a larger matrix merging all the above information, whose application is illustrated in Figure 7.39.

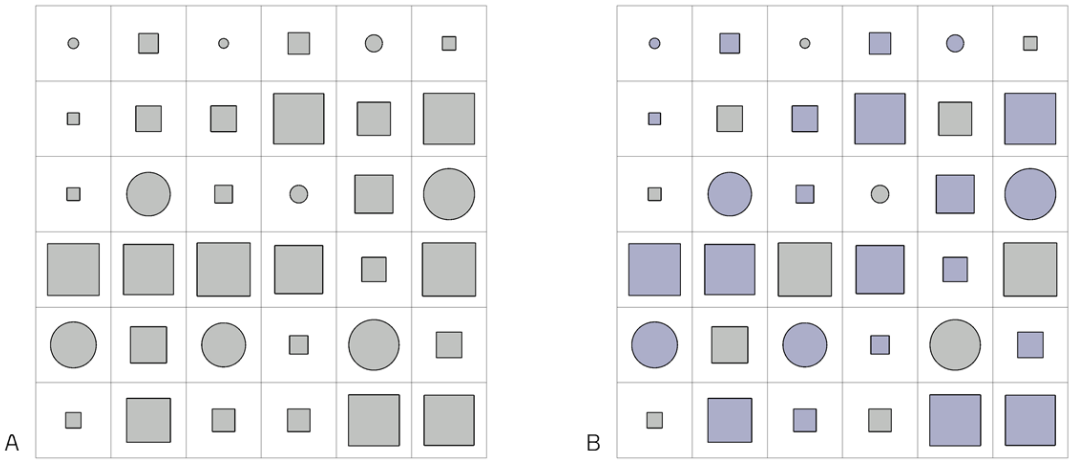


Figure 7.39. Conceptual representation of the algorithms T_{shapes} and T_{colors} composition: the same pattern before (A) and after applying different colors to its elements (B).

As another example, consider the creation of pictorial effects resulting from either strategically scaling, moving, or rotating facade elements as illustrated in Figure 7.40: in the top-left example, for instance, the bricks are simultaneously rotated and moved perpendicularly to the facade surface to create the diamond shape weaving pattern; in both top-center and top-right examples, the bricks are strategically rotated to produce, in the first case, a rhombus pattern and, in the second, a pattern resembling a "basket filled with giant grapes"[70]; and in the bottom examples, the pictorial effect is obtained through differently scaled perforations.

To produce such examples, we select the algorithm $T_{pictorial}$ and provide it with the matrix $ptss$ containing the surface points, plus two new matrices, one with the transformation(s) to apply and the other with the pictorial effect to create:

$$T_{pictorial}(ptss, M_{transformations}, M_{pattern})$$

In practice, $M_{pattern}$ contains the information about how the algorithm(s) of $M_{transformations}$ should perform at each position ($ptss$). As an example, if the matrix $M_{transformations}$ contains the algorithm T_{rotate}, the values of $M_{pattern}$ will be different rotation angles. In case it contains the algorithm T_{scale}, the latter's values will instead be different scaling factors. When executed, the algorithm $T_{pictorial}$ maps the transformation(s) available in $M_{transformations}$ along the surface's shapes according to the factors set in $M_{pattern}$.

[70] https://www.archdaily.com/260612/winery-gantenbein-gramazio-kohler-bearth-deplazes-architekten (Retrieved on <July 12th 2022>).

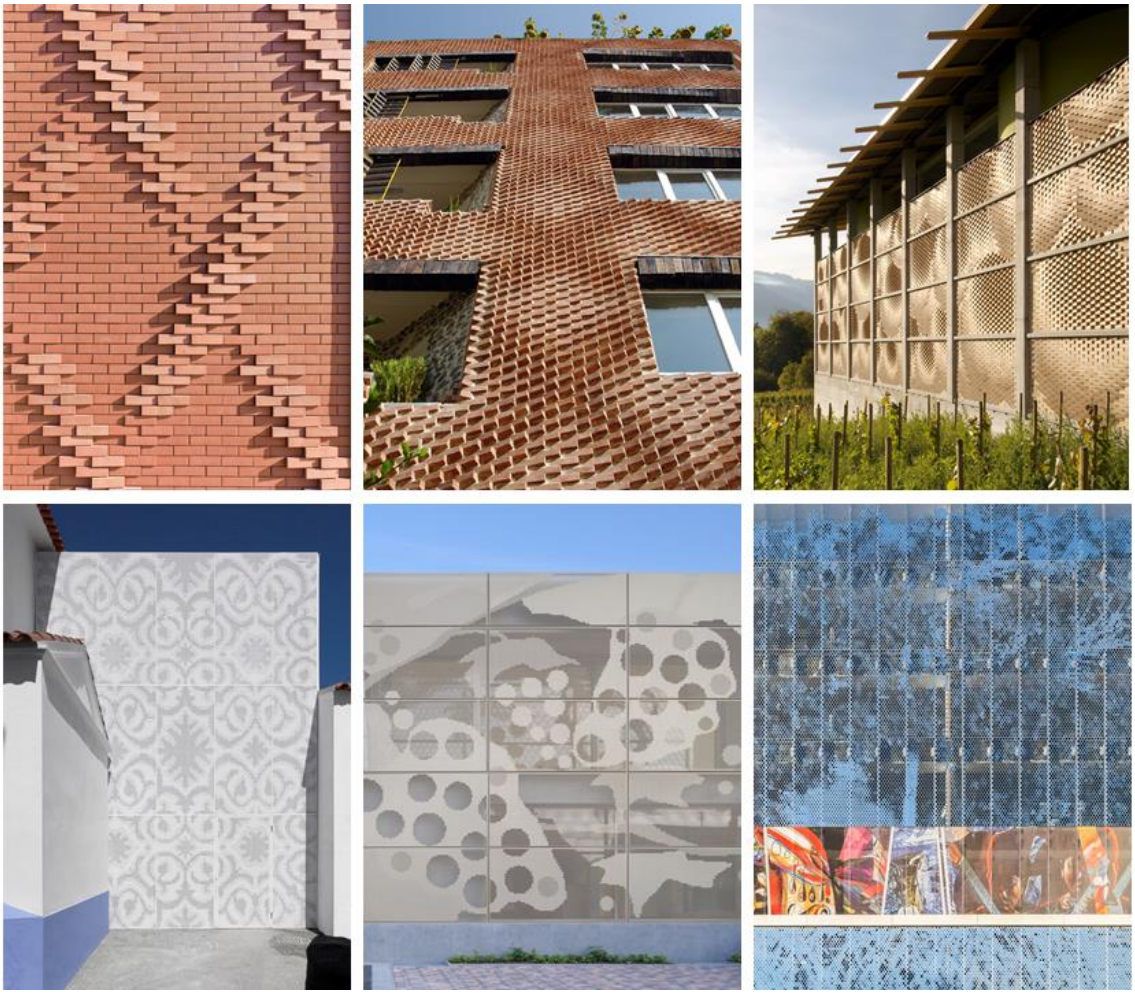


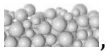
Figure 7.40. Pictorial facade patterns: House for Solidarity by Ellenamehl architects, France (©Hervé Ellena); APT. no7 by Sstudiomm, Iran (©Sstudiomm); Winery Gantenbein by Gramazio & Kohler + Bearth & Deplazes Architekten, Switzerland (©Gramazio & Kohler, Ralph Feiner); Tapestry Museum in Arraiolos by CVDB arquitectos, Portugal (©Fernando Guerra | FG+SG); WG Sasbach by Amann|Burdenski|Munkel Architekten, Germany (©Yohan Zerdoun); 3 MiamiCentral by AECOM, USA (©Poma Architectural Metals).

Imagine we want to create the top-right facade pattern of Figure 7.40. In addition to the surface points ($\boldsymbol{ptss}$), we need to provide the algorithm $\boldsymbol{T_{pictorial}}$ with a row matrix containing the algorithm $\boldsymbol{T_{rotate}}$ and another matrix containing the rotation angles of each surface position. As the latter values depend on the selected image's pixels, manually producing this matrix would require us to, first, assign a factor to each RGB value and, second, identify the RGB values of each pixel. Considering that each image typically has thousands of pixels, this would be a time-consuming and hardworking task.

To avoid this tiresome and time-consuming task, the framework provides the algorithm $\boldsymbol{pixelmap_{image}}$ to automatically convert PNG or JPEG files into a matrix containing their RGB values. This functionality takes as input an image ($\boldsymbol{image}$) and the surface points ($\boldsymbol{ptss}$) on which it will be applied. It then collects the RGB values of the former's pixels, storing them in a matrix whose size

matches that of the given surface: in case its domain is smaller than the picture size, the user can decide whether to calculate the average value between adjacent pixels or not; otherwise, the algorithm calculates new values that approximate the existing ones. When provided to the algorithm $T_{pictorial}$, the latter translates the received RGB values into factors by considering, among others, (1) the surface dimensions, (2) the number of elements, (3) the size of the elements, and (4) the intensity of the pictorial effect. As an example, when applying a 100x100 pixels picture to control the rotation angle of 50x40 bricks, the algorithm automatically adjusts the size of the former matrix to match the size of the latter. Using this functionality, we can therefore use the selected picture as a direct input for the algorithm $T_{pictorial}$ and easily and effortlessly obtain the desired pictorial effect.

Regarding the previous example, we could therefore describe it as

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

obtaining the pictorial pattern of Figure 7.41.

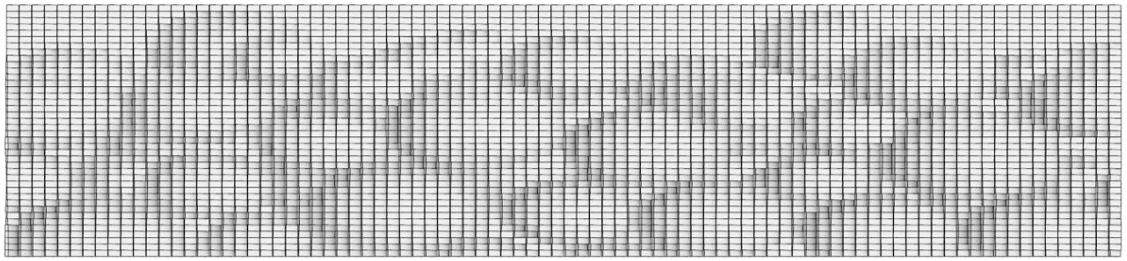


Figure 7.41. Pictorial effect using differently rotated standard bricks.

Using the same input image, we can now easily test other possibilities, such as producing the same pictorial effect by using differently colored bricks (Figure 7.42-A), if we replace T_{rotate} with $T_{colours}$; protruded bricks (Figure 7.42-B), if we use the algorithm $T_{translate}$ instead; or sized bricks (Figure 7.42-C), if we select the algorithm T_{scale}.

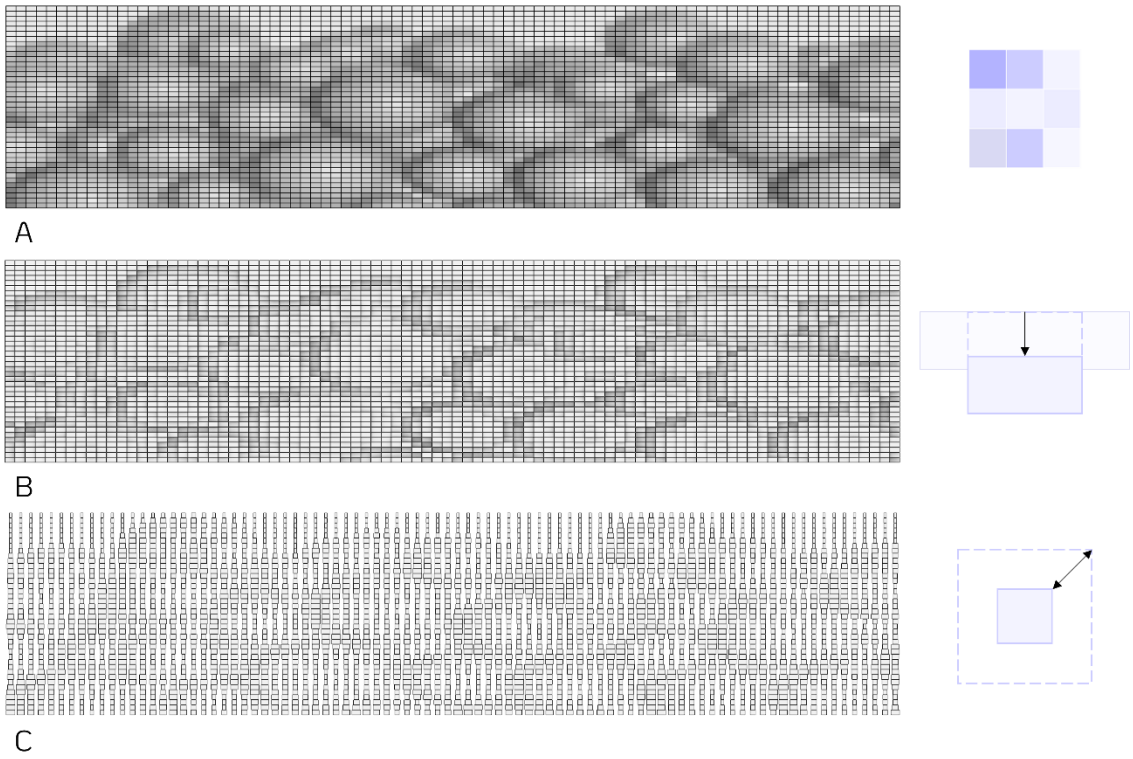


Figure 7.42. The same pictorial facade design effect using different brick (A) colors, (B) protrusions, and (C) sizes.

Now consider the facade designs of Figure 7.33. To obtain such examples, we select the algorithm $T_{subdivide}$, which receives as input a matrix with the surface points ($ptss$), another one with the *finite subdivision rule(s)*[71] to apply ($M_{subdivideRules}$), and a last one with their order of application ($M_{pattern}$). As this algorithm applies the rule(s) of $M_{subdivideRules}$ to each shape recursively, it must be informed about the number of iterations to perform to prevent it from executing the same rule endlessly. For such, it receives an additional argument, a matrix informing about the level of recursion to apply at each surface position ($M_{recursion}$):

$$T_{subdivide}(ptss, M_{subdivideRules}, M_{pattern}, M_{recursion})$$

Note that, in this algorithm, the first argument is a MP, the second is a MF, and the last ones are MIs containing the indexes of the rules available in $M_{subdivideRules}$ and their recursion levels. In practice, while the integers of $M_{pattern}$ inform about the subdivision rule to apply at each position ($ptss$), those of $M_{recursion}$ inform about the number of recursive steps to perform.

As an example, imagine we want to subdivide a triangular tiling [369] into increasingly smaller triangles as illustrated in Figure 7.43. We select the algorithm ($T_{subdivide}$) and provide it with a set of

[71] Subdivision of a shape into recursively smaller shapes.

surface points ($\boldsymbol{ptss}$), which in this case are distributed in a triangular grid; the subdivision rule(s) to apply ($\boldsymbol{M_{subdivideRules}}$), which in this case is a single algorithm; their pattern of application ($\boldsymbol{M_{pattern}}$), which in this case is a single integer; and finally their recursion level ($\boldsymbol{M_{recursion}}$):

$$T_{subdivide}(ptss, M_{subdivideRules}, M_{pattern}, M_{recursion})$$

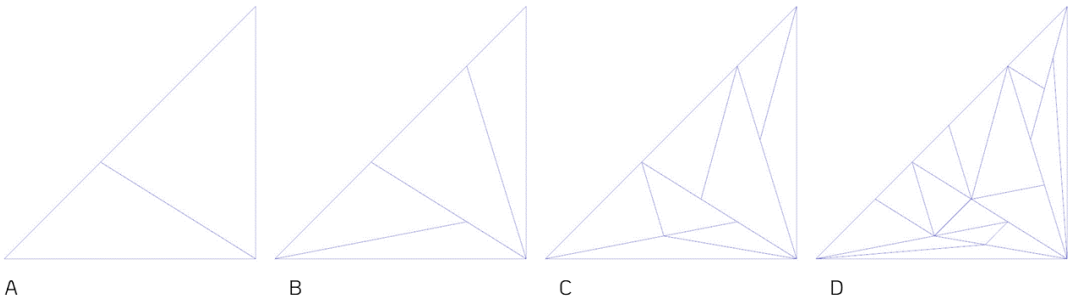


Figure 7.43. An example of a finite subdivision rule: rule A applied twice (B), thrice (C) and four times (D).

In case we set, for instance, $\boldsymbol{M_{recursion} = [0]}$, no subdivision rule is applied, obtaining the top-left example of Figure 7.44. If we instead use $\boldsymbol{M_{recursion} = [1]}$, the rule is applied once, originating the top-right example of the same figure. Lastly, if we set $\boldsymbol{M_{recursion} = [2]}$ or $\boldsymbol{M_{recursion} = [3]}$, the rule is applied twice or trice, respectively producing the bottom left and right examples.

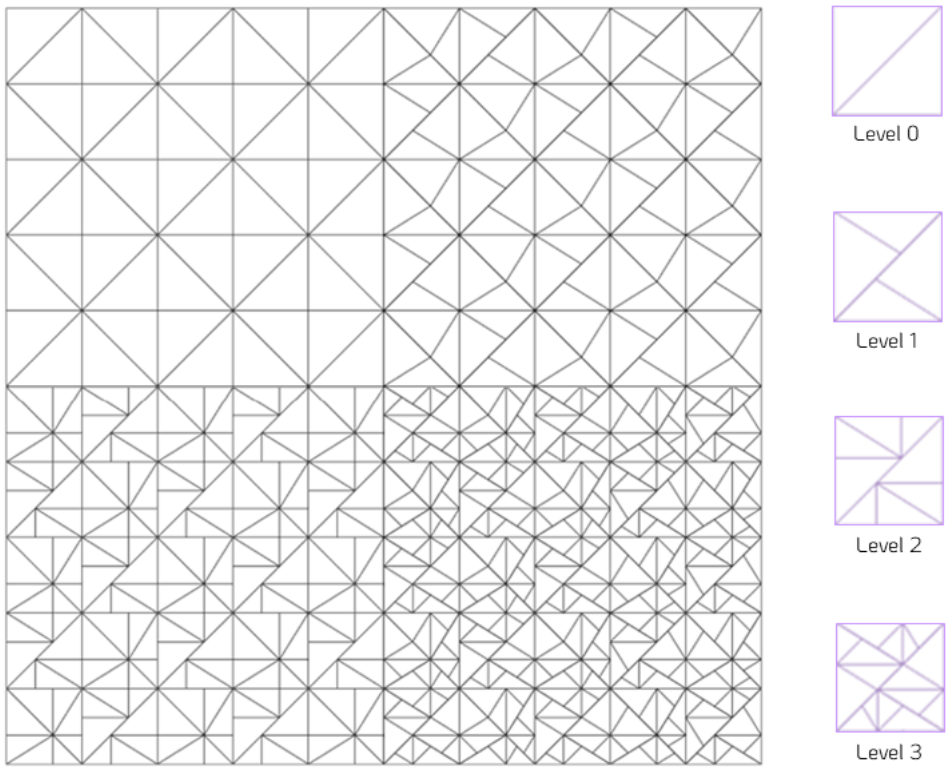


Figure 7.44. The previous subdivision rule applied to a triangular grid with different levels of recursion.

Figure 7.45 illustrates the result of providing the algorithm $T_{subdivide}$ with the same subdivision rule and different grid configurations, such as those produced with the *Distribution* algorithms $grid_{trianglesXY}$ (example A), $grid_{trianglesY}$ (example B), and $grid_{trianglesHexagon}$ (example C).

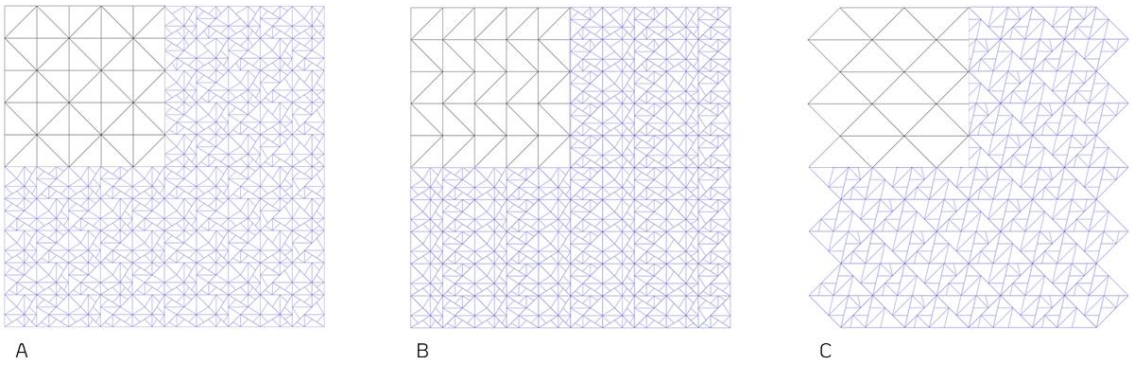


Figure 7.45. The same subdivision rule applied thrice to different grid configurations.

To add a certain level of randomness to these patterns, as it happens in some architectural examples, the algorithm $T_{subdivide}$ was extended with a Boolean parameter ($random$) that, when set as true, adds some randomness to the rules' level of recursion ($M_{recursion}$) and/or pattern of application ($M_{pattern}$). In the former case, it allows setting the range of values between which the recursion level can vary. In the latter case, it allows defining the ratio between the rules of $M_{subdivideRules}$. Both scenarios are illustrated in Figure 7.46 with three examples using the same grid configuration: in the first row we apply rule I with a level of recursion of one, two, and randomly varying between one and three; in the second row, we use rule II with the same recursion levels; and in the last row, we randomly apply either rule I or II in a fifty-fifty chance with the same levels of recursion.

Figures 7.47-48 illustrate further examples combining different finite subdivision rules and grid configurations, with varying levels of recursion and randomness.

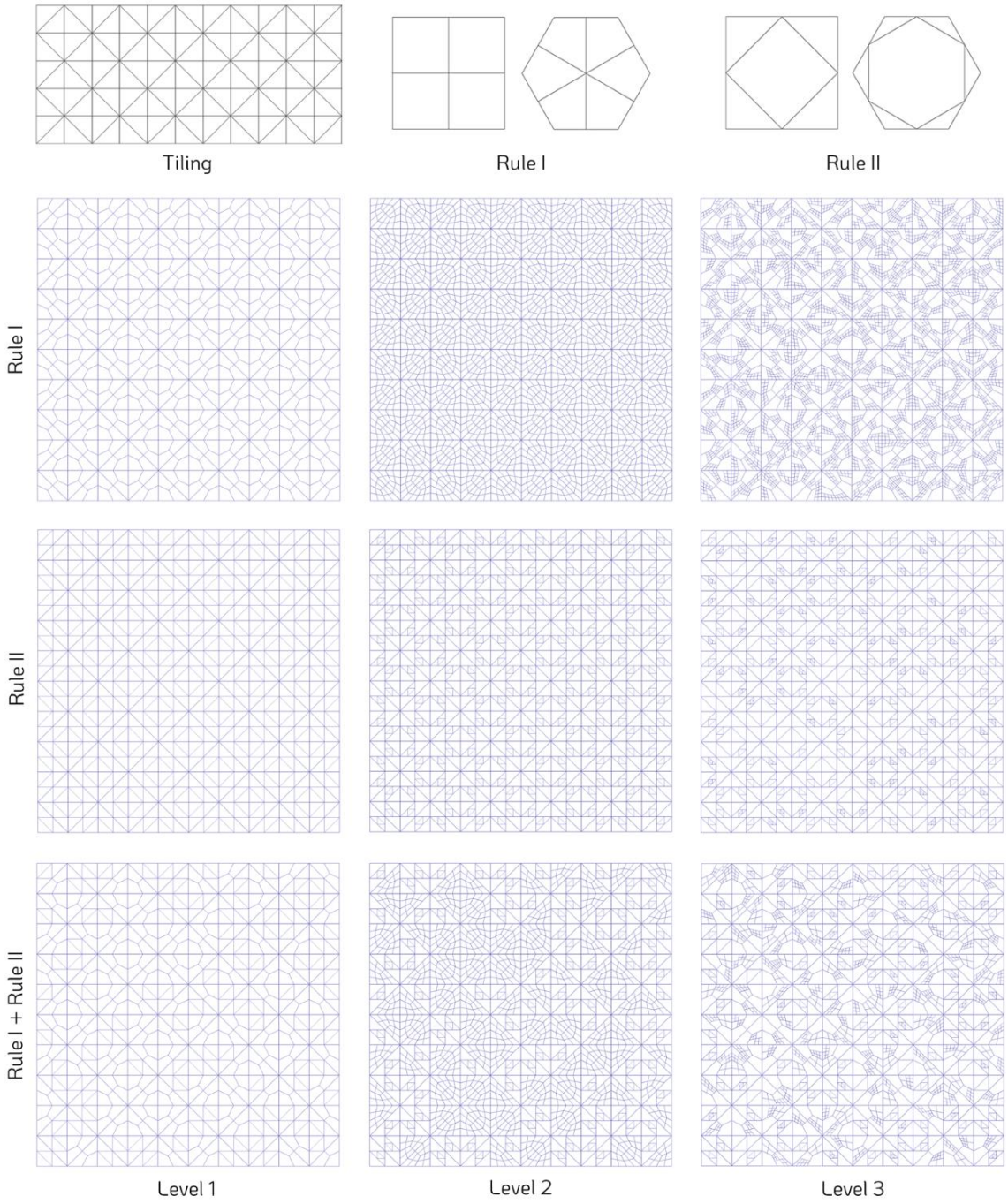


Figure 7.46. Top: the original tiling and the subdivision rules applied to it; bottom: the result of individually and simultaneously applying rules I and II to the original tiling with different recursion levels.

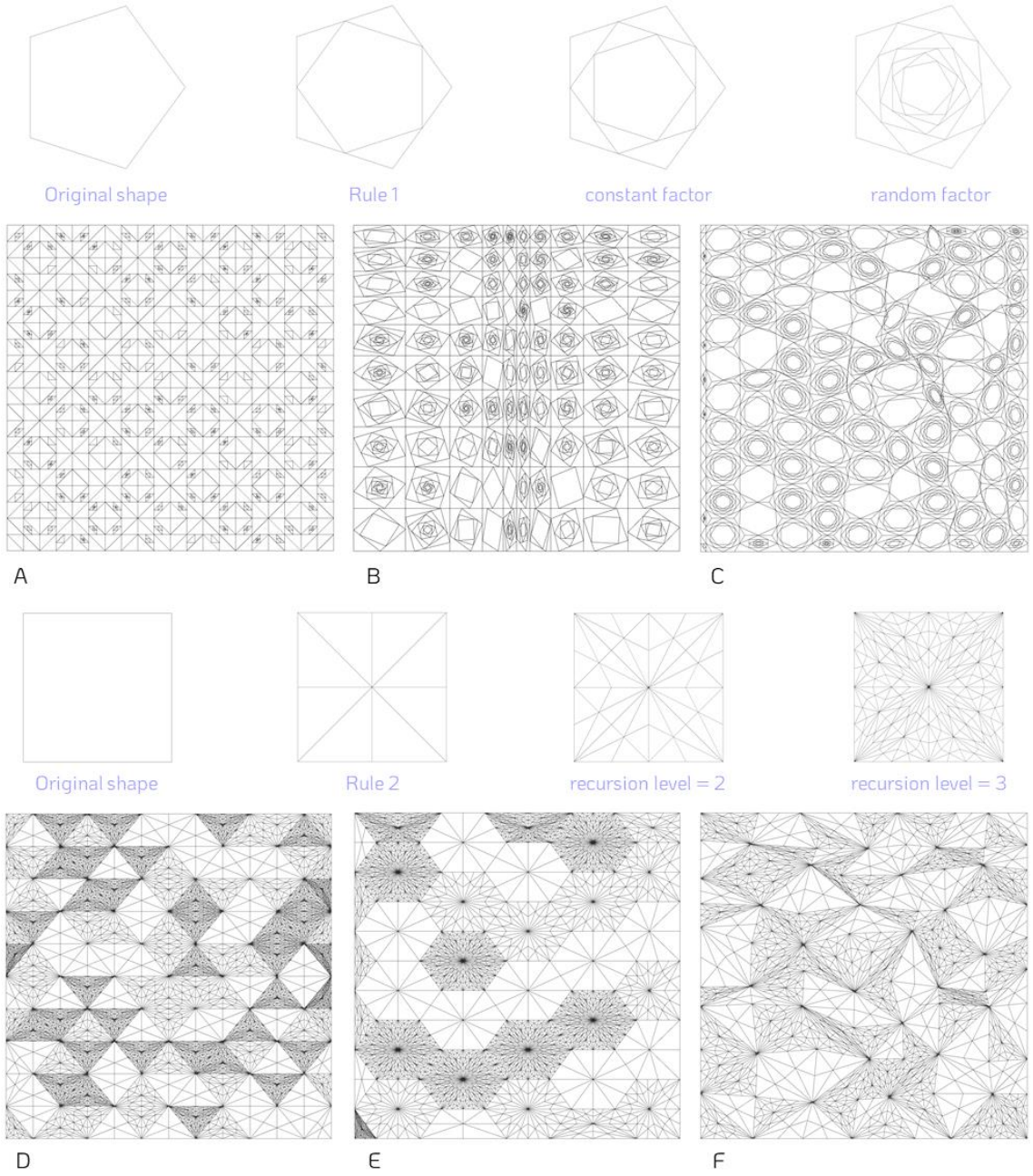


Figure 7.47. Two examples of subdivision rules: creating a polygon inside another polygon according to a factor (rule 1) and triangulating a polygon (rule 2). Examples A to C illustrate rule 1 applied to different grid configurations and with random levels of recursion and factors. Examples D to F apply rule 2 to different grid configurations with random levels of recursion.

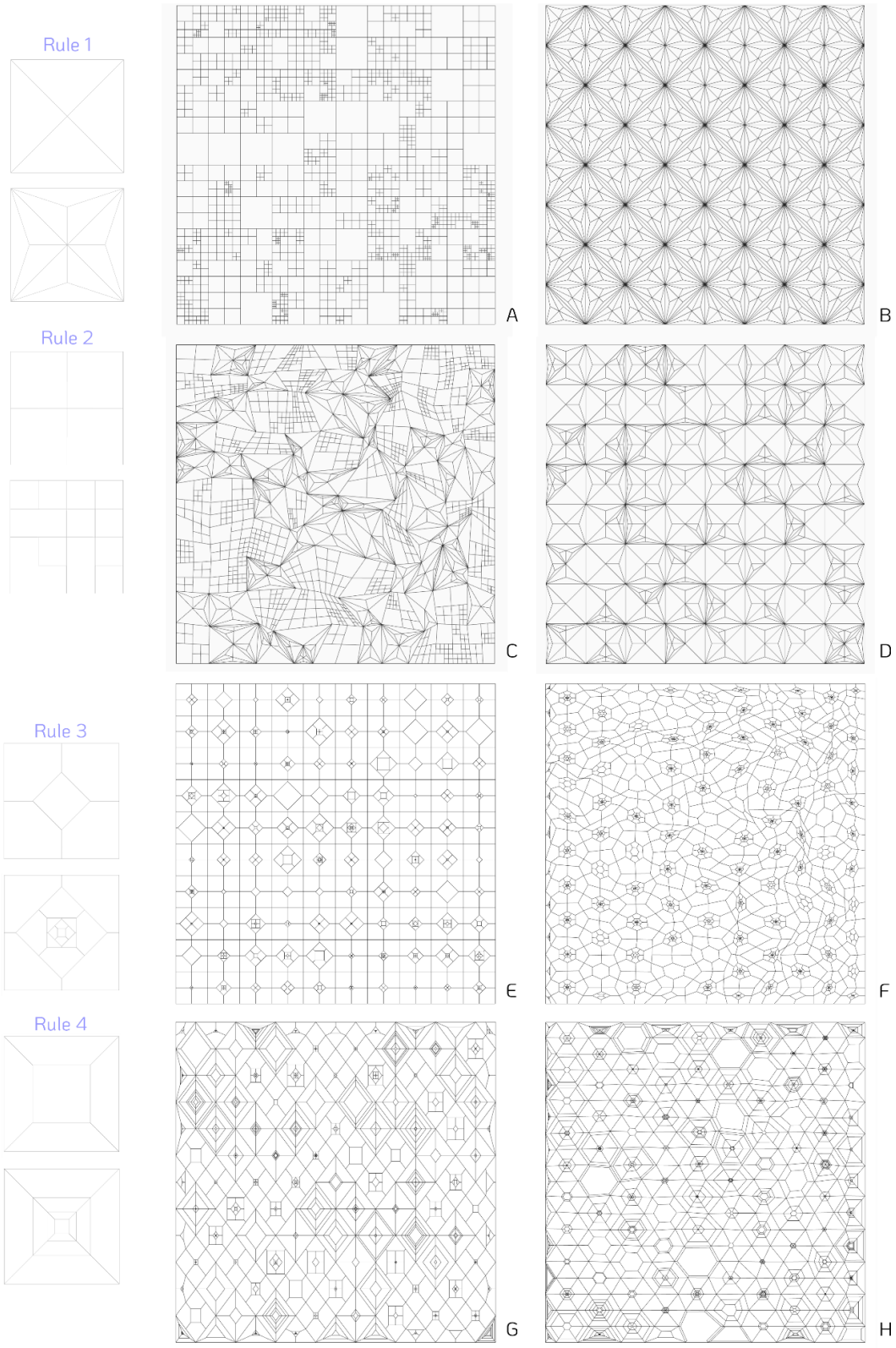


Figure 7.48. Four examples of subdivision rules: connecting the polygon's center to either its corners or edges (rules 1 and 2) or creating a polygon inside another polygon according to a factor with its corners connecting either the outer polygon's corners or edges (rules 3 and 4). Examples A to D apply either rule 1, 2, or both with random recursion levels to different grid configurations. Examples E to H apply either rule 3, 4, or both with random recursion levels and factors to different grid configurations.

Lastly, this section illustrates the application of the algorithm T_{edge}, which deforms the edges of a given shape through bending or folding. Mathematically, the framework represents both deformation movements with a vector function v, their amplitude and direction being the vector's length and sign, respectively (Figure 7.49-A).

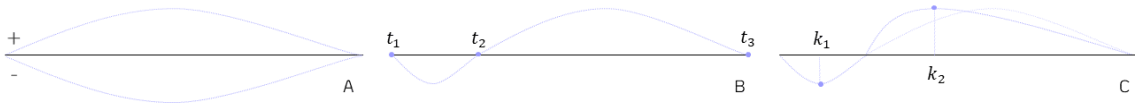


Figure 7.49. Conceptual representation of the algorithm T_{edge}: A. deformation movement sign; B. edge subdomain factors t; C. deformation maximum point factors k.

Using this algorithm, it is possible to apply multiple deformations to the same edge with different amplitudes and directions, as illustrated in Figure 7.49-B. To that end, we provide it with each edge's subdomains to which each deformation movement will be applied, which the framework mathematically represents as different t factors: in this example we divide the edge into two subdomains, from t_1 to t_2 and from t_2 to t_3, applying a different deformation to each one. Moreover, it is also possible to control the deformation maximum point position through different k factors: in Figure 7.49-C, for instance, while the negative deformation is symmetric, because $k_1 = 0.5$, the positive one is not, because $k_2 = 0.3$. Lastly, the framework also allows controlling the deformation smoothness by setting the parameter *smooth* as true when we want the edges bent, or as false when we want them folded.

In sum, this algorithm receives five matrices, one containing the surface points ($ptss$); another with the vector functions organized in sets $\{v_1, \dots, v_n\}$ ($M_{vectors}$); a third one with the t factors organized in sets $\{t_1, \dots, t_n\}$ ($M_{subdomain}$); another one with the k factors $\{k_1, \dots, k_n\}$ ($M_{curvature}$); and a last one with the latter three's order of application ($M_{pattern}$); plus a Boolean parameter ($smooth$) controlling the type of deformation created:

$$T_{edge}(ptss, M_{vectors}, M_{subdomain}, M_{curvature}, M_{pattern}, smooth)$$

In practice, for each surface position in $ptss$, $M_{pattern}$ dictates the set of vector functions $\{v_1, \dots, v_n\}$ to apply, as well as their respective t and k factors. To make this process clearer, Figure 7.50 illustrates a set of examples resulting from the application of T_{edge} to a regular grid.

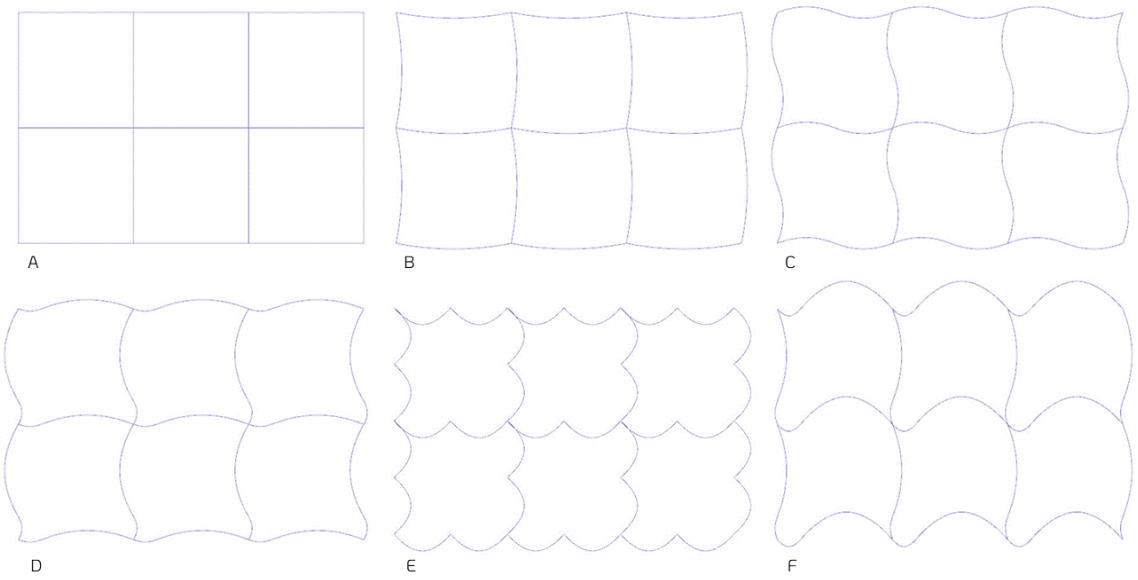


Figure 7.50. The algorithm T_{edge} applied to a squared grid (example A) with a single-direction deflection (example B) and a double deflection with different directions, t factors, and amplitudes (examples C to F).

CONTINUOUS TRANSFORMATIONS

This group contains geometric transformations for continuous patterns inspired on real facade designs (Figure 7.51). It includes algorithms such as *bending*, *folding*, *twisting*, *scaling*, and *weaving*, that, when combined with the algorithm $shape_{stripe}$, affect how each stripe-based element is generated by either twisting, undulating, or even scaling it.

Figure 7.51. Continuous transformations examples: One Ocean, Thematic Pavilion EXPO 2012 by soma (©soma); Liverpool Villahermosa Department Store by Iñaki Echeverria, UK (©Luis Gordoa); The Mantes-la-Jolie Water Sports Centre by Agence Search, France (©Emile Dubuisson).

Comparing to the previous *Transformation* algorithms, these algorithms present a different mathematical structure that requires a different combination strategy. The framework organizes them into two groups, *unidirectional* and *bidirectional transformations*, the first one containing algorithms to fold, bend, twist, and scale stripe-based elements, and the second to intertwine these elements in different ways.

Let us start with the first group of algorithms. When combined with $\boldsymbol{shape_{stripe}}$, which can receive one or more *Transformation* algorithms as optional arguments ($\boldsymbol{args}$ ...) (see section Shape), these algorithms can control its (1) straightness, by either bending or folding it according to different rules ($\boldsymbol{T_{bend}}$ and $\boldsymbol{T_{fold}}$); (2) rotation, by twisting it in different ways ($\boldsymbol{T_{twist}}$); and (3) width, by increasing or decreasing its section size along its length ($\boldsymbol{T_{width}}$).

As a first example, consider the application of a wave movement to a set of stripes. We therefore select the algorithm $\boldsymbol{T_{bend}}$, which receives (1) a row vector ($\boldsymbol{V_{bend}}$) with the undulating movement(s) to apply, which are represented as functions $\boldsymbol{f}$, and (2) a column vector ($\boldsymbol{V_{pattern}}$) dictating their order of application, which are represented as integers $\boldsymbol{i}$:

$$V_{bend} = [f_1, f_2, \dots, f_n], f_i \in \mathbb{R} \rightarrow \mathbb{R}^n \qquad V_{pattern} = \begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ i_n \end{bmatrix}, i_j \in \mathbb{N}$$

As in *ruled-based transformations*, integer 1 in $\boldsymbol{V_{pattern}}$ represents the algorithm of index 1 in $\boldsymbol{V_{bend}}$ ($\boldsymbol{f_1}$), integer 2 the algorithm of index 2 ($\boldsymbol{f_2}$), and so on. Therefore, if we combine the following vectors

$$V_{bend} = [f_1, f_2, f_3] \qquad V_{pattern} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 3 \end{bmatrix}$$

we obtain a stripe-based pattern made of alternated stripes resulting from three *Transformation* algorithms, i.e., $\{f_1, f_2, f_1, f_3, f_1, f_2, \dots\}$.

To simplify the use of this algorithm, the framework provides a set of predefined undulation strategies that, given (1) the surface domain ($\boldsymbol{ptss}$), (2) the intensity of the transformation effect ($\boldsymbol{k}$), and (3) a set of additional parameters depending on the movement to create ($\boldsymbol{args}$ ...), produce stripes with different undulations. As an example, consider Figure 7.52: while the stripes of pattern A result from fixed amplitude, frequency, and phase values, those of patterns B and C result from gradually smaller amplitude values along either the surface's length, in the first case, or height, in the second.

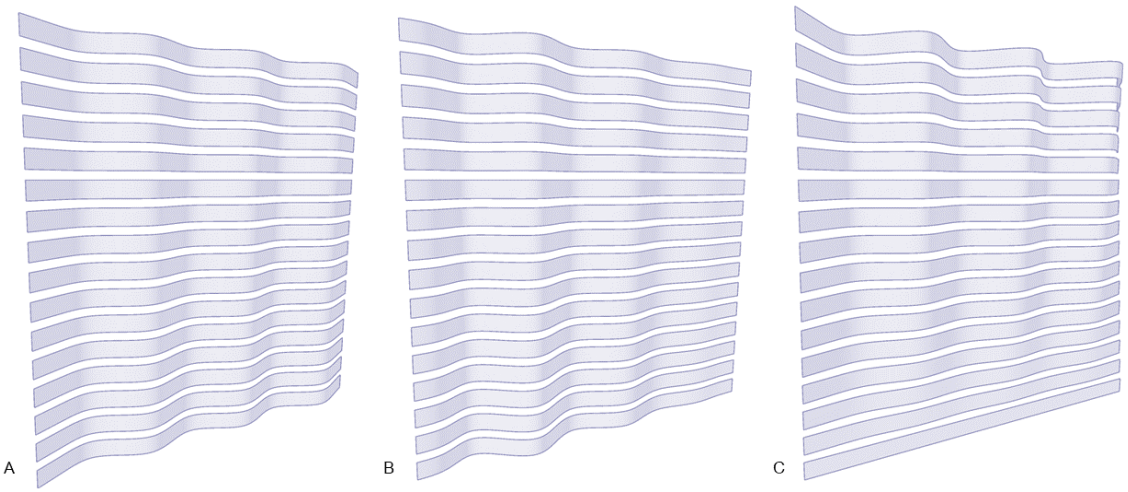


Figure 7.52. Stripe-based patterns resulting from (A) fixed amplitude, frequency, and phase values; (B) decreasing amplitude values along the surface length; and (C) increasing amplitudes along the surface height.

As another example, consider pattern A of Figure 7.53, which applies $\boldsymbol{T_{bend}}$ in an alternated way, keeping the odd stripes unchanged while bending the even ones. To achieve this result, we used a vector containing two algorithms, the no transformation ($\boldsymbol{T_{id}}$) and constant undulation ($\boldsymbol{f_1}$) ones, and another vector describing their alternate application:

$$V_{bend} = [T_{id}, f_1] \qquad V_{pattern} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

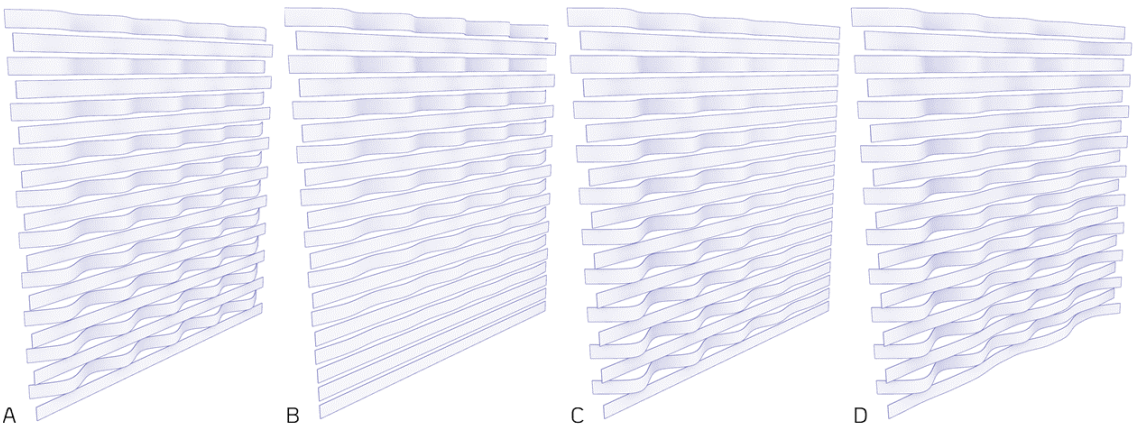


Figure 7.53. Application of the algorithm $\boldsymbol{T_{bend}}$ to a stripe-based pattern to (A) uniformly undulate the even stripes; (B) increase the even stripes' amplitude along the $\boldsymbol{v}$ direction; (C) decrease the even stripes' amplitude along the $\boldsymbol{u}$ direction; (D) undulate the odd stripes with an amplitude increasing in the $\boldsymbol{u}$ direction.

As integer 1 matches the first algorithm ($\boldsymbol{T_{id}}$) and integer 2 the second ($\boldsymbol{f_1}$), the odd stripes suffer no transformation, whereas the even ones are bent in a constant way. Imagine we replace the constant undulation amplitude ($\boldsymbol{f_1}$) with an increasing one in the $\boldsymbol{v}$ direction ($\boldsymbol{f_2}$). We obtain pattern B. If we

instead replace it with a decreasing amplitude in the $\boldsymbol{u}$ direction ($\boldsymbol{f_3}$), we obtain pattern C. Lastly, if we replace the first algorithm ($\boldsymbol{T_{id}}$) with an increasing amplitude in the $\boldsymbol{u}$ direction, we produce pattern D.

The same logic applies to the remaining algorithms $\boldsymbol{T_{fold}}$, $\boldsymbol{T_{twist}}$, and $\boldsymbol{T_{width}}$, which receive a row vector with the folding ($\boldsymbol{V_{fold}}$), twisting ($\boldsymbol{V_{twist}}$), and scaling ($\boldsymbol{V_{scale}}$) effects to apply, respectively, and a column vector with their order of application ($\boldsymbol{V_{pattern}}$). Figure 7.54 illustrates their application on the previous stripe-based pattern.

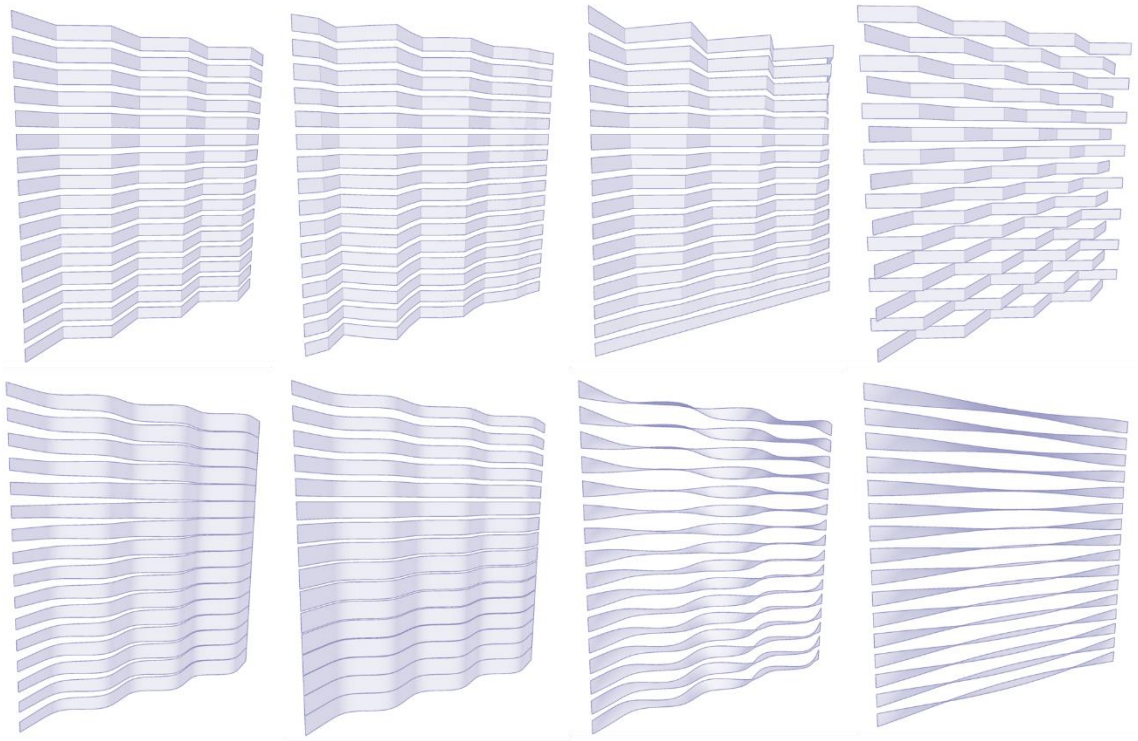


Figure 7.54. Stripe-based patterns resulting from the application of the algorithms $\boldsymbol{T_{fold}}$ (top), $\boldsymbol{T_{width}}$ (bottom left ones) or $\boldsymbol{T_{twist}}$ (bottom right ones).

Regarding the *bidirectional transformations* group, it contains algorithms to create different weaving patterns as illustrated in Figure 7.55. Also known as woven structures, these patterns are no more than stripe-shaped elements strategically bent to not intersect each other [370]. This means that we can benefit from the previous algorithm $\boldsymbol{T_{bend}}$ to interlace a set of perpendicular stripe-based elements following different weaving strategies.

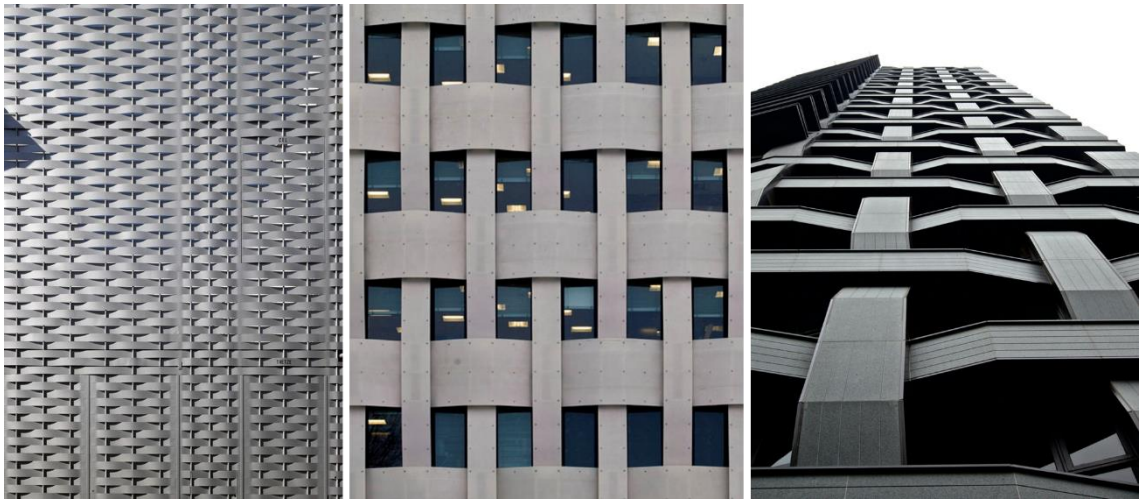


Figure 7.55. Weaving patterns examples: Casa AAG by Manuel Cerdá Pérez, Spain (©Joan Roig); Eemsmond Building by Team 4 Architecten, Netherlands (©Holland Composites BV); Yong He Yuan residential buildings by NEDELEC Architecture (©Florent Nedelec).

In a first approach, we could simply apply the algorithm $\boldsymbol{T_{bend}}$ to a set of $\boldsymbol{n}$ vertical and $\boldsymbol{m}$ horizontal stripes in a coordinated way. However, this would require us to solve the intersection problems between stripes for each design variation tested by, for instance, increasing the stripes' frequency when changing their number; varying both their phase and frequency values when changing the weaving strategy; adjusting their amplitude, phase, and frequency values when changing the surface dimension and/or curvature; and so on. As in architecture several design iterations are usually performed, this apparently simple task would quickly become a time-consuming and tiresome process. To automate it, the framework provides the algorithm $\boldsymbol{T_{weave}}$, which receives the surface points ($\boldsymbol{ptss}$), the number of vertical and horizontal stripes ($\boldsymbol{u_{stripes}}$, $\boldsymbol{v_{stripes}}$), the weaving amplitude ($\boldsymbol{amp}$) and type ($\boldsymbol{M_{weave}}$), the stripes widths ($\boldsymbol{M_{widths}}$) and thicknesses ($\boldsymbol{M_{thicknesses}}$), and a set of optional arguments ($\boldsymbol{args}$ ...), including a Boolean parameter ($\boldsymbol{straight}$) that makes the stripes straight when $\boldsymbol{true}$ and undulated when $\boldsymbol{false}$ and a set of additional transformations controlling the stripes' rotation and/or width:

$$T_{weave}(ptss, u_{stripes}, v_{stripes}, amp, M_{weave}, M_{widths}, M_{thicknesses}, args \dots)$$

Regarding their mathematical representation, the framework adopts a strategy inspired by that of Grünbaum and Shephard [371] to represent fabrics, which uses a two colors 2D squared mesh to represent different weaving types, where black means the stripe along the vertical direction passes over the stripe in the horizontal direction and white the opposite (Figure 7.56).

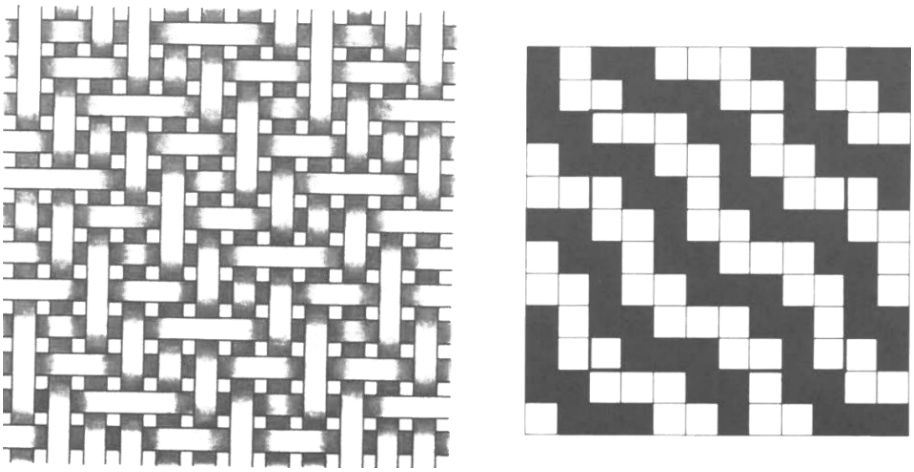


Figure 7.56. On the left, a sketch of a fabric (left) and, on the right, its conceptual representation, the outlined squared area corresponding to its simplest pattern unit [372].

To adapt this approach to the mathematical nature of the proposal, the 2D mesh graphical representation is converted into a numerical one where black corresponds to integer 1 and white to integer -1. This means the weaving type parameter ($\boldsymbol{M_{weave}}$) is a MI where all elements are either 1 or -1. Based on this information, the algorithm $\boldsymbol{T_{weave}}$ automatically applies different amplitude, frequency, and phase values to each stripe, originating different weaving outcomes (Figure 7.57).

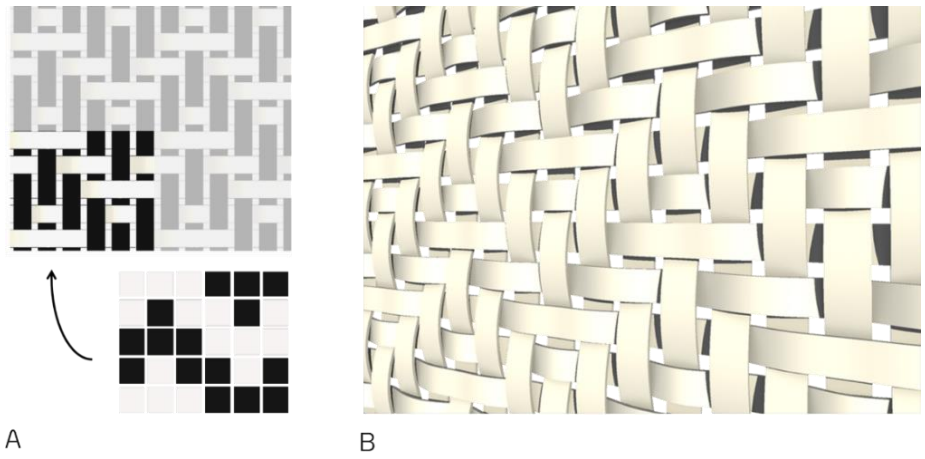


Figure 7.57. Weaving type graphical representation: A. the weaving type to apply (top) and its corresponding two-color matrix (bottom); B. the resulting weaving pattern.

To facilitate the development of weaving facade patterns, the framework provides several predefined weaving types that can be directly combined with the algorithm $\boldsymbol{T_{weave}}$. Figures 7.58-59 illustrate some examples of their application.

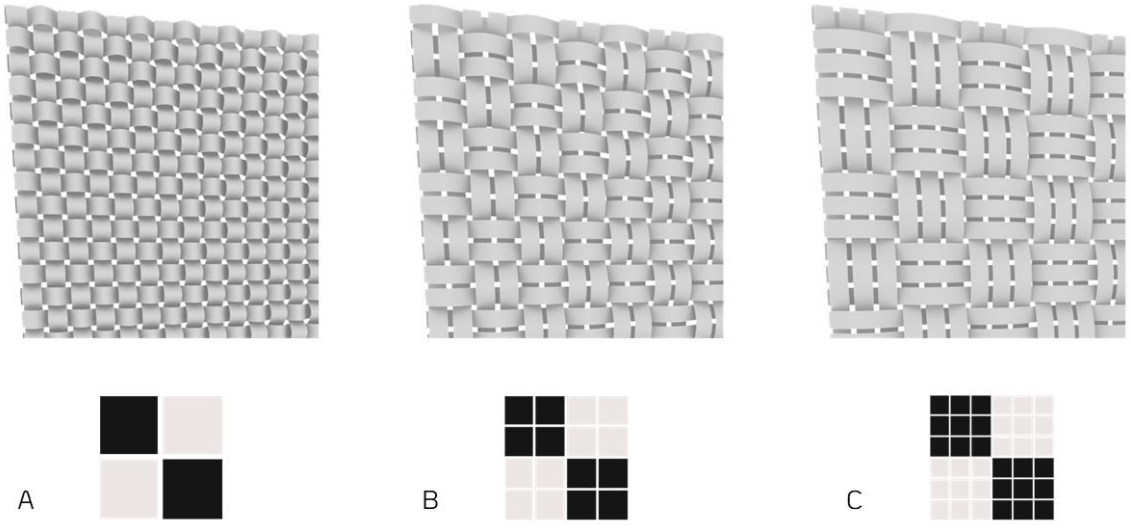


Figure 7.58. Three weaving types – chess (A), double-chess (B), and triple-chess (C) – with the corresponding representation matrices.

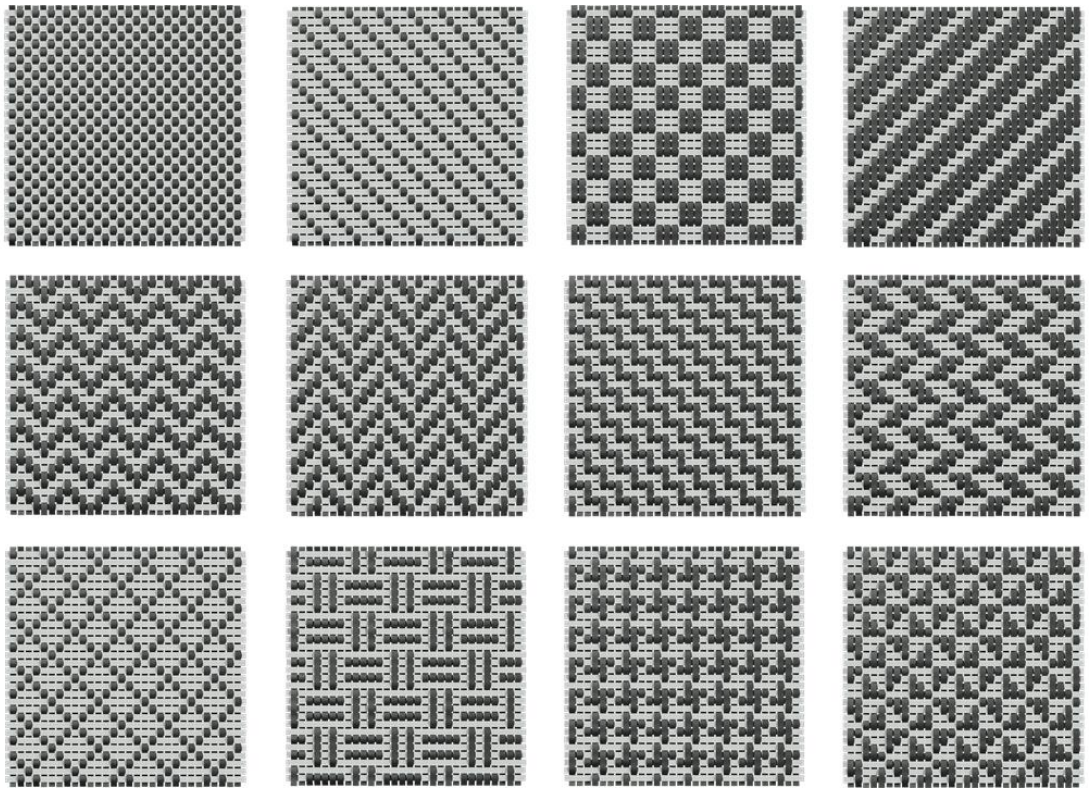


Figure 7.59. Twelve examples of predefined weaving types.

Lastly, this section illustrates the flexibility of $\boldsymbol{T_{weave}}$ in automatically adapting the selected weaving type to different numbers of stripes (Figure 7.60), surface shapes (Figure 7.61), and geometric transformations (Figure 7.62). Note that, in all examples, no additional effort was needed, for instance, to (1) adjust the

stripes' bending movement and curvature, (2) guarantee their equidistance along the surface, and (3) change their rotation angle or section size according to the rules set.

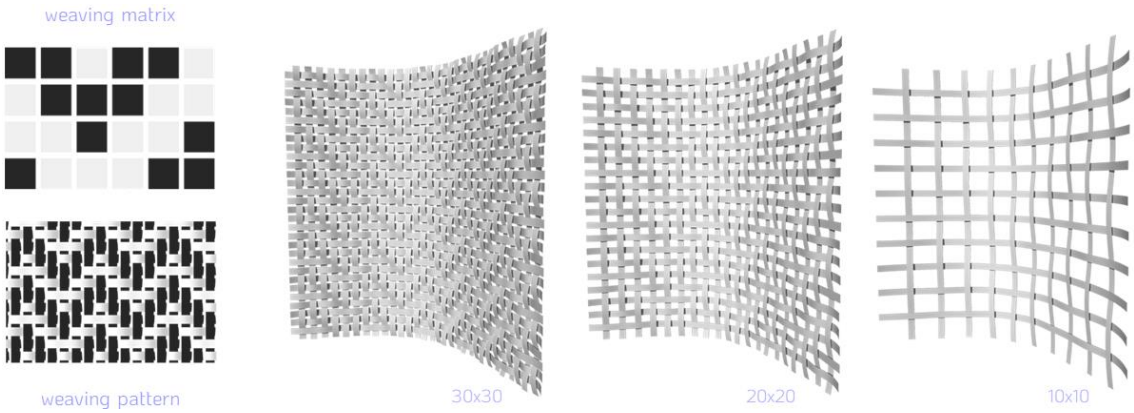


Figure 7.60. The weaving type on the left with different numbers of stripes.

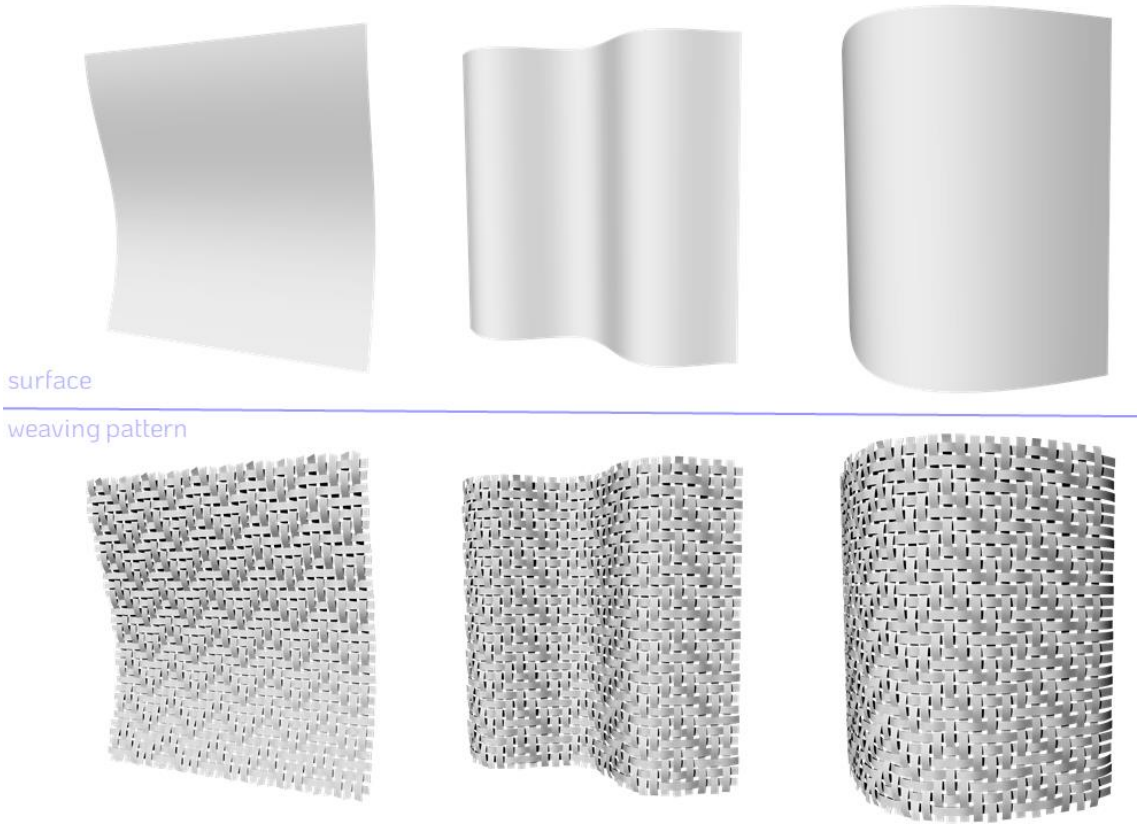


Figure 7.61. The same weaving type applied to different surfaces.

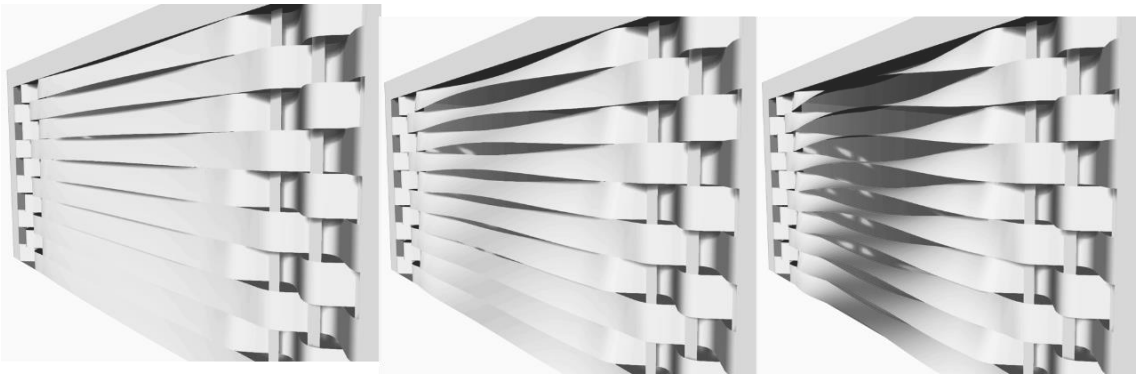


Figure 7.62. The same weaving pattern with differently rotated stripes.

7.3.4. OPTIMIZATION

The previous sections showed how the available strategies allow to generate different facade design solutions based on different geometry-related parameters. This section demonstrates how some of these parameters can be manipulated according to different performance requirements. To that end, the *Optimization* category contains algorithms to support the analysis and optimization of the facade design solutions produced with the previous categories regarding different performance criteria. When combined with the previous algorithms, these algorithms provide insight on the quality of the evaluated solutions with respect to the selected metrics, while allowing their incremental refinement through different optimization strategies.

Given the complexity and technicality of performance evaluation and optimization strategies, this category provides algorithms that, instead of dealing directly with these processes, automate the connection to the already existing specialized tools and libraries for that purpose, such as Robot, Frame3DD, and Radiance, among others (see chapter 5). This approach avoids the wasteful replication of the expertise of these tools.

By encapsulating relevant procedures of different analysis tools into simple and easy-to-use algorithms, this category simplifies the use of different performance evaluation strategies. Additionally, it also enables the integration of the framework's predefined algorithms with those of other frameworks to benefit from their analysis capabilities. This integration is important to ensure that the entire design workflow, from design conception to performance evaluation, can be algorithmically described, allowing architects to automate the execution of different analysis cycles, as well as the integration of the analysis results in the design workflow.

In general, the algorithms available in this category are HOFs that receive the function composition describing the facade design to analyze (***shapes***) and a set of additional arguments (***args*** ...) that depend on both the type of analysis to perform and the existing fitness requirements, as

well as the specificities of the analysis tool to use. Based on the received information, these algorithms then establish the connection with the selected tool, automating the transfer of design data according to the type of analysis to perform and the execution of the different analysis-related tasks involved in it. The results can be graphically visualized in the analysis tool being used or they can be returned for further processing and/or visualization in a different external tool by coupling the framework to a modeling tool or visualization engine. After reflecting on the results and perceiving the quality of the solution developed, the architect might change the design and re-evaluate the solution to check if improvements were made.

As an example, imagine we want to adjust two design variables of pattern A of Figure 7.63, namely the size of the diagonals ($\boldsymbol{d_u}$ and $\boldsymbol{d_v}$), towards the shading goal illustrated by example B. We select the most suitable algorithm from this category, providing it with:

1. The algorithmic description of the design solution ($\boldsymbol{shapes}$), which, in this case, is a composition of the algorithms $\boldsymbol{straight}$, $\boldsymbol{grid_{rhombus}}$, and $\boldsymbol{shape_{rhombus}}$ of the categories *Geometry*, *Distribution*, and *Pattern*, respectively.
2. Information about the type of analysis to perform, performance metrics to evaluate, and the analysis tool to use.

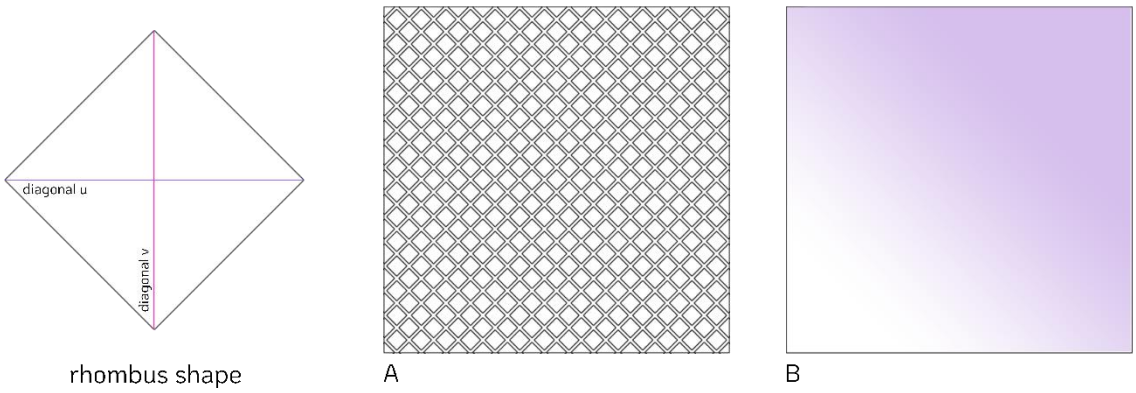


Figure 7.63. On the left, the design variables to adjust; on the right, conceptual representation of (A) the design to evaluate and (B) the goal considered - the more intense the color, the greater the shading effect should be.

Based on the results, we then adjust the size-related parameters of pattern A, and then re-evaluate its shading performance to check for improvements. Figure 7.64 illustrates the result of this process after a set of iterations where only one (examples A and B) or both (example C) diagonals were changed.

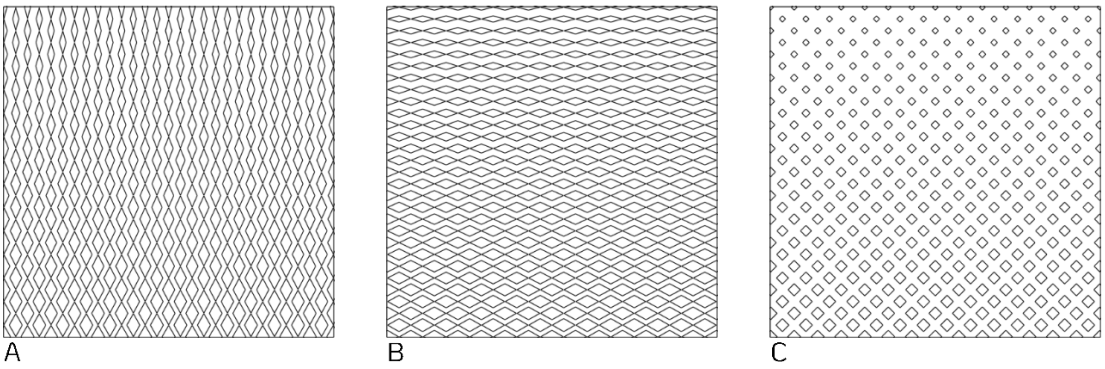


Figure 7.64. The result of an iterative process of analyzing and adjusting the design, in this case the rhombuses' size-related parameters (A) diagonal d_u; (B) diagonal d_v; or (C) both diagonals.

Through the use of algorithms that generate samples of possible solutions (such as those of the *Geometry*, *Distribution*, and *Pattern* categories) and algorithms that simplify their evaluation according to different criteria (such as those of this category), it becomes possible to incrementally improve the design by repeating the *design-evaluation-redesign* cycle in an iterative way. This process can be executed either manually, where the architect, on each iteration, evaluates the results and changes the design accordingly, or automatically, by applying an optimization routine. To support the latter scenario, the framework provides the means to couple the available geometry- and analysis-related strategies with optimization algorithms. Having this ability, it becomes easier to benefit from the large variety of optimization libraries currently available. Depending on the design parameters allowed to change during this process, we obtain design solutions with either elements of different sizes, when changing size-related parameters; positions, when altering translation-related values; or even shapes, when changing transformation- or shape-related functions.

To better illustrate this ability, consider an example where the framework is coupled with an optimization routine to improve both the structural performance and cost of a pyramidal truss-like facade (Figure 7.65).

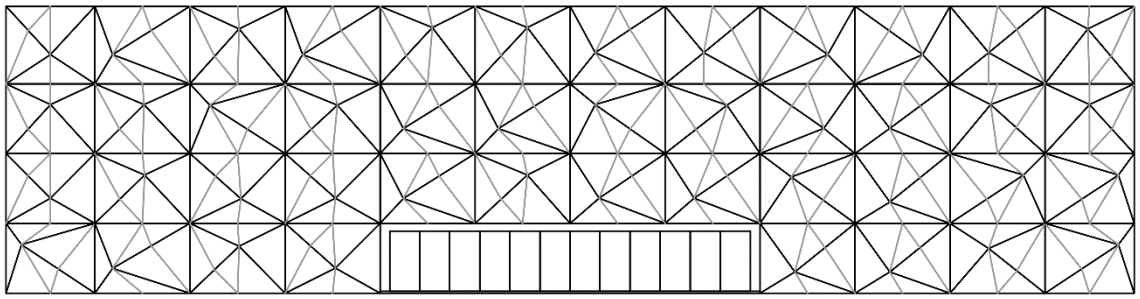


Figure 7.65. Front view of a truss-like facade: the black lines represent the pyramidal truss elements and the grey lines the elements subdividing their faces.

To create the pyramidal truss structure, we select algorithms from the categories:

- *Geometry*, to produce a straight surface ($\boldsymbol{straight}$).
- *Distribution*, to create a rectangular grid configuration ($\boldsymbol{grid}_{\boldsymbol{rectangles}}$).
- Pattern, to create the pyramidal truss elements ($\boldsymbol{shape}_{\boldsymbol{truss}}$ and $\boldsymbol{shape}_{\boldsymbol{pyramidVertices}}$).

By combining the first two algorithms, we obtain a set of surface positions arranged in a rectangular grid and by mapping the last two algorithms onto it, we obtain a truss with a rectangular configuration and pyramidal elements (Figure 7.66-A).

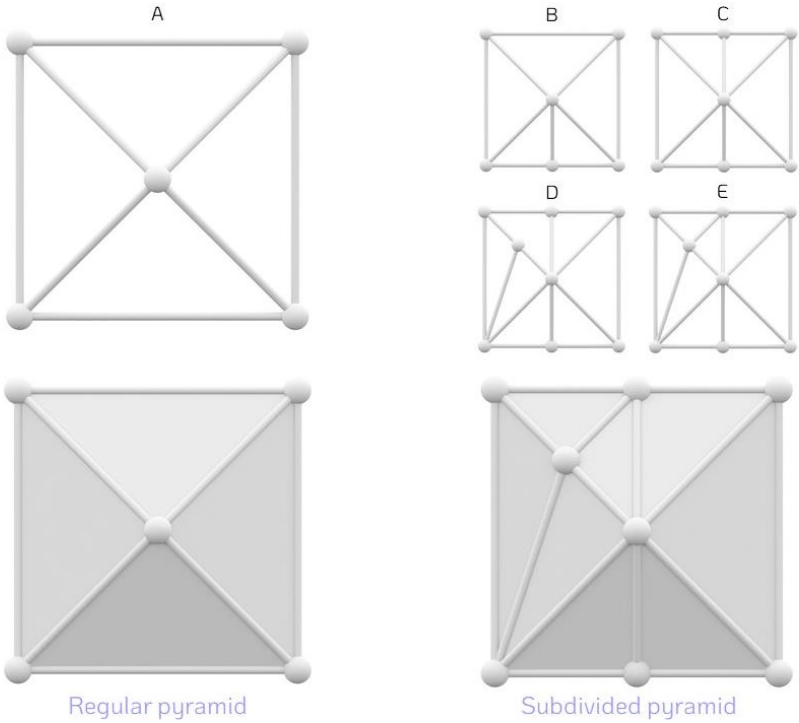


Figure 7.66. Pyramidal truss elements before (A) and after the application of a subdivision rule (B-E).

To create the subdivision effect illustrated in Figure 7.66 (on the right), we combine the transformation algorithm $\boldsymbol{T}_{\boldsymbol{subdivide}}$ with those generating the pyramidal elements, and to obtain the random effect illustrated in Figure 7.67, we add two more transformation algorithms to the previous composition, namely $\boldsymbol{T}_{\boldsymbol{translate}}$ and $\boldsymbol{T}_{\boldsymbol{random}}$, making the pyramids' apices vary in a random way.

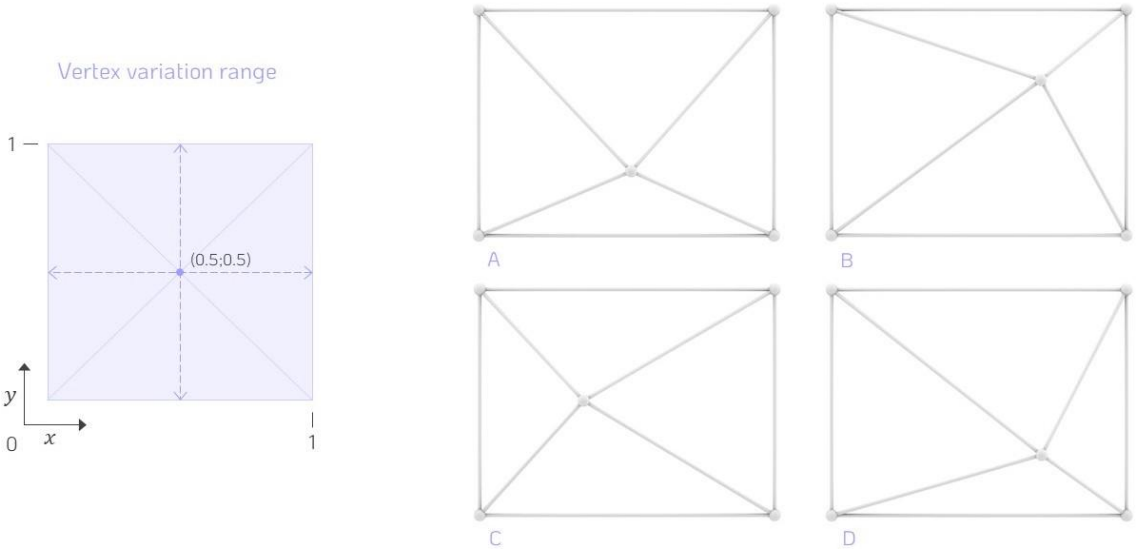


Figure 7.67. Left: ranges between which the pyramids' apices can vary in both x and y directions (e.g., when $kx = ky = 0.5$ the apex is centered); Right: four pyramids resulting from different factors (A: $kx = 0.5; ky = 0.3$; B: $kx = ky = 0.65$; C: $kx = 0.35; ky = 0.5$; D: $kx = 0.7; ky = 0.3$).

Then, to analyze the solution in terms of structural performance, we select the algorithm $eval_{structure}$ and the structural analysis tool (in this case Robot), obtaining the results (1) numerically, through tables containing structure-related information such as the structure's weight and displacement, and (2) geometrically, by visualizing the 3D models of both the original and deflected structures in the modeling tool in use. Figure 7.68 illustrates this process with eight design variations under the same load conditions: truss A has all nodes fixed and pyramids with constant heights and panel subdivisions; truss B is equal to A but without panel subdivisions; trusses C and D are similar to A but their pyramids' heights randomly range from 1 to 4 times the original value, in the first case, or have a base with twice the original size, in the second; truss E is equal to A but only with outer nodes fixed; finally, trusses F, G, and H are equal to E but have smaller section bars, increased loads, and a different truss material, respectively.

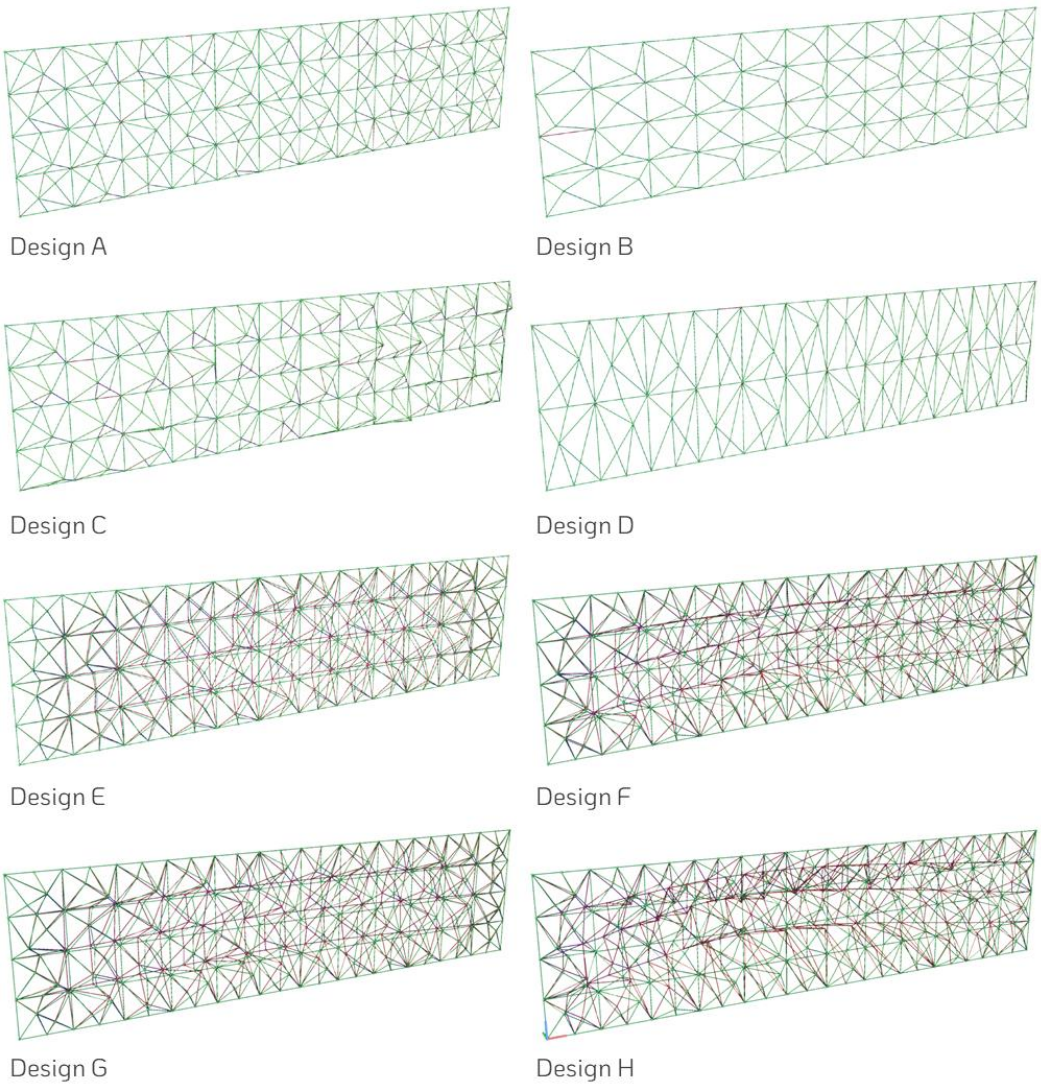


Figure 7.68. Original (green) and deflected structures (red) of eight design variations.

At this stage, the analysis of the results allows us to perceive that:

- Panel subdivisions sightly increase the truss deformation.
- Pyramids with a greater height increase the truss weight.
- Pyramids with larger bases reduce the truss weight but slightly increase its deformation.
- Less fixed nodes and/or thinner truss bars result in higher buckling.
- Different loads or truss bar materials directly affect the structure displacement.

To further improve the solution's structural performance and cost, we can automate iterative structural analyses by coupling the previous algorithms with an optimization routine. For instance, we can set as optimization variables the pyramids' height size, apices' random variation, and bar material; as objective functions the structure's maximum displacement and cost; and as optimization algorithm the NSGA-II.

Figure 7.69 presents the results of 300 structural evaluations, where M0, M1, M2, and M3 correspond to the different materials used and the lilac curve to the set of best solutions (i.e., the Pareto front). Based on its analysis, we conclude that to significantly improve the truss structural performance, we need to use more expensive materials, as is the case of solution I1. Nevertheless, we can also consider other acceptable solutions that, despite having a slightly lower structural performance, are more cost effective, as is the case of solutions I2 and I3.

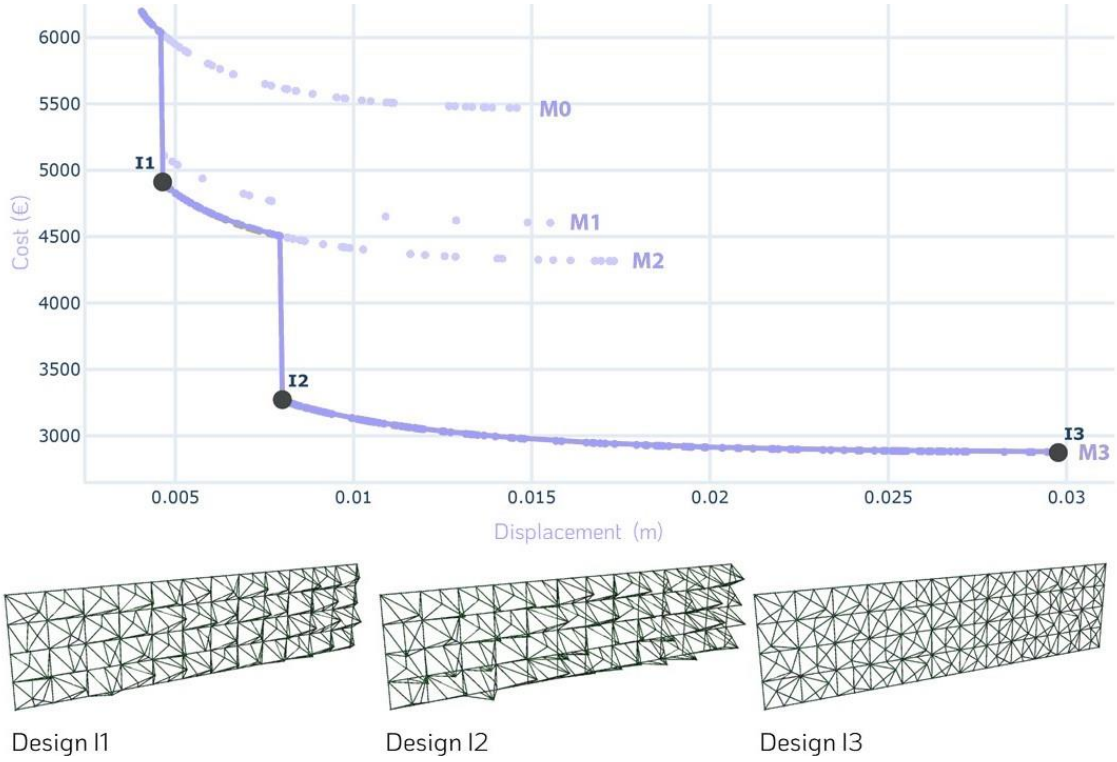


Figure 7.69. Optimization of the truss' structural performance and cost using four materials (M0,M1,M2,M3): on top, the resulting Pareto front (lilac curve) and, at the bottom, three optimal designs (I1, I2, I3).

Despite the simplicity of the previous examples, they demonstrate the ability of the framework to simplify the performance evaluation of a given design solution regarding different criteria. They also demonstrate its ability to incorporate additional design tasks and to be coupled with external frameworks and libraries specialized in different analysis and optimization routines and benefit from their different strategies.

7.3.5. RATIONALIZATION

Rationalization is the geometric simplification of a design for manufacturing purposes. It can involve the reduction of the number of elements composing a solution, the simplification of the manufacturing method adopted, or the incorporation of additional design constraints, among others. To facilitate the integration of these strategies into the facade design process, the Rationalization category provides several algorithms that, when combined with those of the previous categories, allow:

1. Counting the number of different elements (or typologies) of a given solution.
2. Identifying their spatial locations and positions.
3. Controlling the maximum number of typologies allowed.
4. Simplifying the resulting solution for manufacturing purposes.

These are important aspects to consider not only to increase the designs' feasibility, reducing both its manufacturing costs and the time and resources needed for their production, but also to facilitate the ensuing manufacturing stage and on-site assembly. Given the framework's flexibility, it is possible to combine these algorithms with design requirements other than construction viability, such as creative intents and environmental performance, and search for solutions that are feasible and successful in terms of aesthetic and performance as well.

To identify and count different facade elements (typologies), the framework provides the algorithm $\boldsymbol{tallying}$. Mathematically, this algorithm is a HOF that receives a function, or a composition of functions, describing the facade design solution:

$$tallying(shapes)$$

It then returns the list of typologies composing the design solution with the corresponding quantities and spatial locations in two formats: numerically, by presenting tables with the stored information, and graphically, by displaying the results in the design tool in use by means of a color scheme (Figure 7.70). While the former allows obtaining accurate and detailed information about the existing typologies, the latter facilitates their interpretation by increasing the perception of their quantities and precise locations.

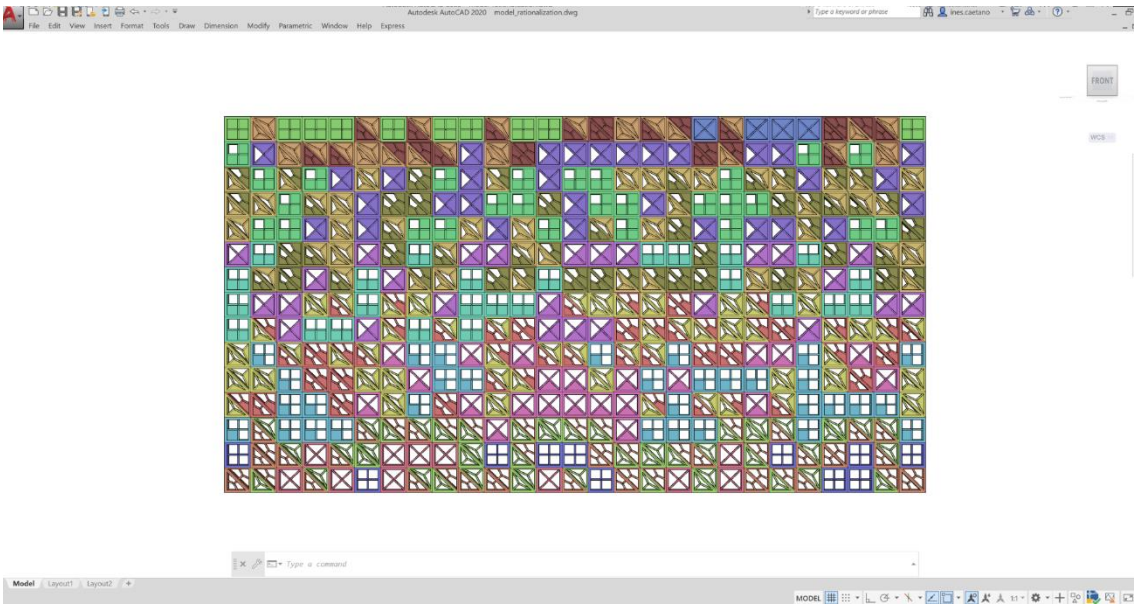


Figure 7.70. Graphical visualization of different typologies of bricks using a color scheme.

To control the maximum number of typologies allowed, this category provides the algorithm $\boldsymbol{rationalize}$. When executed, this algorithm iteratively reduces the geometric diversity of a given design solution ($\boldsymbol{shapes}$) until reaching the range of typologies allowed ($\boldsymbol{n_{typologies}}$):

$$rationalize(shapes, n_{typologies})$$

Mathematically, this algorithm is also a HOF that, given the function or composition of functions describing the design to rationalize, reduces the number of accepted values by one or more of the latter's variables, converting a continuous interval into a set of $\boldsymbol{n}$ discrete values matching the number of typologies allowed ($\boldsymbol{n_{typologies}}$). In practice, it collects the values that differ from element to element, which can be, among others, size-related parameters, transformation factors, or even material properties. Based on this information, it then produces a new set of discrete values varying within the range of original values (i.e., the maximum and minimum values found in the previous stage) but containing at most the maximum number of typologies set ($\boldsymbol{n_{typologies}}$). Finally, it replaces the variable(s)' original inputs with their closest match in the discretized interval, thus guaranteeing the final solution has no more than $\boldsymbol{n_{typologies}}$. Figure 7.71 illustrates this process.

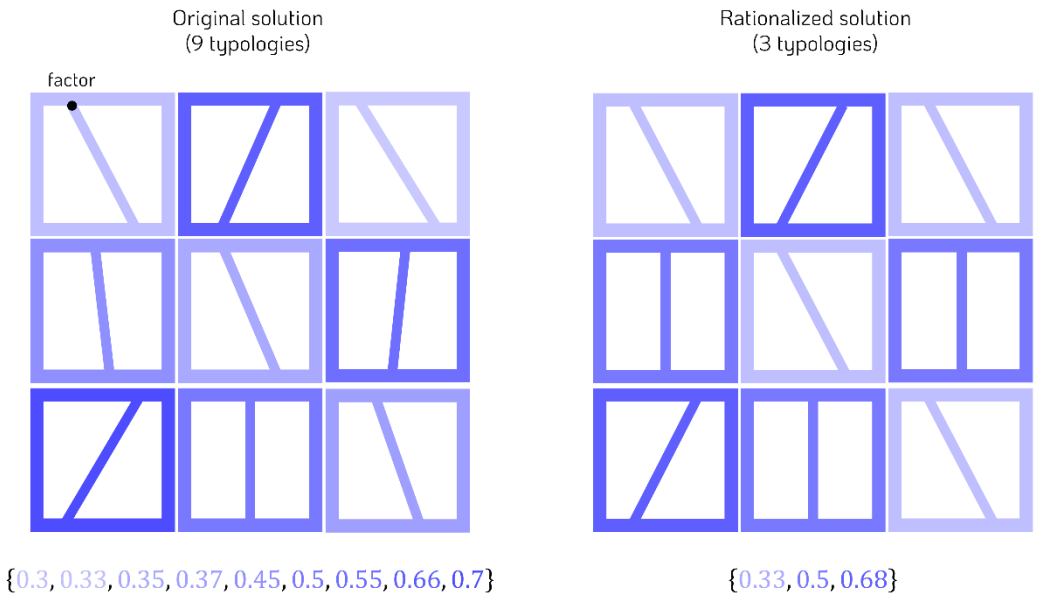


Figure 7.71. Conceptual representation of the rationalization process: on the left, the original solution composed of 9 different elements resulting from the set of factors below; on the right, the solution after the rationalization process with $n_{typologies} = 3$, the initial set of nine factors being reduced to three.

The simplicity and speed of this process allows the architect to perform several iterations with different values and assess their impact on the solution's visual expression and performance quality and thus control the number of typologies produced more conscientiously, increasing the solution feasibility without compromising the design intent.

Lastly, to divide a facade design into smaller segments by considering similarities between them, the framework provides the algorithm $\boldsymbol{discretize}$. Mathematically, this algorithm is also a HOF that receives the function composition describing the facade design to discretize ($\boldsymbol{shapes}$) plus a level of refinement ($l_{refinement}$), which controls the degree of discretization to apply. In practice, the higher this value, the smoother the geometric transition between segments, and the greater the number of different segments.

$$discretize(shapes, l_{refinement})$$

To better understand the previous algorithms applicability, consider the pattern illustrated in Figure 7.72-E.

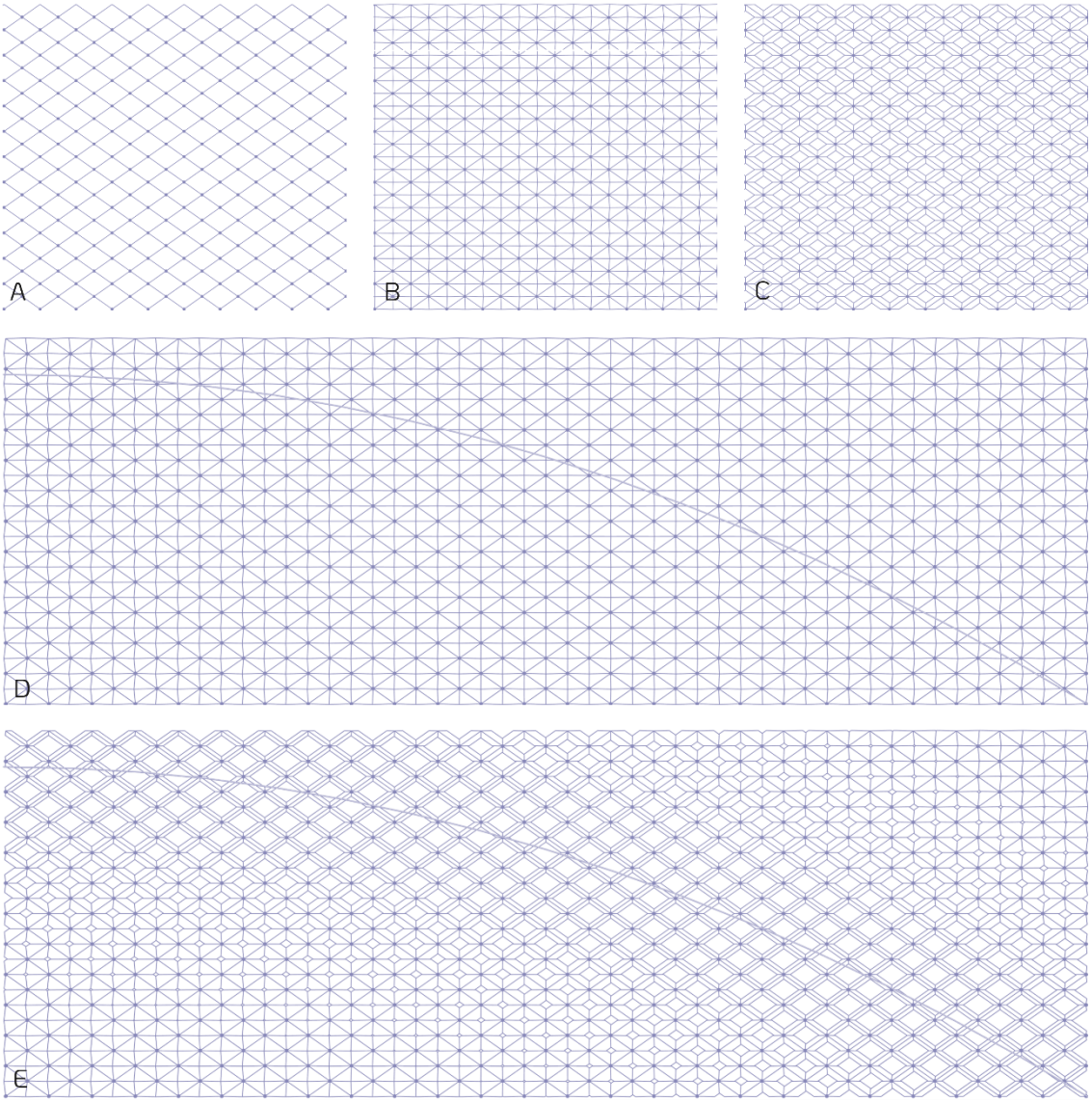


Figure 7.72. Pattern's geometric implementation: creating a rhombus grid (A); mapping the pyramidal elements (B); creating the pyramids' openings (C) while controlling their aperture factor according to their distance to a curve (D-E).

To produce the diamond-shaped elements, we select algorithms from the categories (1) *Geometry*, to produce a planar surface ($\boldsymbol{straight}$); (2) *Distribution*, to organize the latter's positions into a rhombus grid ($\boldsymbol{grid}_{rhombus}$); and (3) *Shape*, to create the pyramidal elements ($\boldsymbol{shape}_{pyramid}$). To make the apertures vary in the desired way, we select the algorithms (4) $\boldsymbol{T}_{scale}$, to control their size, (5) $\boldsymbol{attractor}_{curve}$, to calculate the aperture factor according to their distance to a curve, and (6) $\boldsymbol{curve}_{parabola}$, to describe the curve.

Then, we select the algorithm $\boldsymbol{tallying}$ to count the total number of elements and existing typologies composing the final solution, obtaining a total of 6144 triangular panels from which 2424 are different. To make their production viable in terms of cost and resources, we select the algorithm

$\boldsymbol{rationalize}$, gradually reducing the number of allowed typologies, while increasing their frequency of use. The results of this process are illustrated in Table 7.1, where the first column (ID) identifies the existing typologies, and the remaining ones present their frequency of use for different $n_{typologies}$.

Piece typology ID	∞	100	60	20	10	5
1	432	478	478	478	728	1208
2	1560	1578	1602	1684	1806	2558
3	2	70	116	254	502	1054
4	2	36	108	250	488	1324
5	2	44	108	246	546	
6	2	52	120	256	578	
7	2	52	122	254	640	
8	4	46	118	260	856	
9	4	60	140	268		
10	4	42	134	274		
11	2	54	154	300		
...	...	...	...	...		
Σ	2424	69	42	15	8	4

Number of piece typologies used

Table 7.1. Typologies' ID number (left column) and frequency of use (remaining columns) for different $n_{typologies}$: ∞, 100, 60, 20, 10, and 5 typologies allowed.

As, during this process, it is possible to visualize the results graphically (see Figure 7.73), it becomes easier to compare the solutions in terms of cost and resources: the less colors they have, the less panel typologies exist and the more affordable their manufacturing is. Similarly, it is also possible to visualize the typologies' frequency of use through a color code (Figure 7.74), where red tones represent the least used panels and green tones the most used ones, and thus perceive almost immediately which are the most and least cost-effective typologies, i.e., whose frequency of use either justifies or not its manufacturing.

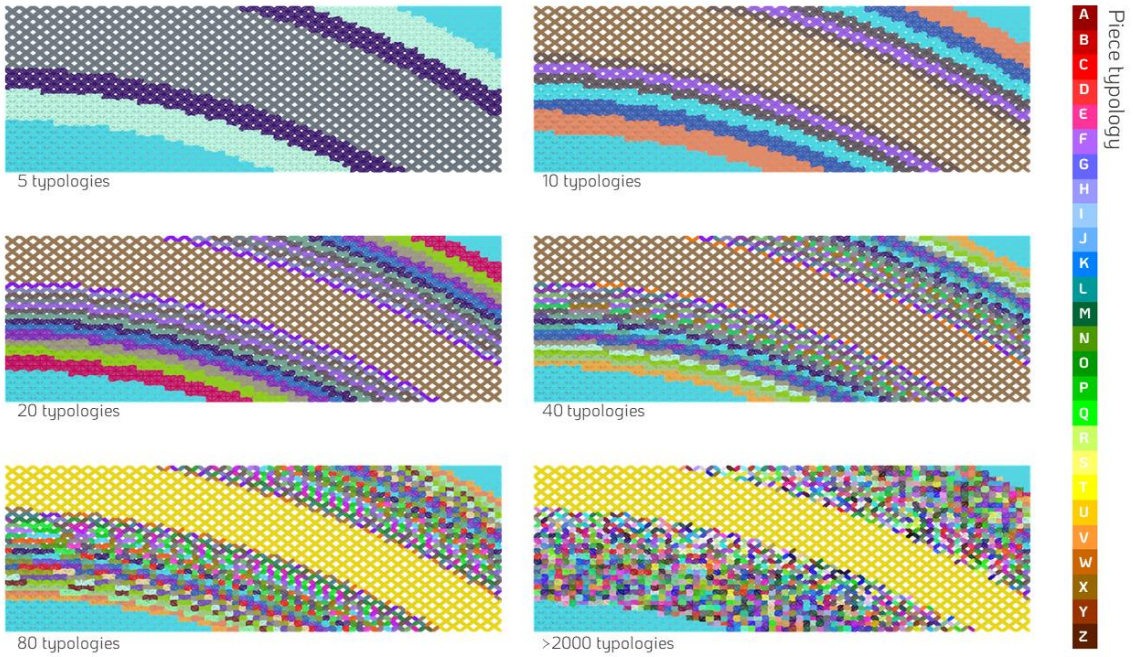


Figure 7.73. Representation of the existing typologies for different rationalization levels using a color scheme.

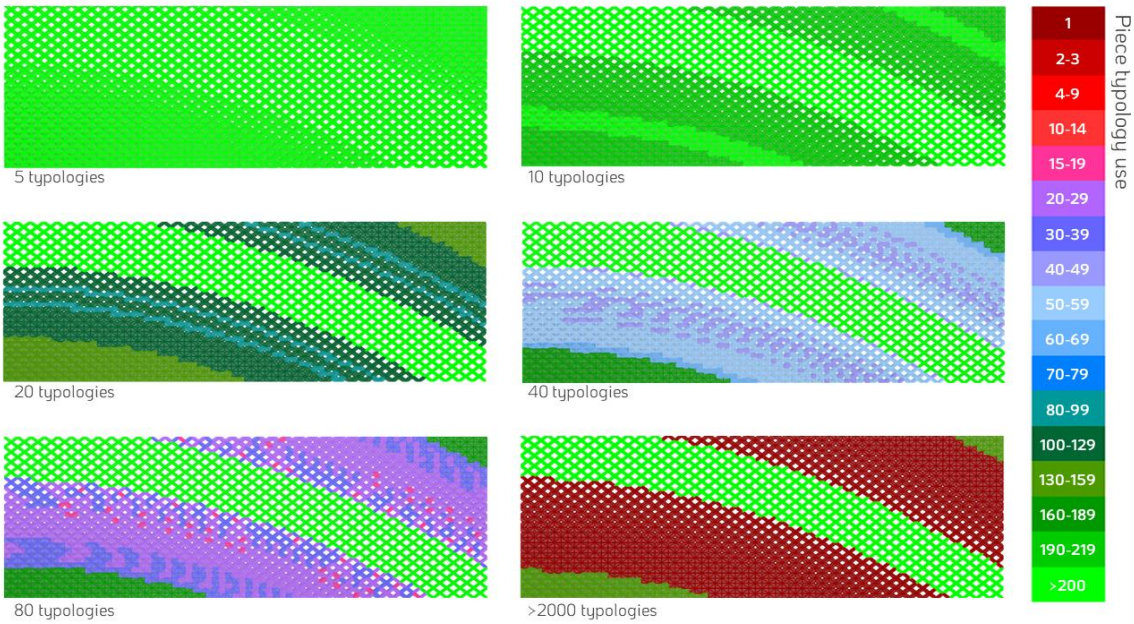


Figure 7.74. Visualization of the typologies' frequency of use for different rationalization levels using a color scheme.

7.3.6. FABRICATION

The previous sections addressed the framework's algorithms to geometrically explore, analyze, and rationalize different facade design solutions. This section focuses on those that materialize the developed solutions through different manufacturing strategies and materials. Figure 7.75 presents a set of examples where the different facade elements, whether in terms of shape, size, and material, among others, require different manufacturing, assembly, and support strategies.

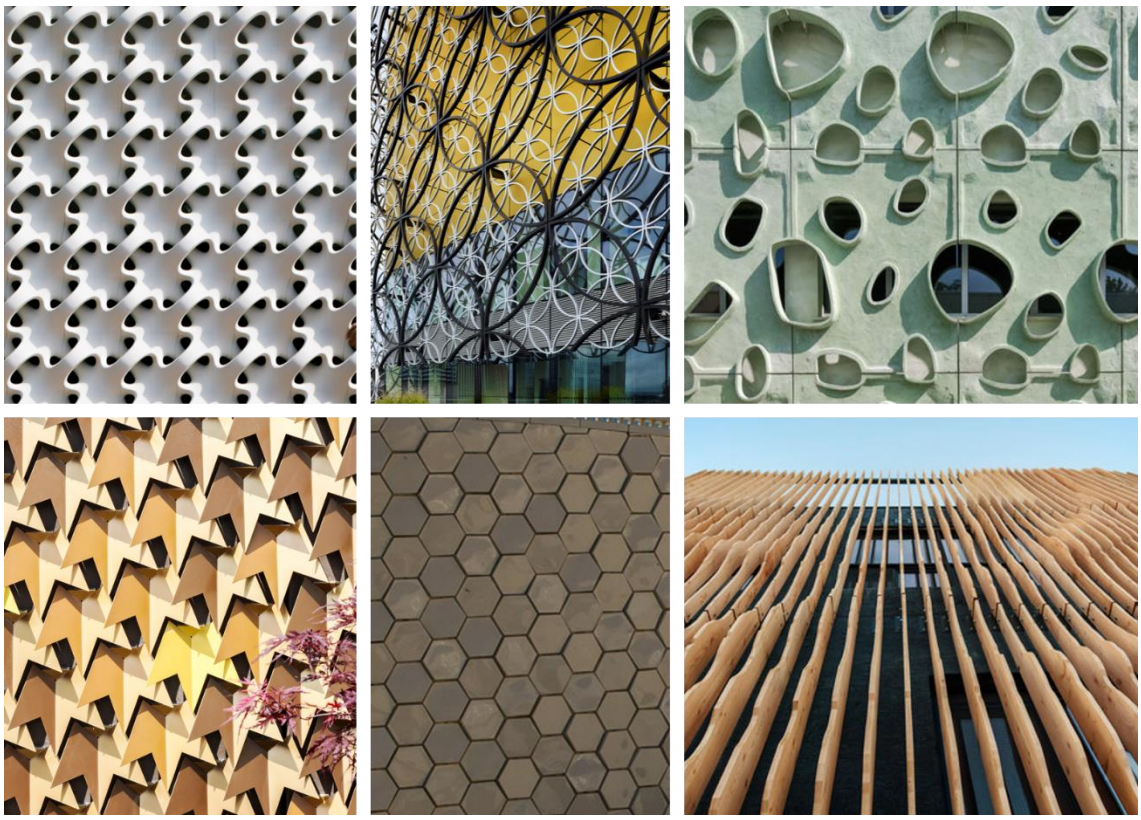


Figure 7.75. Cobogó House by studio mk27, Brazil (©nelson kon); Library of Birmingham by Mecanoo, UK (©Christian Richters); Institute of Bio-Sustainability by Cláudio Vilarinho, Guimarães (©Joao Morgado); Mayfair House by Squire and Partners, London (©Gareth Gardner); Leixões Cruise Terminal by Luís Pedro Silva Arquitecto, Portugal (©Fernando Guerra | FG+SG); Junction 9 by MODA, Canada (©MHB Photo-graf).

In general, this last category contains algorithms to:

- Extend the solutions with construction details.
- Automatically produce technical drawings according to different manufacturing strategies.
- Increase the control over fabrication processes.

In practice, when receiving the shape(s) to detail and/or produce, these algorithms analyze their geometric characteristics, adapting their algorithmic structure according to the selected (1) construction detail(s), in the first case; and (2) manufacturing scenario, in the remaining ones.

The first set includes algorithms such as $detail_{screwhole}$, $detail_{anglebracket}$, $detail_{uprofile}$, and $detail_{panelend}$, that, given the shape to detail and a set of detail characteristics, automatically extends it with, in this case, screw hole(s), angles bracket(s), u-profile(s), or panel ends, automatically fitting the elements' different sizes and shapes.

The second set includes algorithms such as $fab_{3Dprint}$, $fab_{forming}$, and $fab_{lasercut}$, among others. When provided with the facade design to produce, they convert it into different representation schemes (Figure 7.76): for *3D printing* the original shapes are converted into a set of printing paths; for *sectioning* strategies they are translated into a series of profiles whose superposition creates the desired surface or structure; for *forming* techniques they are transformed into their 3D negative shapes for casting purposes; and for *cutting* they are converted into two-dimensional planar curves delimiting the areas to cut, engrave or fold. The result is a new algorithm representing the original solution ($shapes$) according to the specifications of the selected manufacturing strategy, from where all the fabrication information and technical documentation needed can be automatically extracted.

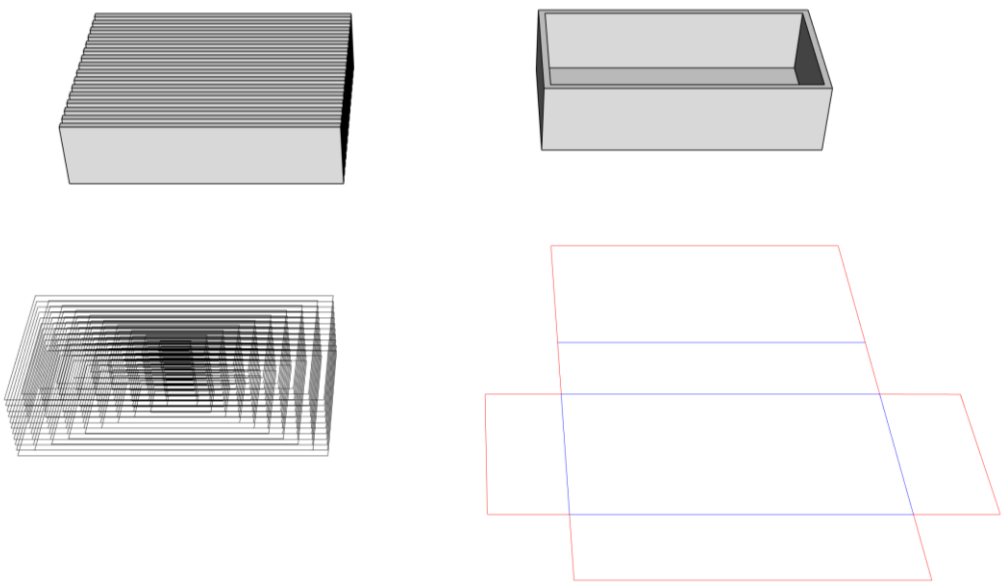


Figure 7.76. Automatic production of technical documentation for different manufacturing techniques: 3D printing, sectioning, laser cutting, and casting.

Finally, the last set includes algorithms to control material waste and costs of a given solution, enhance the efficiency and accuracy of its manufacturing process, or label its different elements for assembly purposes. When combined with the previous algorithms, they provide architects with a high-level control over the solutions' manufacturing and on-site assembly processes, increasing both the viability and quality of their production: e.g., by planning the 3D printing paths in different ways we can obtain various levels of element density, structural integrity, and surface quality, among others; by

manipulating the profiles' thicknesses in sectioning strategies we can achieve different element materialities and visual effects, among others; by optimizing the planning of the technical drawings through nesting techniques we can reduce material usage and costs; by labelling the produced elements we can more easily position and fit them in the final solution, etc.

To better illustrate these functionalities, we apply them to produce the technical documentation needed to manufacture the set of customized bricks of Figure 7.77 (left) through different strategies and materials. As a first example, imagine we select the 3D printing algorithm ($fab_{3Dprint}$). We automatically obtain a set of 2D paths for the printer head to pass creating the intended three-dimensional shapes. 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, different distances between them, and different trajectories. Accordingly, the resulting machining time, surface finishing, geometric accuracy, and material used also change [373].

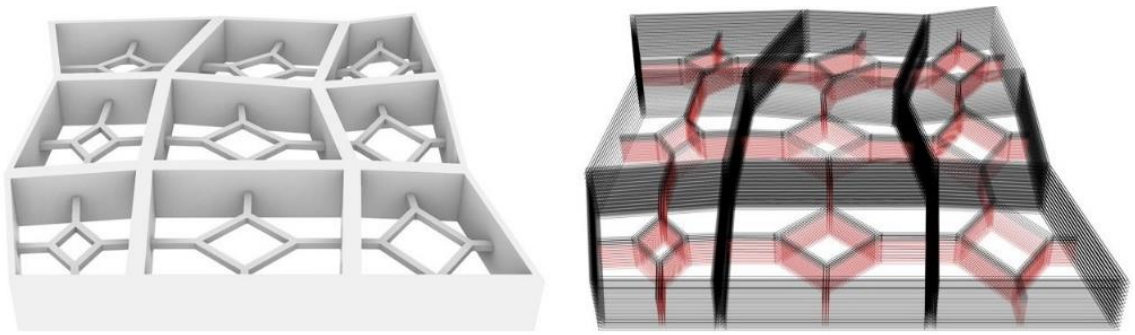


Figure 7.77. The original geometric model and the resulting printing paths (black lines) and support structures (red lines).

Figure 7.77 illustrates the result of combining the algorithm $fab_{3Dprint}$ with those producing the customized elements ($shape_{cobogo}$), where $head_{mm} = 3$, $layer_{mm} = 1.5$, and the planning type is based on alternated paths by level. Regarding the resulting algorithmic representation, this conversion of 3D elements ($shape_{cobogo}$) into 2D paths ($shape_{3Dpaths}$) requires changing its algorithmic structure, the originally used geometric solid primitives being replaced by paths operations, whose distribution along the generated volume is controlled by the previous parameters:

$$fab_{3Dprint}(shape_{cobogo}) \rightarrow shape_{3Dpaths}$$

As another example, imagine we want to produce the same facade elements through casting strategies. We select the *forming* algorithm ($fab_{forming}$), automatically obtaining the 3D models of their negative

shapes (Figure 7.78), together with cost-related information, such as the number of different molds and material quantities. As, in this case, both original ($shape_{cobogo}$) and translated models ($shape_{negative}$) correspond to 3D, albeit inverse, geometric shapes, the resulting algorithm is also composed by geometric solid operations but organized in a slightly different way:

$$fab_{forming}(shape_{cobogo}) \rightarrow shape_{negative}$$

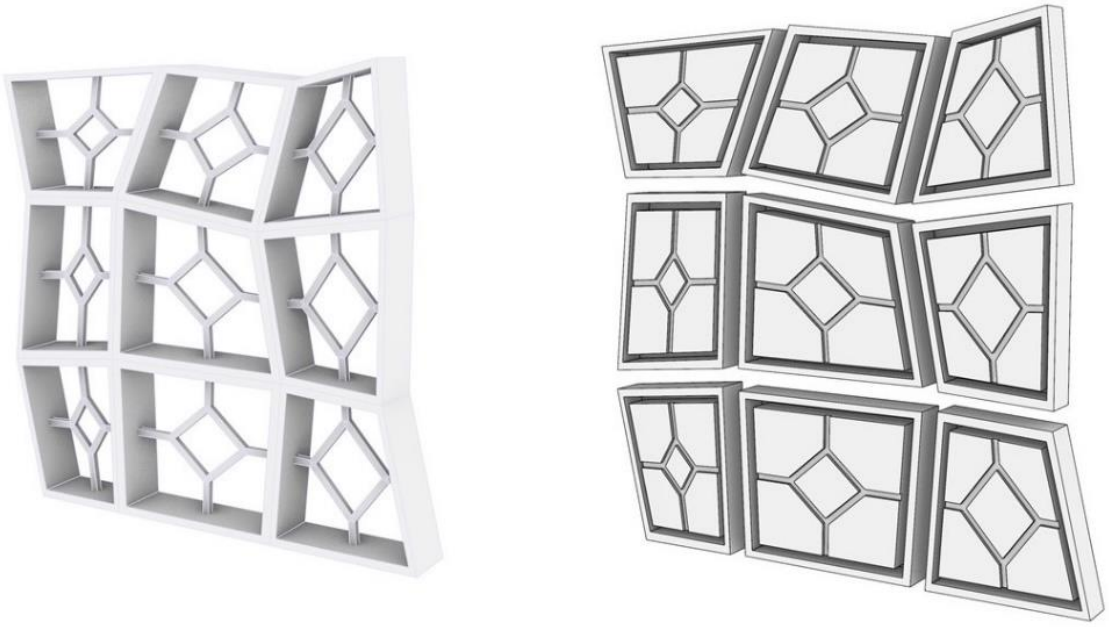


Figure 7.78. The original geometric model (left) and the corresponding negative shapes (right).

Having the molds' 3D models, the possibilities for their production are the same as for the brick elements, allowing us to select, for instance, the algorithm $fab_{3Dprint}$ and plan the paths for their 3D printing or the algorithm $fab_{milling}$ and set the instructions for their CNC milling:

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

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

Lastly, imagine we opt for manufacturing the previous elements through *sectioning* strategies. We select the algorithm $fab_{section}$, automatically obtaining a set of profiles, whose number is controlled by the $n_{profiles}$ parameter and whose juxtaposition creates the intended shapes. 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 material and element thicknesses are 10 mm and 90 mm, correspondingly, we set $n_{profiles} = 9$, obtaining the set of profiles illustrated in Figure 7.79.

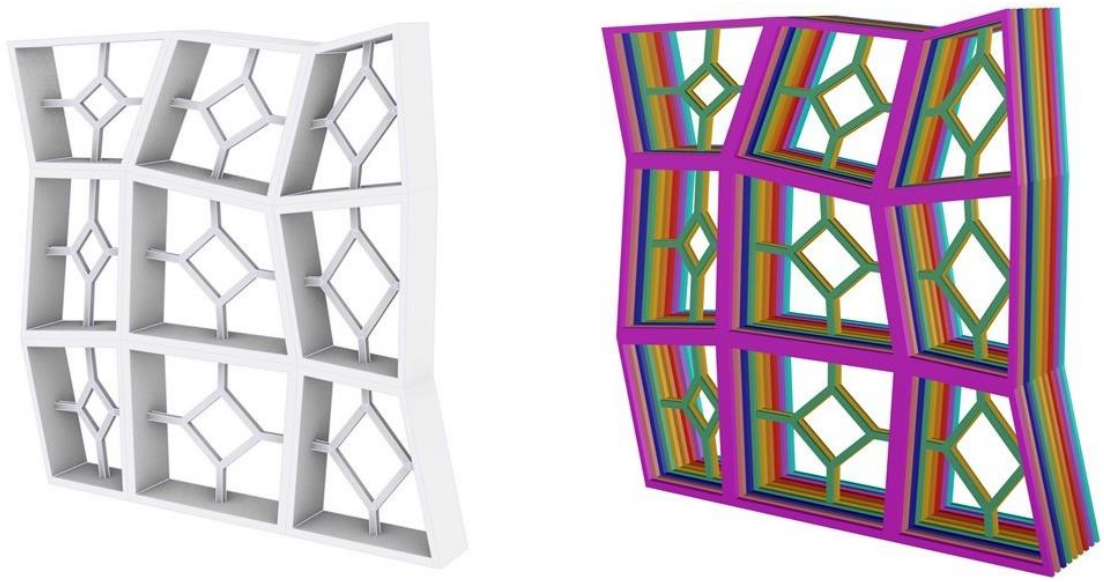


Figure 7.79. The bricks' geometric (left) and sectioned models (right).

Given the profiles' relatively small thickness (i.e., 10 mm), we can manufacture them, for instance, by using laser cutting techniques, selecting the algorithm $fab_{lasercut}$ to produce the 2D drawings of the areas to cut (Figure 7.80).

Regarding the resulting algorithmic representations, while the combination of $shape_{cobogo}$ with $fab_{section}$ causes mostly organizational changes, as both the original and converted algorithms use similar geometric solid primitives but in different ways, its combination with $fab_{lasercut}$ causes more radical changes, as the previous primitives are replaced with line operations defining the elements' contour shape:

$$fab_{section}(shape_{cobogo}) \rightarrow \{shape_{layer01}, \cdots, shape_{layern}\}$$

$$fab_{lasercut}.(shape_{cobogo}) \rightarrow shape_{contour}$$

In sum, these algorithms reduce the time and effort spent with manufacturing-related tasks, while increasing the accuracy and quality of the produced solutions. They therefore facilitate the exploration of different construction possibilities, assessing their different advantages and disadvantages in terms of production times, material waste, and overall costs, as well as their impact on the solutions' aesthetic quality, geometric precision, and physical properties.

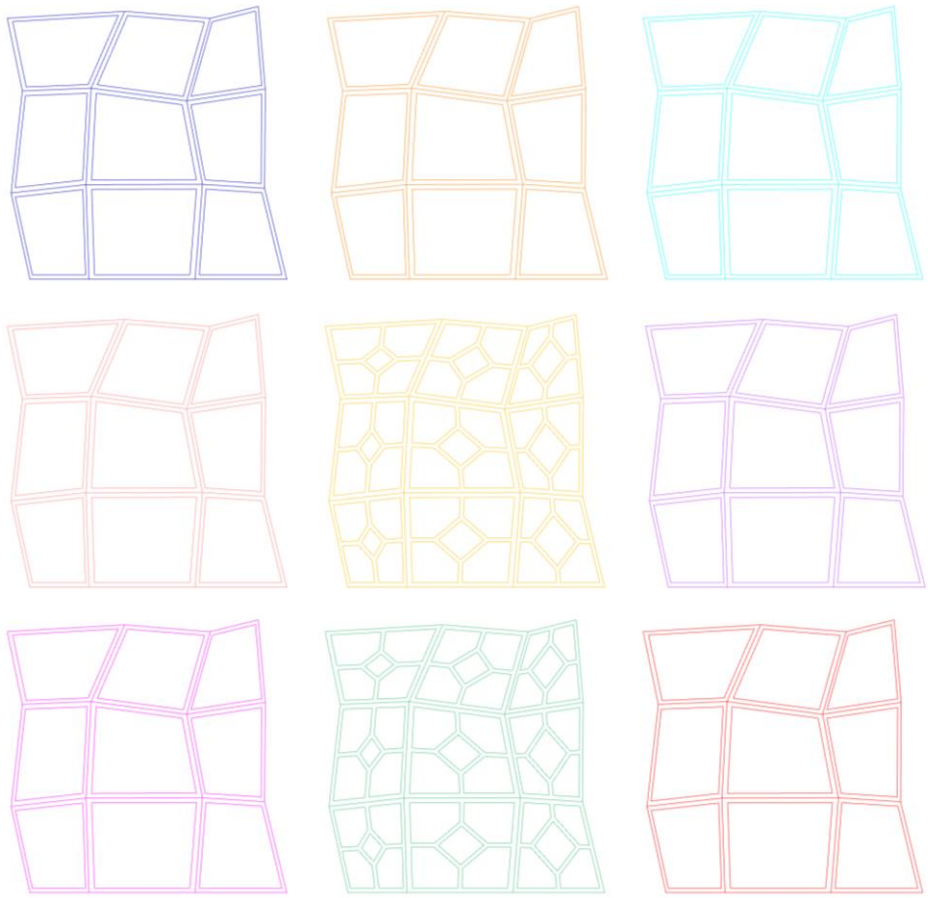


Figure 7.80 The two-dimensional drawings of the sectioned model's profiles for laser cutting.

7.4. CHAPTER OVERVIEW

This chapter presented the proposed framework for facade design, illustrating the structure and application of its mathematical principles with a set of conceptual examples.

To avoid making this chapter too long and repetitive, only part of the available strategies in each category were presented, their mathematical structure and combination serving as example for the remaining ones. Given the hierarchy imposed by the procedural development of the proposal, which first tackled geometry-related aspects of facade design (categories 1 to 3) and only then technical and manufacturing-related ones (categories 4 to 6), its different categories do not have the same level of development. Nevertheless, the goal of this thesis is to lay the foundations for solving the research problem addressed, allowing the proposed framework to be extended with more principles and eventually additional categories when the need arises.

The next chapter evaluates the proposed framework for facade design in a set of architectural application studies, applying the formulated general principles in practice-based design scenarios.

IV

Evaluation

8. FRAMEWORK APPLICATION

The previous chapter explained the framework's categorical structure and mathematical principles, illustrating its ability to reproduce an already existing corpus of facade designs. This chapter evaluates the framework's ability to apply the formulated principles to generate a novel corpus of facade designs responding to different design practices and problems. To that end, the framework is implemented on an Algorithmic Design (AD) tool, in this case Khepri [246], and applied in the development of a set of architectural application studies involving different creative intents and design problems. The choice for Khepri is due to its portability between different CAD, BIM, analysis, and visualization tools, allowing the framework's mathematical principles to benefit from the compatibility ensured by this AD tool.

Figure 8.1 illustrates the AD workflow adopted in the application studies: it starts with the design intent (arrow 1) and its implementation using the framework (arrow 2); it continues with the design's geometric exploration (arrow 3) and the testing of several variations considering both the design intent and the established performance criteria (arrow 4); finally, when satisfactory results are achieved, it assesses the solution's feasibility (arrow 5) in terms of cost and material waste, while preparing it for manufacturing. The result is a set of solutions balancing design intent, performance criteria, and feasibility (design space) among which we can then choose the one that most pleased us (arrow 6).

The next sections present five of the application studies developed using the framework. Based on the results and feedback received from the application studies' participants, the next chapter assesses the applicability and usefulness of the proposed principles for real-world design scenarios, identifying the strengths and weaknesses of the proposed mathematics-based methodology and framework, as well as concepts and requirements that may be relevant to address in the future.

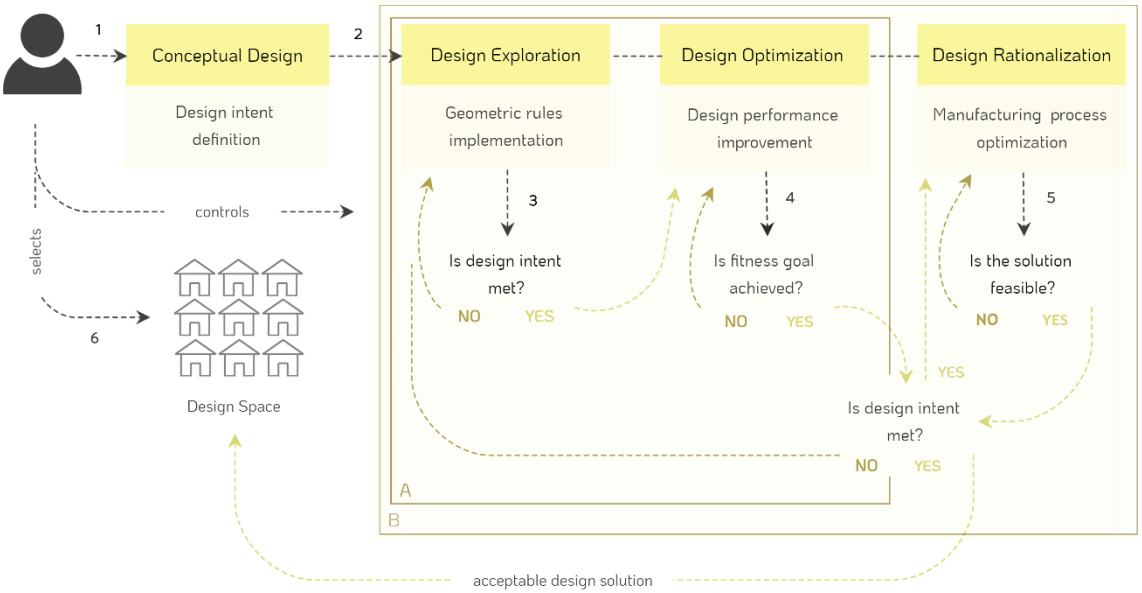


Figure 8.1. AD workflow: design intent definition (1) and implementation using the framework (2); design geometric exploration (3) and improvement regarding a certain fitness criterion (4); control of the design's manufacturing cost and waste (5) and selection of the final solution (6); A. iterative process of geometrically adjusting the design to meet the fitness goal(s) set; B. iterative process of balancing design intent, performance, and feasibility.

8.1. STUDY 1

This application study assesses the framework's ability to generate facade design solutions based on their structural performance from early stages. To that end, we adopt the methodology of Figure 8.2 in the development of a glazed truss-like facade structure inspired by the Blue Crystal building (Figure 8.3).

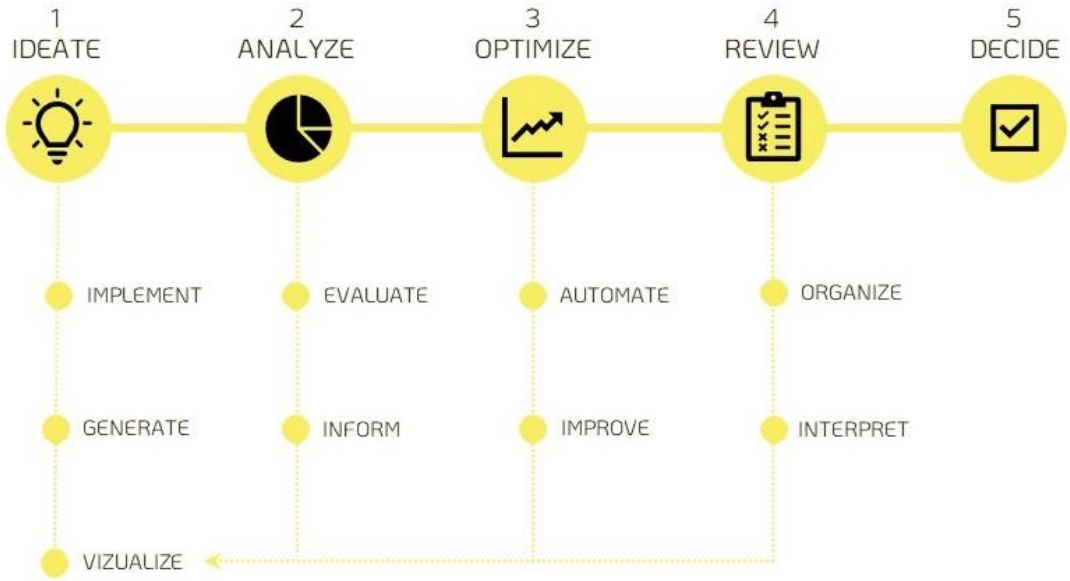


Figure 8.2. (1) Implementation, generation, and visualization of the design intent; (2) design space navigation based on iterative structural analyses; (3) design optimization by automating the search for better-performing solutions; (4) organization and interpretation of the optimized design space; and (5) final decision.

Figure 8.3. Blue Crystal building in Anand, India, by KPA Deesign Studio and constructed in 2019 (©KPA).

8.1.1. Algorithmic Implementation

The first stage entails the implementation of the design intent in a parametric algorithm describing both its geometric principles and degrees of freedom, which include the irregularity of the truss configuration, the amplitude of its three-dimensional effect, and the number of triangular panels. To that end, we select the most suitable algorithms to (1) compute the truss' irregular configuration; (2) create its metal profiles and glazed panels; (3) produce different geometric patterns on the panels; and (4) generate horizontal support bars. Figure 8.4 illustrates this process.

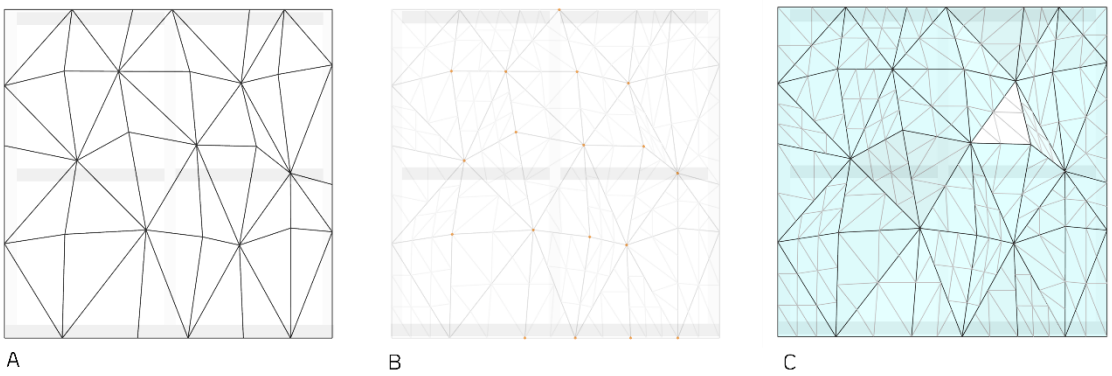


Figure 8.4. Design intent implementation: creating the truss bars (A), horizontal supports (B), and panels (C).

8.1.2. Performance-based Geometric Exploration

This stage focuses on the geometric exploration of the truss-like facade by considering both its design intent and structural behavior. After defining its material properties and expected loads, we iteratively explore the design space in a process guided by structural analyses and the visualization of the corresponding 3D models. Rather than obtaining accurate results, the goal at this stage is to get a general idea of how the design behaves to guide future design changes, not only identifying potential structural inconsistencies early on, but also enhancing the perception of the impact of each design variable on its visual expression and structural performance.

Figure 8.5 illustrates this process with an initial version of the design, whose structural analysis reveals, first, errors in the developed truss, such as the existence of overlapping truss bars and the lack of physical connections in some corner nodes (example A); then, after fixing the errors, the occurrence of structural instabilities deriving from sets of coplanar nodes (example B); and lastly, the truss' tendency to deform asymmetrically due to the different length sizes of its horizontal supports (example C).

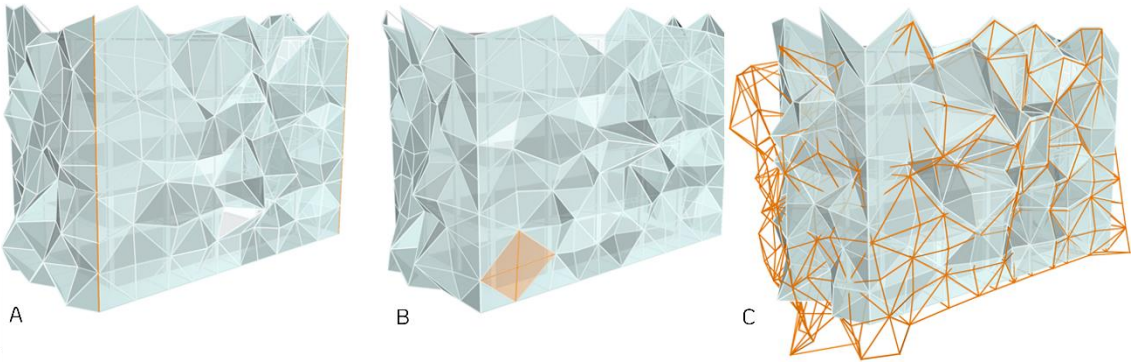


Figure 8.5. Iterative structural analysis using default truss bar materials and cross-sections and gravitational loads automatically calculated from the truss self-weight: A. overlapping truss elements and insufficient physical connections; B. node coplanarities; C. truss deformation tendency. At each iteration, the design was changed according to the results and reanalyzed to check if all problems were solved.

When a higher level of detail is needed, we extend the algorithm with additional (1) geometric constraints, namely the irregularity and amplitude of the crystal-like structure, (2) physical properties, namely the type of supporting systems, materials, and sections, and (3) environmental variables, such as wind loads, iteratively repeating the previous *generation-analysis-regeneration* cycle with different combinations of values.

8.1.3. Structural Optimization

To make the search for the best solutions viable in terms of time and labor, we couple the previous algorithm with an optimization step, automating the previous cycle towards the goals of (1) meeting the design intent and (2) minimizing both the structure's displacement and cost. Despite the latter being a typical case of conflicting goals [374], since the stronger the structure is, the more expensive it becomes, the aim at this stage is not to find the strongest structure but rather the least expensive one that complies with safety standards [282].

Since it is not easy to describe creative intents in optimization processes nor to understand their impact on the solutions' performance beforehand, it is important to be able to compare the solutions found both quantitatively, i.e., in terms of performance quality, and qualitatively, i.e., in terms of visual expression. To that end, we also couple the framework to an external visualization library, allowing us to organize the optimization results in interactive scatter plots where each point corresponds to a design solution that can be visualized by clicking on it.

To perform the optimization, we set as decision variables (Figure 8.6) the truss' (1) number of panels, (2) grid irregularity, (3) inner, outer, and top amplitudes, and (4) bar material and section size; and as optimization goals its (5) maximum displacement and (6) cost. We also set the range of acceptable values for the latter two requirements to ensure that both the safety standards and budget are met, not only avoiding solutions above those limits (Figure 6, green area), including Pareto-optimal ones, but also considering as acceptable solutions other than those of the Pareto front that may be

preferable in terms of visual expression. Finally, we select as optimization algorithm the genetic algorithm NSGAII [375], due to its popularity within the community and its suitability to deal with architectural optimization problems [278, 376], particularly those addressing structural analysis [377].

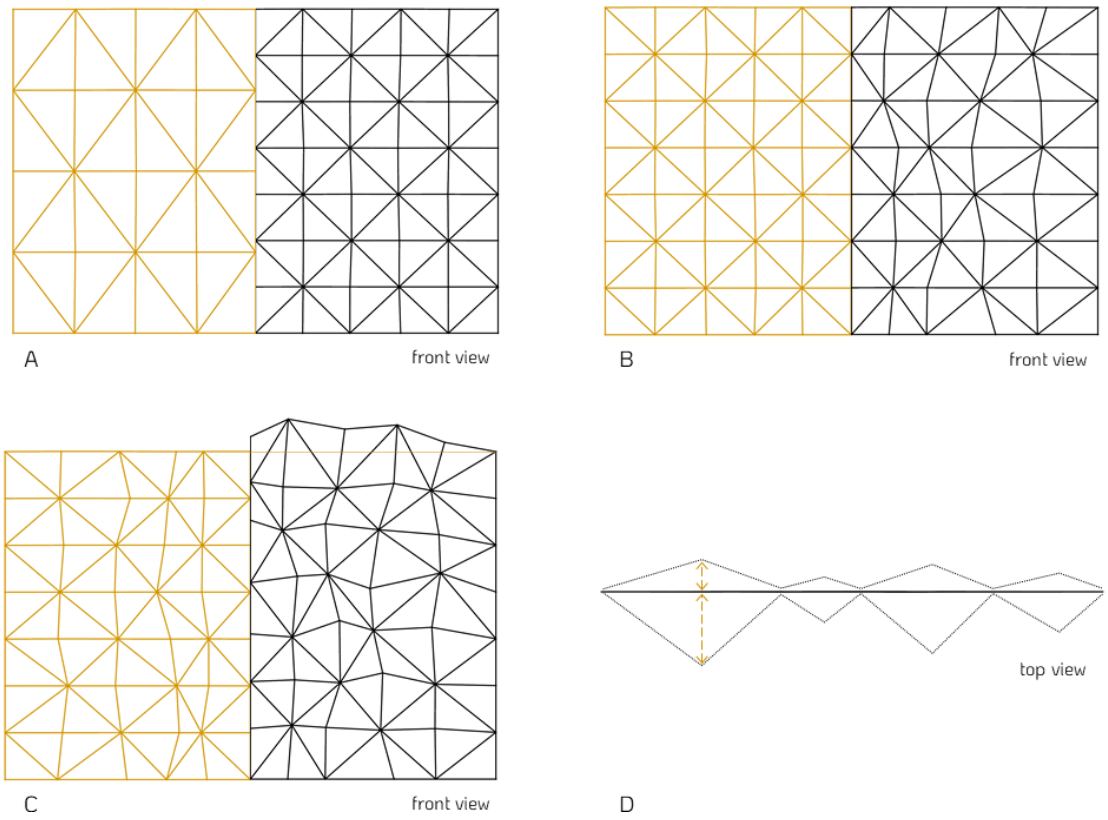


Figure 8.6. Decision variables: A. number of triangular panels; B. grid irregularity; C. top amplitude; D. inner and outer amplitudes.

By performing iterative optimization cycles considering both the previous analysis results and the impact of each decision variable, the likelihood of finding better-performing solutions increases. This process is illustrated in Figure 8.7, first, with different materials, each corresponding to an orange dot in the graph, and then, with both different materials and section sizes, each combination (material and section) corresponding to the yellow dots. In both cases, the remaining variables are kept fixed with the values found in the previous iteration. The results, however, demonstrate that changing only these two variables is not enough to meet the established goals since none of the obtained solutions match the acceptable area (Figure 8.7 green area). After repeating this process by varying an additional parameter, namely, the inner amplitude, we obtain the results illustrated by the red curve in Figure 8.8. As they only result in slight improvements, we now also let the outer amplitude vary, and then the top amplitude, the results corresponding to the yellow and orange curves of Figure 8.8, respectively.

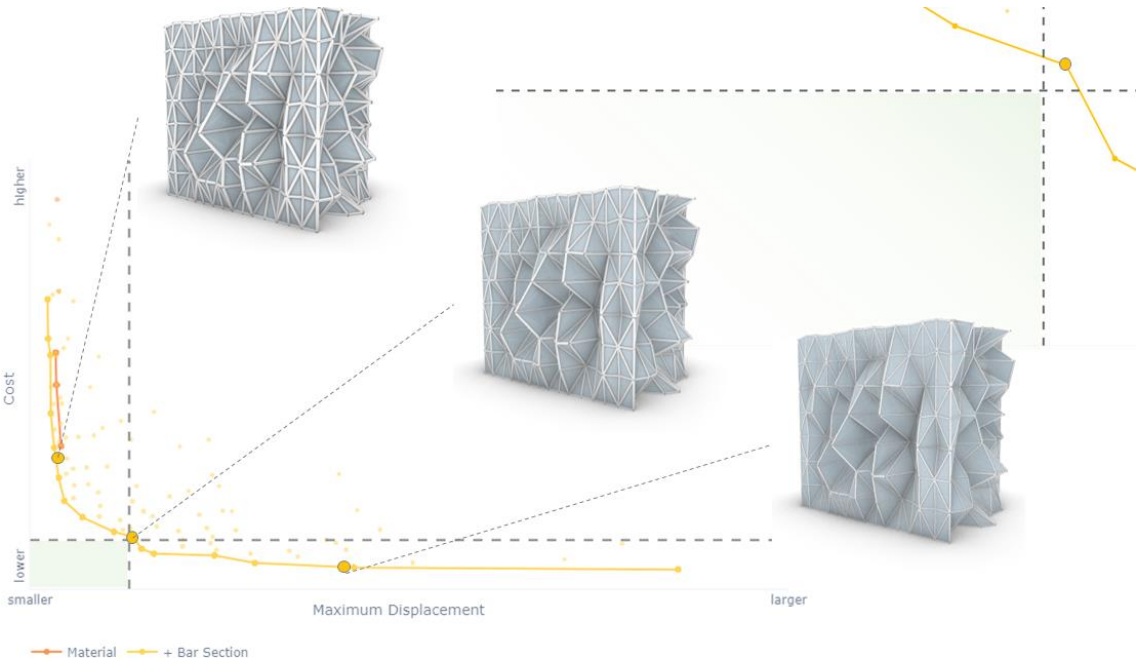


Figure 8.7. Truss optimization with fixed horizontal supports and different truss bar materials and section sizes: bottom left, the area of acceptable solutions identified in green, the horizontal and vertical dashed lines delimit the maximum acceptable cost and displacement, respectively; top right, a close-up view of the acceptable area.

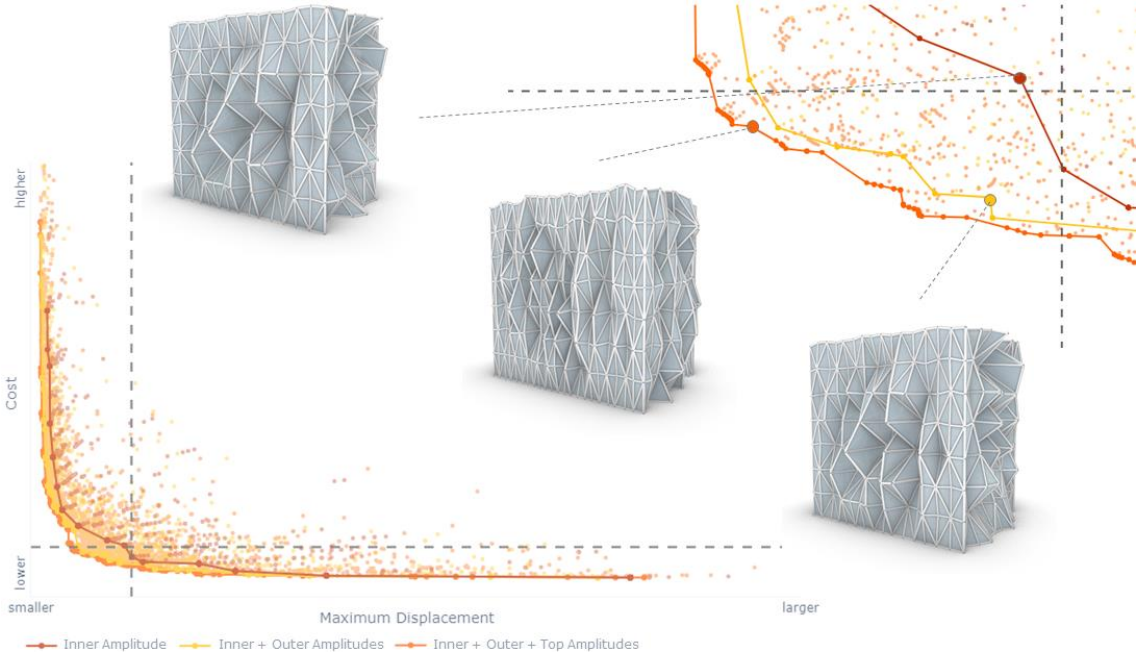


Figure 8.8. Truss optimization with the previous variables plus additional variables for different inner, outer, and top amplitudes of the crystal-like effect.

Although solutions with acceptable values have already been found at this stage, we continue the optimization process by varying, first, the number of panels (Figure 8.9, orange curve) and then, the truss grid irregularity (Figure 8.9, yellow curve). Despite greatly improving the results in terms of cost and strength, the design intent is however neglected, resulting in solutions with markedly vertical panels that deviate from the desired crystal-like effect (Figure 8.9).

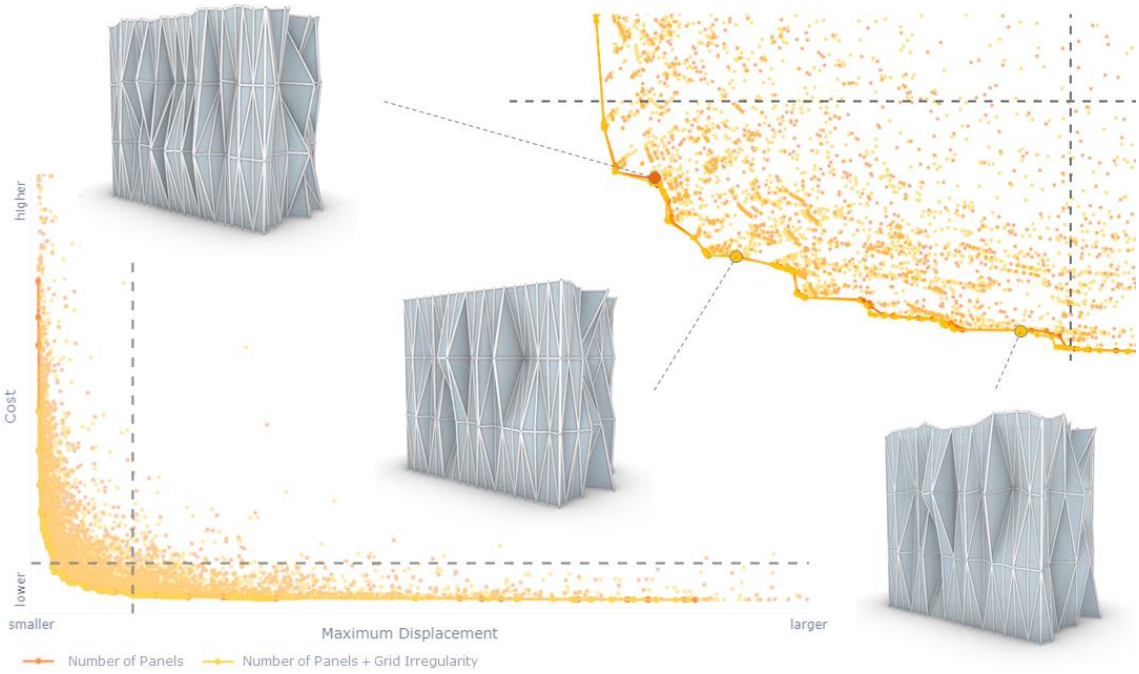


Figure 8.9. Truss optimization with the previous variables plus a different number of panels (orange) and grid irregularities (yellow).

Given the proximity of the previous Pareto fronts, we now repeat the optimization process with a fixed number of panels to see how it impacts the results. As is visible in Figure 8.10, despite the obtained solutions (red curve) not being as good as the previous ones (orange and yellow curves) in terms of cost and strength, they are nevertheless preferable in terms of design intent (Figure 8.10 bottom-right image). We could now continue repeating this process with different combinations of values, e.g., with a slightly lower number of panels or even additional pinned supports at the truss' bottom nodes (Figure 8.11), searching for improved solutions in an incremental, flexible way, coordinating creative and performance intents.

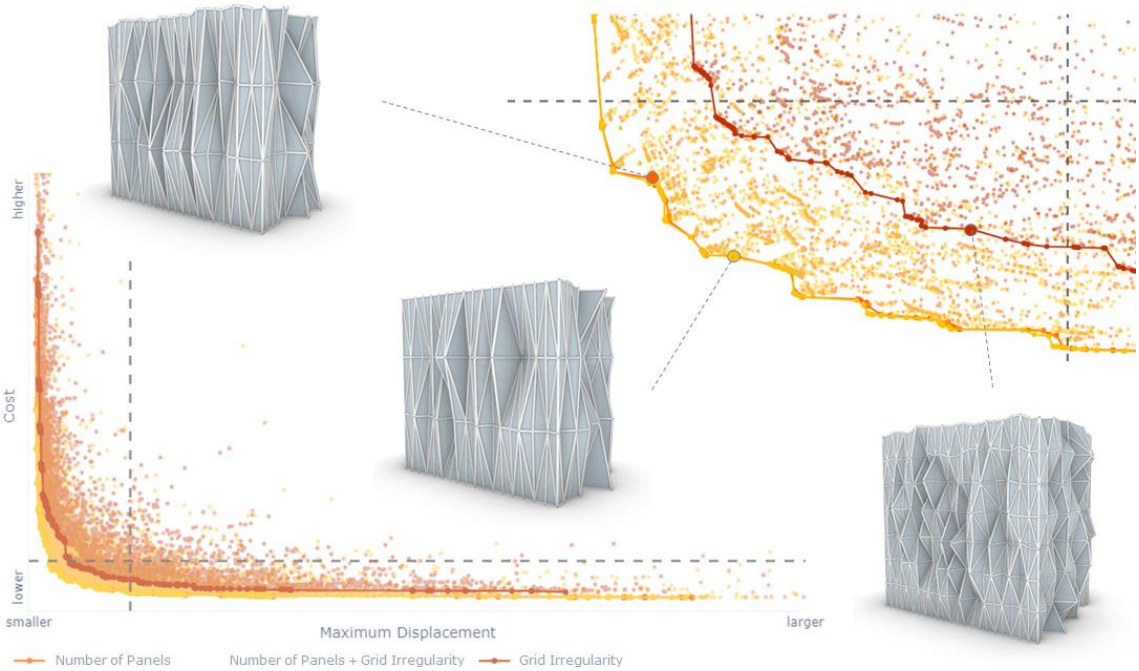


Figure 8.10. Comparison of the previous truss optimizations (orange and yellow) with one with the number of panels fixed at its original value (red).

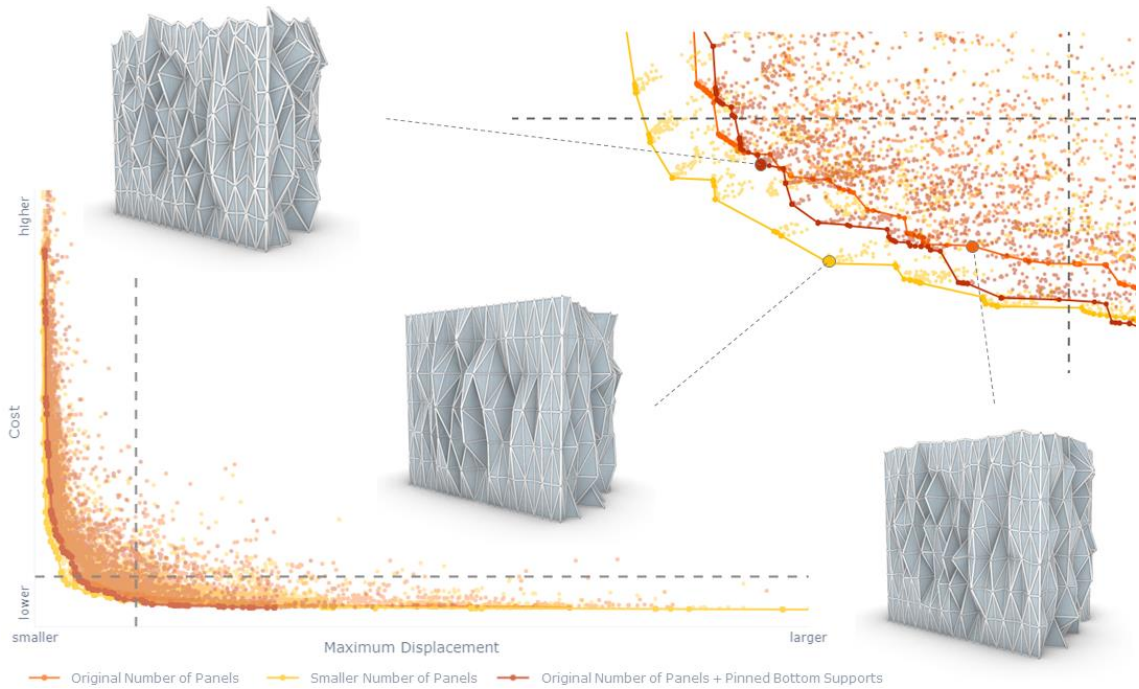


Figure 8.11. Truss optimizations with the number of panels fixed at its original value (orange), at a slightly value (yellow), and at its original value plus the bottom supports pinned (red).

8.1.4. Revision and Decision

To restrict the sample of possible solutions, this stage only considers the Pareto-optimal designs that we find to have the best trade-offs between creative intents, cost, and strength. Given the framework's ability to benefit from the graphical and interactive capabilities of external visualization libraries, the

presentation of the optimization results becomes more intuitive and also more informative for the decision-making process: by allowing us to more easily balance design intent and performance, i.e., by simply pointing to the dots in the graph and quickly visualizing the corresponding 3D models (Figure 8.12), we can navigate the optimized design space more consciously and efficiently and select the solution that we think best satisfies both.

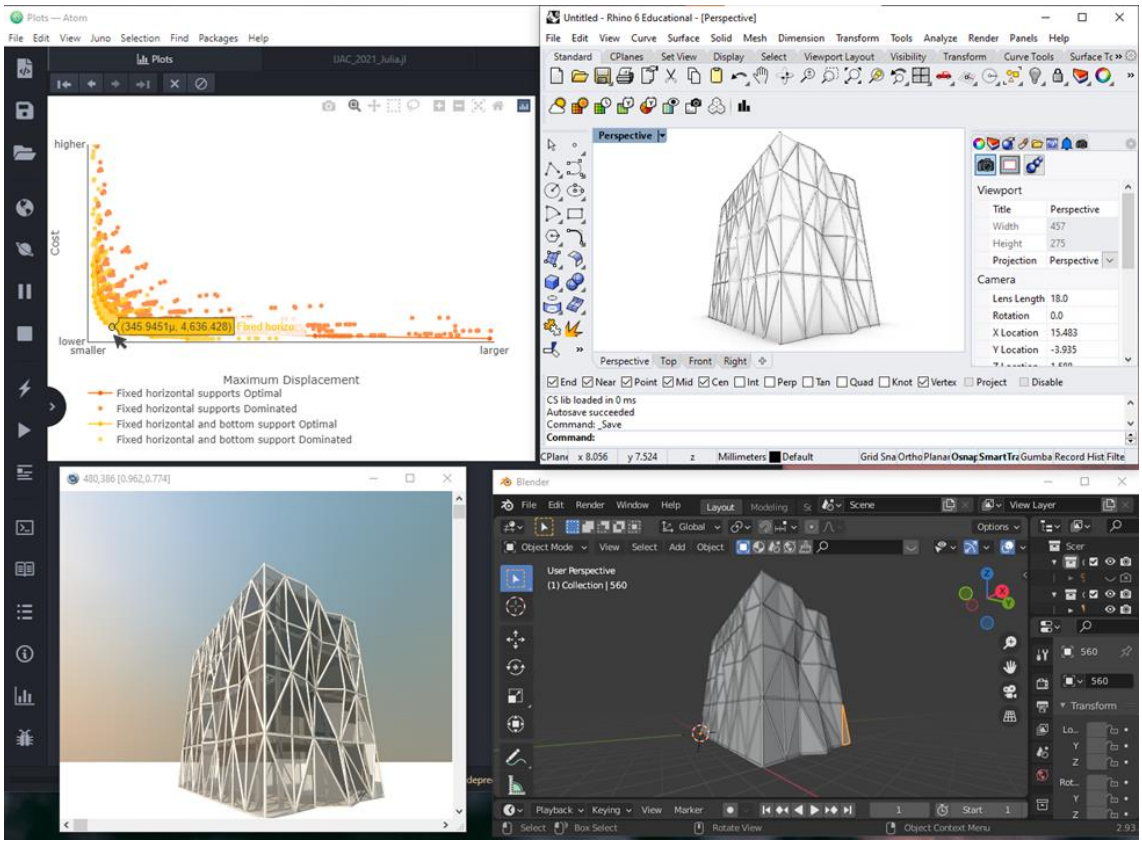


Figure 8.12. Visualization of the solution selected in the interactive graph (top-left) in multiple tools: a point-and-click in one of its dots triggers the generation of the corresponding 3D model in, for instance, Rhinoceros 3D (top-right), POVRay (bottom-left), or Blender (bottom-right).

8.1.5. Discussion

The previous results confirm that the framework supports design exploration processes responding to both creative intents and performance criteria, facilitating, in this case:

- The algorithmic implementation of the design intent, allowing us to benefit from the ready-to-use algorithms that can be easily combined.
- The incremental development of the resulting algorithm to include additional requirements, allowing us to iteratively test design variations and immediately see the results.
- The structural analysis of different solutions, allowing us to understand their structural integrity and the impact of each design variable on it.

- The optimization of the solutions in terms of structural performance, cost, and aesthetic preferences, allowing us to balance both quantitative and qualitative design criteria.

By reducing the time and effort spent on the previous tasks, we were able to focus on creative processes and add aesthetic-related constraints to the optimization process, such as the truss irregularity and the number of panels, and more efficiently explore the trade-offs between design intent, structural performance, and cost.

8.2. STUDY 2

The second application study results from a collaboration with the Portuguese design studio Atelier dos Remédios. It involved the design of a set of facade shading panels for a private house in Ericeira that addressed the complex daylight situation on both its South and West facades (Figure 8.13).

As the panels faced different sun exposures, their design could not present the same geometric configuration nor the same opacity level. Additionally, to respond to the ever-changing daylight conditions and achieve a balanced illumination and ambience comfort, the panels should allow users to adapt their position accordingly. Lastly, to balance the initial design intent with both performance and material requirements, the panels' stereotomy should (1) be geometrically irregular, (2) soften the rigidity of the almost completely opaque North facade, (3) adapt to the house's different materials, and (4) adjust its opacity level to the adjacent inside space function.

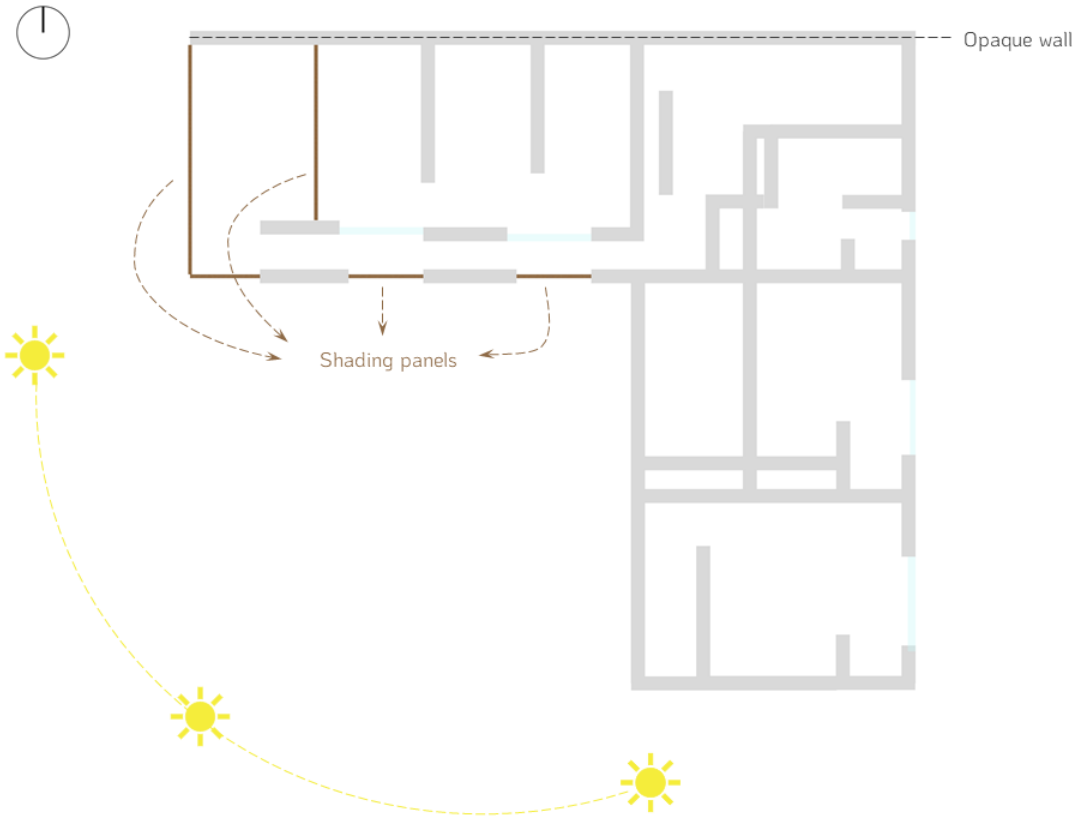


Figure 8.13. Private house conceptual plan and its daylight exposure.

8.2.1. GEOMETRIC EXPLORATION

During this stage, the framework allowed us to easily explore different panel configurations in collaboration with the design studio in an iterative way, evolving the geometric pattern in a short period

of time. The result was a set of facade shading panels with a geometric pattern based on horizontal wood bars of different alternating sizes (Figure 8.14-A).

To achieve the desired geometric irregularity, the design studio suggested changing both the size and position of the smaller bars in a random way (Figures 8.14, examples B and C), while adding some restrictions to its random behavior to allow controlling the panels' shading levels. To that end, we extended the panels' algorithmic description with new variables, namely the maximum and minimum sizes (L_{min} and L_{max}) and the size increment (ΔL) allowed for the smaller bars, as well as the maximum distance (L_{max}) between them (Figure 8.15).

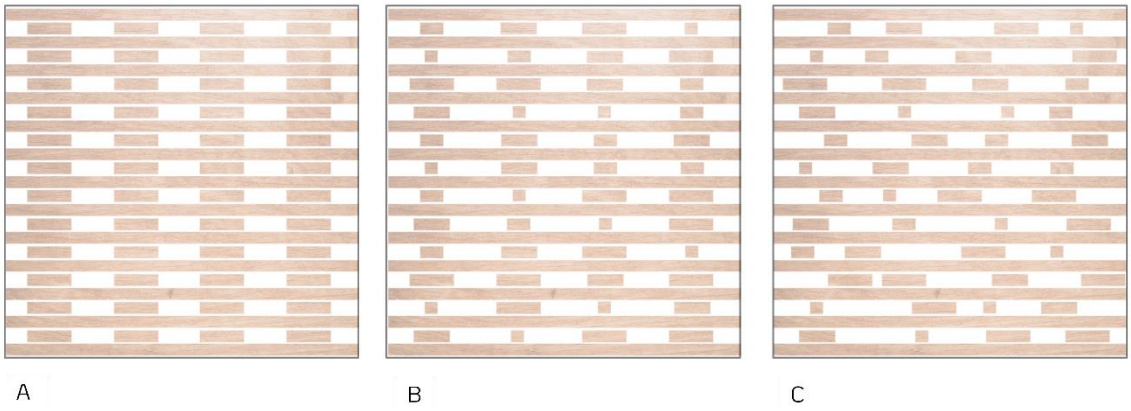


Figure 8.14. Facade panels' geometric evolution: alternating between one full-length bar and a set of equally distanced smaller bars (A); and randomly controlling the smaller bars' size (B) and position (C).

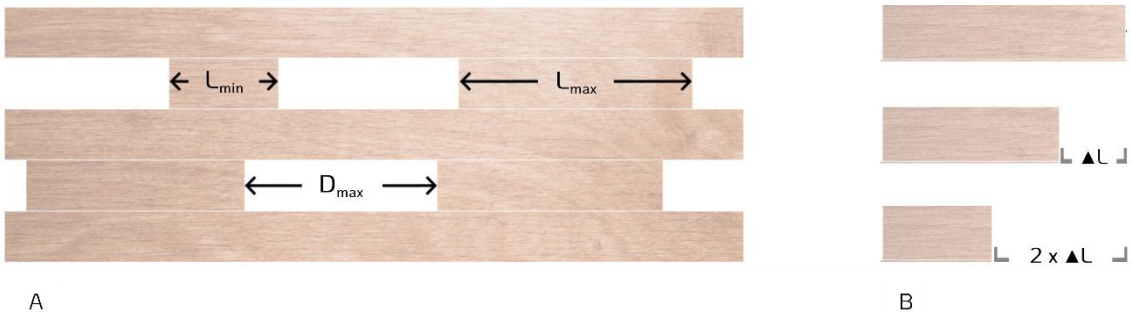


Figure 8.15. Additional geometric restrictions: A. maximum and minimum possible lengths and maximum distance between smaller bars; B. smaller bars' size increment.

The next step involved testing different design alternatives for the facade panels by simply assigning different values to its variables. This allowed the team to iteratively visualize and evaluate a wide range of solutions, while suggesting improvements to be implemented. The result was a collaborative design process based on the *generation-visualization-regeneration* of different solutions that facilitated the search for the best ones from an aesthetic point of view.

8.2.2. DAYLIGHT OPTIMIZATION

In addition to geometric complexity, the facade panels should also have a good daylight illumination performance. To that end, we coupled the framework with an optimization routine, selecting as performance metric the Spatial Useful Daylight Illuminance (sUDI) [378], which measures the daylight provision and levels of solar exposure by using percentual scales. This allowed us to automate the iterative application of design changes and the daylight analysis of the resulting solutions until an acceptable design was achieved. Given the short time available for this project and the effort required to manually perform each task individually, the use of the framework was critical, allowing us to generate and evaluate a higher number of solutions within the same period.

To reduce the time taken by each performance evaluation, we only considered the inside spaces containing facade shading panels. Then, to generate several panel alternatives, we used different sampling techniques, starting with Monte Carlo Sampling [379] to test the optimization workflow, and later transiting to Latin Hypercube Sampling (LHS) to reduce the number of candidates, while improving the coverage and variance of the design space [380]. This allowed us to achieve a design solution with 100% of sUDI but also with a high daylight glare probability (DGP), which reduced the inside spaces' comfort (Figure 8.16).

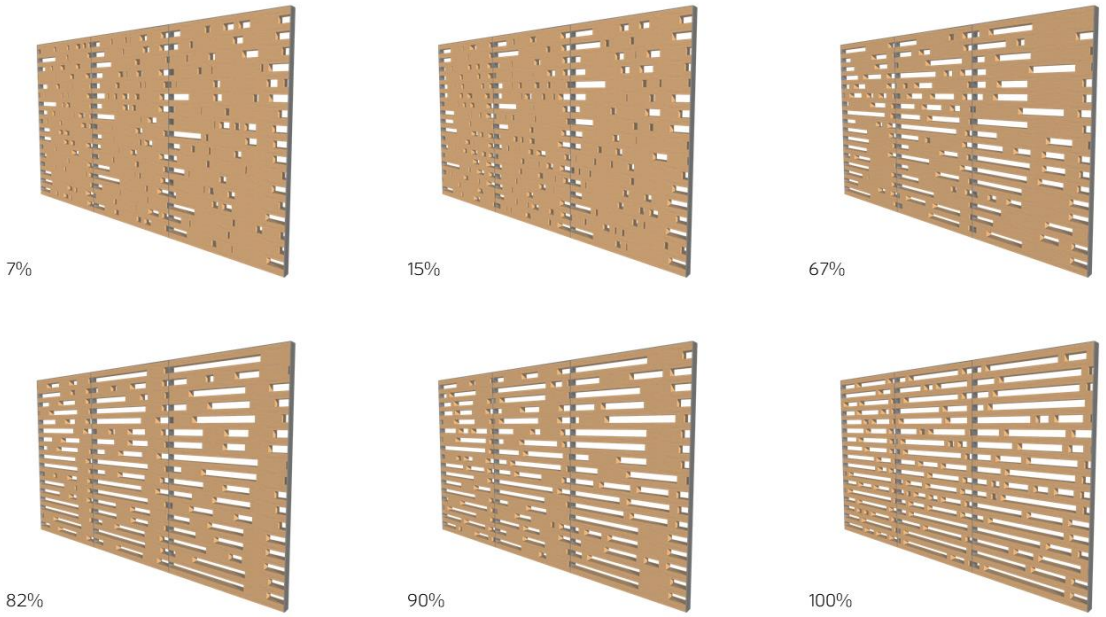


Figure 8.16. Six facade panel configurations with the corresponding sUDI levels.

We then repeated the optimization process with an additional cost constraint that limited the smaller bars' length to 5, 10, 15, 20, or 25 cm. We implemented this requirement by fixing the parameters L_{min}, L_{max}, and ΔL to 5, 25, and 5, respectively, and only allowing the distance between smaller bars (D_{max}) to vary, dictating the level of daylight entering the room.

The design studio suggested starting the optimization process by setting $D_{max} = 20\ cm$. However, after generating 50 samples, the best sUDI value achieved was around 45%, which was far from being optimal. We, therefore, repeated the process, this time using a $D_{max} = 100\ cm$ and generating 200 samples, the results being presented in Figure 8.17. Unfortunately, despite achieving solutions with higher sUDI values, most of them deviated from the initial design intent: for example, the best solution reached, solution G, presented an almost non-existent or even null random effect.

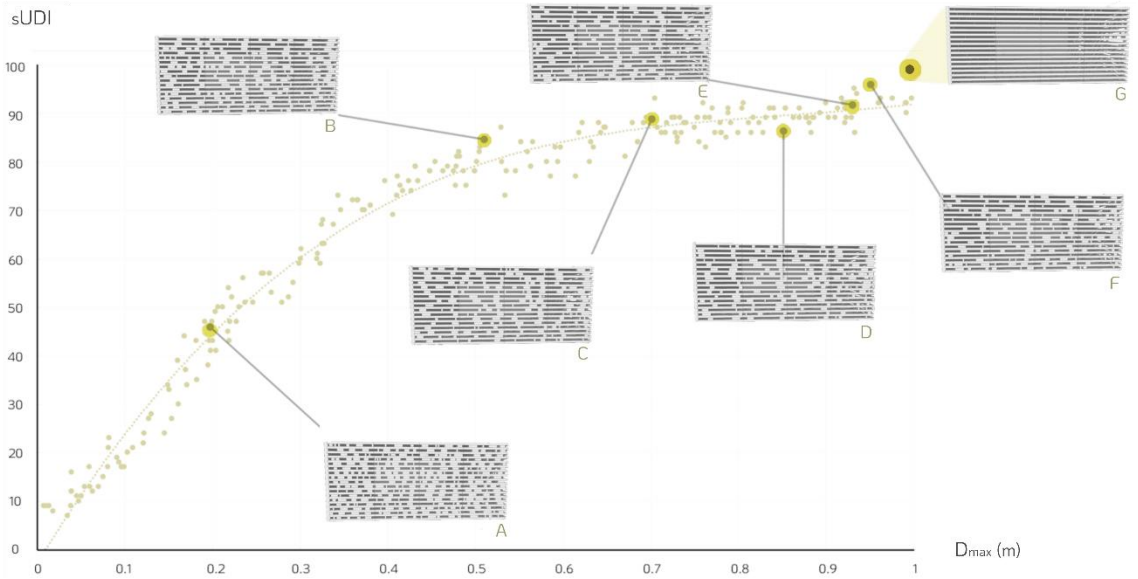


Figure 8.17. The results of the second optimization process (200 samples): solutions A to G correspond to the set of examples presented to the architects.

Given the high number of solutions obtained during this process, the next challenge was selecting one that had a good performance and matched the initial design intent. To that end, we decided to assess how much the design studio's initial suggestions restricted their final choice, evaluating the ease with which they accepted design options deviating from their design intent. We presented seven solutions to them (from A to G), which were carefully selected to be as heterogeneous as possible, without informing them about the variables' values and corresponding sUDI values. The final sample contained one solution fitting all the constraints proposed by the design studio (option A), $D_{max} = 20$, $L_{min} = 5$, $L_{max} = 25$, and $\Delta L = 5$; five solutions matching all constraints except the distance between bars (options B to F); and one solution not considering most of the constraints (option G).

After analyzing the seven options, the design studio selected option C as the best solution, option D as the second-best solution, and option A as the worst solution, which ironically was the solution that matched all their requirements. Based on the results, we concluded that the solutions with more balanced characteristics, i.e., with an acceptable sUDI value (higher than 80%) and close enough to the design intent, were the preferred ones. In contrast, none of the solutions with the highest sUDI values (options E, F, and G) was selected, meaning that these were not considered as good at responding to

the design intent as other solutions like options B and C. Nevertheless, they were also not considered the worst solutions, demonstrating that even when the design deviates from the initial design intent, as is the case of option G, it can still be considered as a possible solution. Regarding the worst solution, option A, although it had the worst sUDI value, it matched all the geometric constraints, but the design studio considered the panels' geometric pattern as excessively dense.

As a result of this process, the team included an additional design constraint, the *percentage of non-opaque areas*, which, in this case, should be at least 50%. After implementing this requirement, we automatically repeated the entire optimization process, obtaining, in the end, a solution that successfully met all the requirements set: responding to the design intent, having good daylight performance, only using small bars with sizes $\{5,10,15,20,25\}$, having a maximum distance between small bars of 20 cm, and presenting less than 50% of opaque area. Figure 8.18 shows the final solution in Revit.

Figure 8.18. A perspective of the house's north and west facades.

8.2.3. Discussion

This application study was important to evaluate the framework's ability to adapt to different design practices and tools, while enhancing both creative design and decision-making processes. In this case, despite having no experience in AD, the design studio could still benefit from the framework's flexibility during the design exploration process, by constantly suggesting design changes and incrementally refining its geometric configuration, as well as during the design development stage, by gradually improving the design in terms of daylight and privacy levels. They could also benefit from its portability and use more performant and flexible design tools during the design exploration phase (in this case Rhinoceros 3D), to navigate a wider design space, and after deciding on a panel solution, effortlessly

generate its corresponding model in the desired tool (namely Revit), obtaining the detail and construction information required. All these scenarios contributed to increase (1) the design space explored and thus the variety of solutions considered, (2) the perception of the impact of the design changes made on the results, and (3) the probability of achieving better solutions.

8.3. STUDY 3

The next application study is the result of a collaboration with two Portuguese design studios, Atelier dos Remédios and FOR-A Architects, in the context of an architecture competition. The aim was to create a facade design for a residential building in Lisbon that ensured different degrees of permeability and, simultaneously, communicated the idea of randomness. Given the tight deadline for its development (six days), and the fact that the design intent was not yet defined in terms of geometry and materiality, using AD was the only solution that provided the design flexibility and generation speed needed to quickly explore different designs. Unfortunately, the architects had no experience in AD and, clearly, there was no time to teach them. The only possibility was for us, architects with AD skills, to collaborate with them as specialized service providers.

Considering the short deadline and the advanced progress of the building design when we were contacted, the design studios continued designing the still-missing parts of the project and we developed the algorithmic facades in a three-stage process: two days to materialize the design intent and explore different design alternatives, two days to improve the chosen solution regarding a set of requirements, and two days to detail the solution for manufacturing. As the 3D model of the entire project was done in Revit, which is a relatively slow tool, we decided to take advantage of the flexibility and visualization speed of a CAD tool during the design exploration stages, and only, in the end, did we transit to the BIM tool to produce the final model with the construction information.

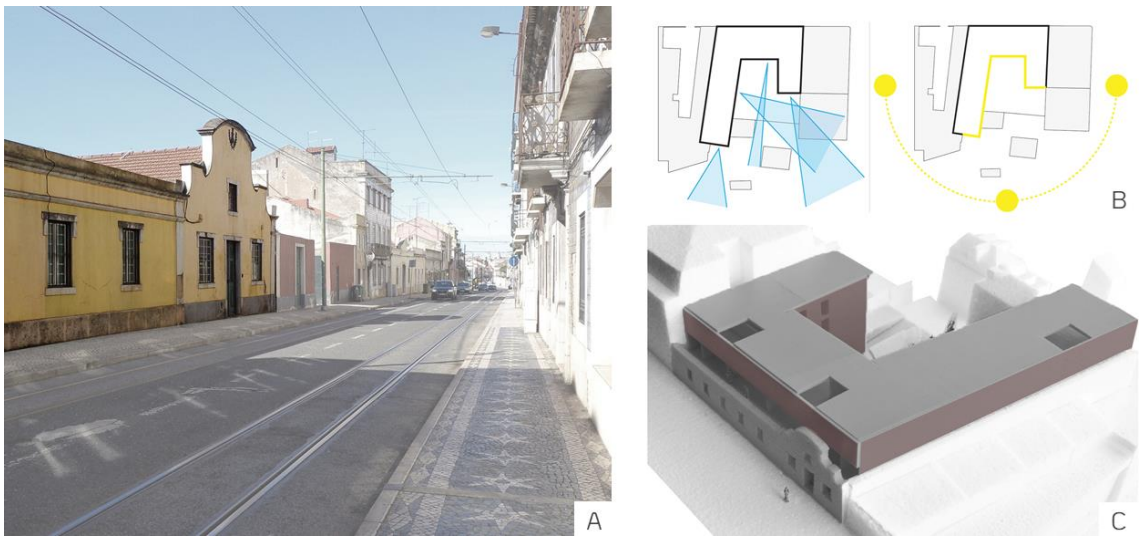


Figure 8.19. A. street view of the intervention lot; B. conceptual analysis of the dwellings' views and daylight; C. physical model of the intervention (developed facades are in dark red).

8.3.1. GEOMETRIC EXPLORATION

This stage involved the geometric exploration of three facade designs for the residential building, which had to consider three constraints: (1) the tight time limit of six days; (2) the need to incorporate a pre-existence of the intervention lot in the final design (Figure 8.19-A); and (3) developing three design solutions fitting the already defined building shape (Figure 8.19-C) and simultaneously meeting the design intent and existing design requirements (Figure 8.19-B).

Regarding the design intent, the aim was to create a facade design with different levels of opacity that varied according to the inside spaces' function: i.e., more permeable when coinciding with the dwellings' terraces and less permeable or almost opaque when covering more private areas, such as bedrooms and kitchens. To implement it, we used the framework and, since there were no constraints regarding the design's materiality at this stage, we addressed the concept of *permeability-opacity* through different materials, e.g., bricks, tiles, and concrete.

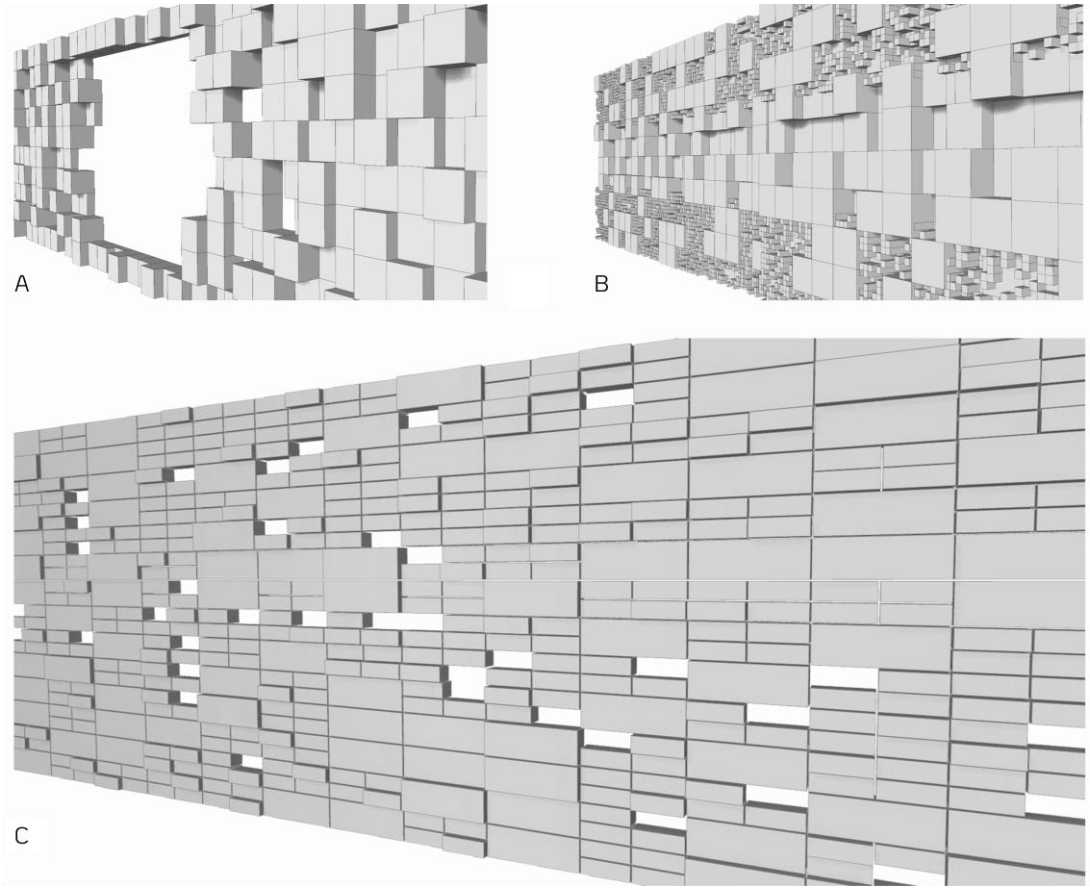


Figure 8.20. Two of the design ideas initially explored (A and B) and the final solution (C).

Thanks to the use of AD, two days were sufficient for us to explore multiple design solutions of diverse geometries and materials, which we then presented to the leading architects to receive feedback about possible design improvements. The first suggestion was to merge some features of the two options illustrated in Figure 8.20 (examples A and B), such as playing with the presence and absence of bricks

to create different permeability levels and using bricks of different sizes and positions to obtain the desired geometric randomness. The second suggestion was to reduce the degree of variation of the facade pattern to achieve higher control over its feasibility and final cost, decreasing the range of possible protrusions and brick sizes to only two. The last suggestion was to limit the creation of facade voids to the absence of small bricks and never of large bricks. The result was a design solution that met the initial design intent but used more restricted design variables (example C): the placement of either one large brick or four small bricks is randomly controlled, as also is the creation of both facade voids and protrusions.

8.3.2. Design Improvement

The next stage addressed the generation of a more detailed facade model that matched the building's dimensions and had different permeability levels according to both privacy and daylight requirements. Figure 8.21 illustrates some of the small changes applied to the design at this stage, which included (1) the balance between the placement of a large brick and a set of four small bricks, (2) the percentage of protruded bricks, and (3) the percentage of absent bricks.

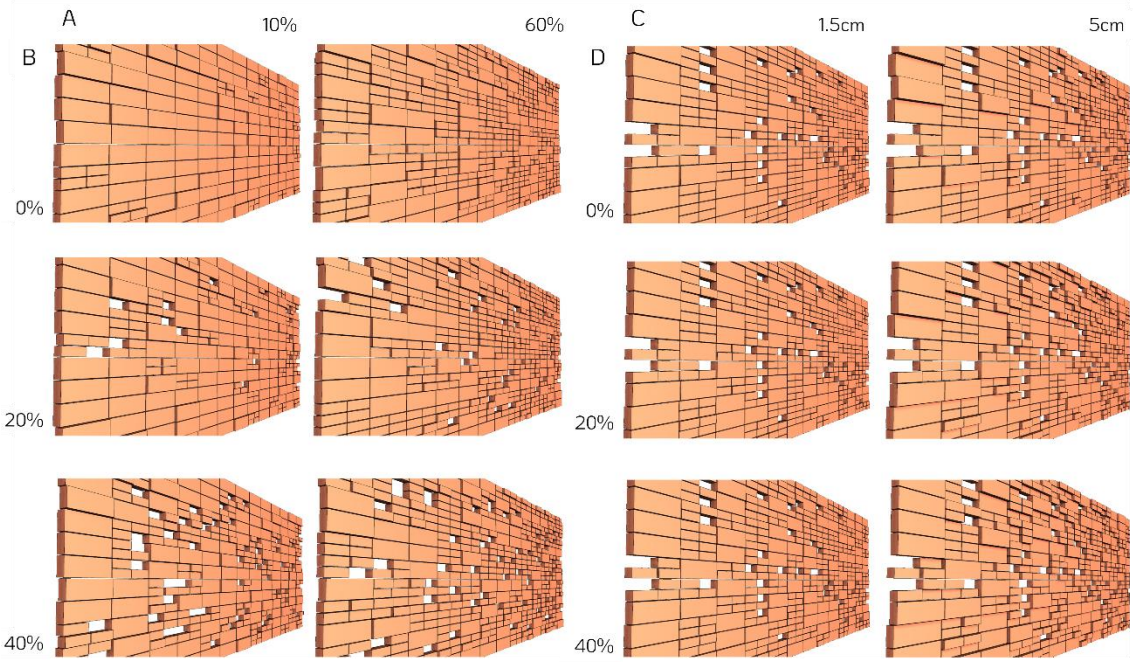


Figure 8.21. Design variations explored: A - Percentage of four small bricks; B - Percentage of absent bricks; C - Protrusions' depth size; D - Percentage of protruded bricks.

8.3.3. Design Rationalization

The last stage addressed the construction feasibility of the developed solution. As the budget for this project was relatively small, and the availability of advanced manufacturing processes was limited, it was important to adopt a simple manufacturing strategy that addressed the challenges resulting from the

facade design complexity. The bricks used in this project had a non-conventional size, which meant they had to be prefabricated. To reduce the solution's manufacturing cost, the design team had already limited the number of brick sizes to only two. However, there were still other challenges to be solved, namely the placement of the bricks.

The first limitation was the depth of the bricks, which was too narrow to obtain the necessary stability when stacked. We overcame this situation by simply doubling the bricks' depth. The second limitation was the presence of small bricks immediately above a facade void, which meant these bricks would have no support. We could have solved this by adding another constraint to the algorithm dictating that only large bricks could be placed on top of facade voids. However, this would decrease the visual intricacy of the pattern and so the design team adopted another strategy: fabricating the sets of three and four small bricks as single units.

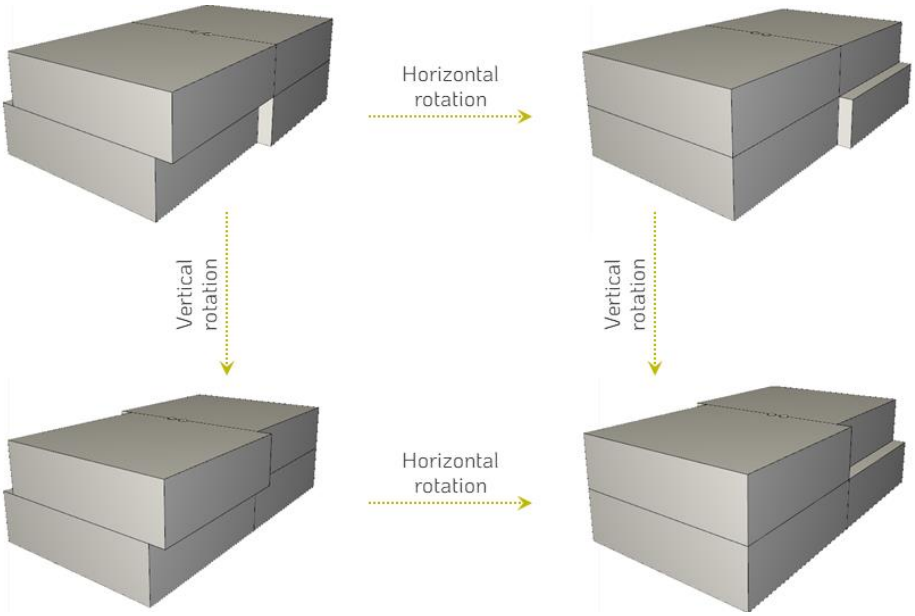


Figure 8.22. Possible rotations of a brick typology.

This solution, however, raised another challenge, namely the number of molds needed to produce the range of possible configurations. As the fabrication cost of the molds usually exceeds that of the elements produced [170], the design team searched for a strategy that minimized the number of customized molds needed, while preserving the solution geometric complexity. The strategy found relied on the fact that the back face of a set of small bricks had the inverted pattern of the front face, meaning that we could horizontally and vertically rotate it to obtain different pattern configurations (Figure 8.22) and thus reduce the number of produced molds to one-fourth of its original value. Figure 8.23 shows some of the brick configurations created together with their possible variations.

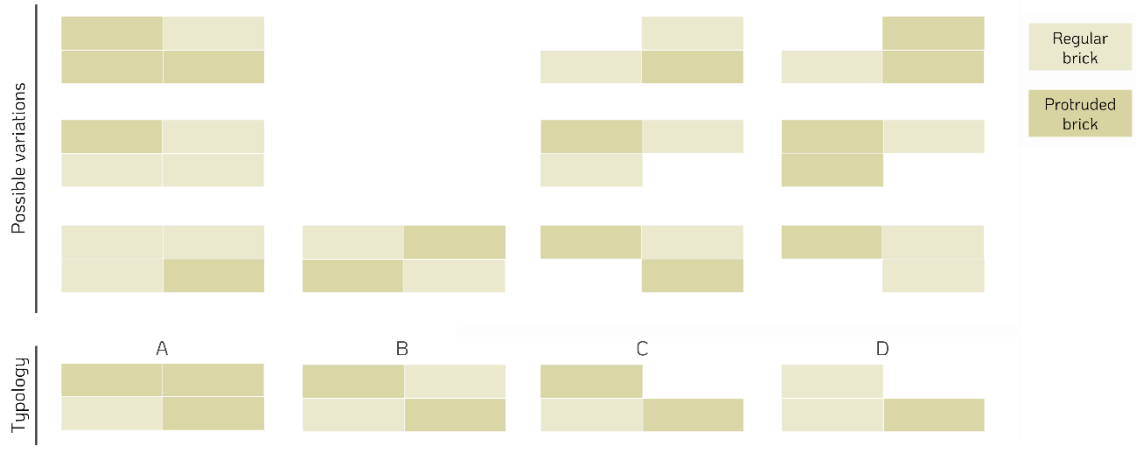


Figure 8.23. Four typologies (A to D) with the corresponding variations resulting from their rotation.

Another limitation was the placement of the different brick configurations on-site. This task promised to be challenging given the geometric complexity of the facade pattern and the need to accurately stack the bricks on-site to achieve the desired result. Fortunately, using the framework, we could easily obtain the list of positions of the different brick typologies on the facade, while graphically displaying them (Figure 8.24), facilitating their identification and accurate placement on site.

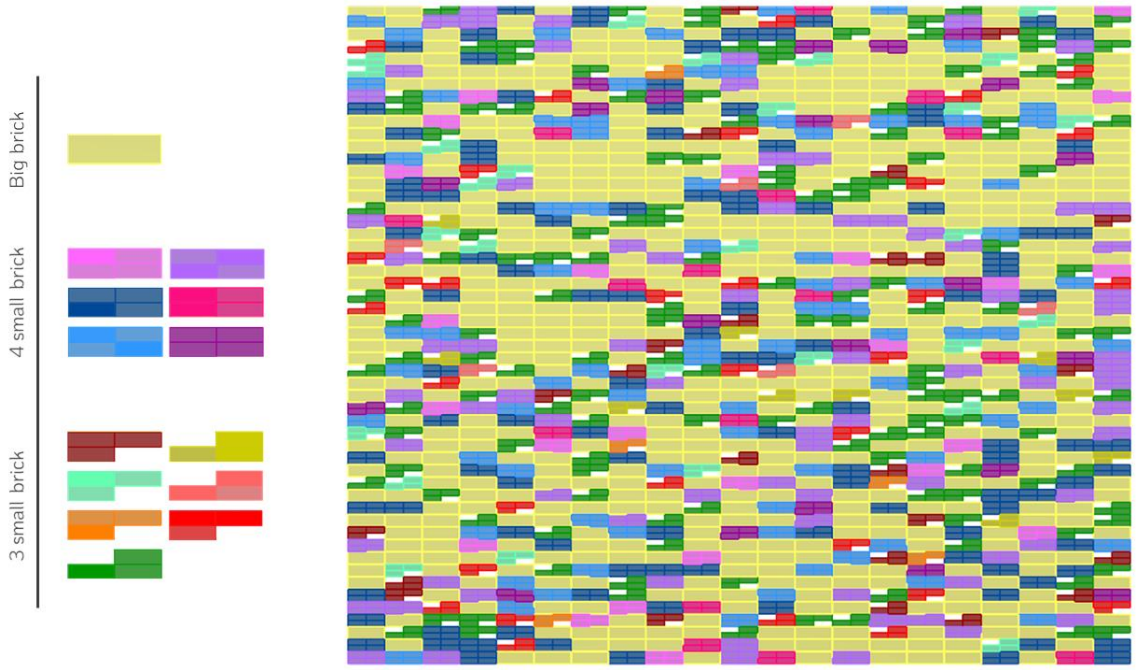


Figure 8.24. Graphical representation of the different brick typologies.

Regarding the bricks' support on-site, the design team opted for a hybrid strategy using metal profiles, to keep the verticality of the stacked bricks, and angle brackets, to fix them to the facade. Therefore, we used the framework to further detail the bricks for construction and develop the fixing elements, while balancing the solution's structural stability, natural ventilation, and material waste. We extended

the algorithm producing each brick to create two small grooves on its top and bottom surfaces to then place the metal profiles, allowing the placement of the bricks in two different positions: when the brick is protruded, the below and above metal profiles fit the first grooves; otherwise, they fit the second ones (Figure 8.25-A). Then, to give structural stability to the bricks' supporting system, we placed two metal profiles on all bricks, except the top and bottom row ones, and distributed angle brackets holding alternating rows of bricks (Figure 8.25-B). As is visible in Figure 8.25-C, the resulting solution had the advantage of creating a thermal isolation zone between the bricks' structure and the building's exterior wall that improved both its natural ventilation and thermal performance.

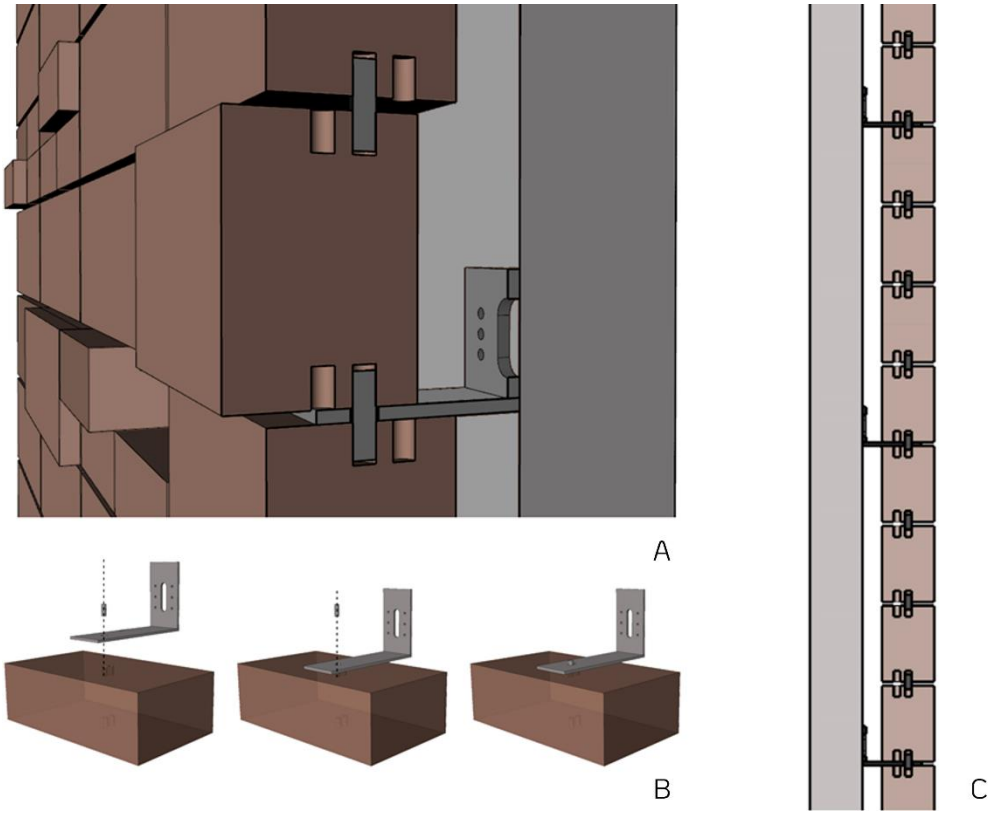


Figure 8.25. Bricks supporting system: A. facade section illustrating the two possible stacking positions; B. angled brackets and metal profiles fitting process; C. facade section showing the thermal isolation zone.

Lastly, we used the framework to estimate the solution's cost and extract the list of quantities of the facade elements (Figure 8.26). In the end, the design team was provided with the 3D model of all facade elements, including the bricks' molds, metal profiles, and angle brackets, and their corresponding quantities, as well as the design's construction information, such as the different elements' positions, material, and dimensions. This information would have allowed the design studios to proceed with the fabrication and assembly of the different facade elements if they had won the competition. Moreover, the adopted methodology would have also facilitated the response to the constraints imposed by both manufacturing and construction processes, allowing the design to be constantly updated with further details and information.

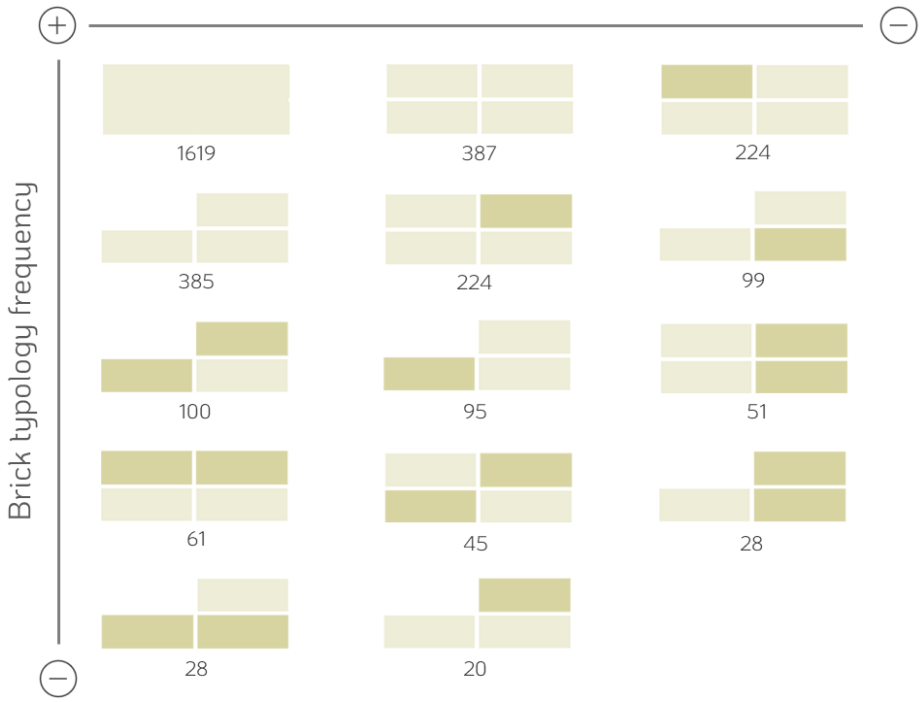


Figure 8.26. All brick types and frequency of use.

8.3.4. Discussion

The application study demonstrated how the application of the framework in a non-AD context allowed for a more flexible design process, resulting in a dynamic design workflow with direct participation and feedback from both architects with and without AD skills. It is also noteworthy that the design resulting from this collaboration is the kind of solution that is quite difficult to achieve manually, especially with the existing time constraint.

It also evidenced the framework's ability to enhance creative design processes and gradually evolve a non-conventional, complex facade design responding to multiple requirements in a short period of time (six days): as it allowed the design team to quickly visualize the impact of the suggestions made, the search for solutions that met both creative intents and design constraints was easier. Throughout this process, the design team could make design changes according to the needs of each stage: e.g., applying more drastic changes at initial stages, which was critical to narrow the design space being explored, and more subtle changes at later stages, which was important for the step-by-step improvement process.

The application study also proved the framework's capability to detail the solutions with algorithmically developed construction elements, in this case, metal profiles and angle brackets, while producing the corresponding construction information, including dimensions, locations, and material quantities. Given its design complexity, these processes would not have been trivial if we had not used AD, due to being often difficult to generate and get this type of information about nonstandard elements.

Lastly, it also demonstrated how the framework's principles can easily be applied to different design tools, making it possible to alternate between tools depending on the momentary needs for speed or detail and generate equivalent model with no additional effort. In this case, it allowed us to benefit from the higher performance of AutoCAD at initial stages and only later transit to Revit to obtain more detailed and heavyweight 3D model. Figures 8.27 presents the resulting AD workflow and Figure 8.28 two rendered views of the final model in Revit.

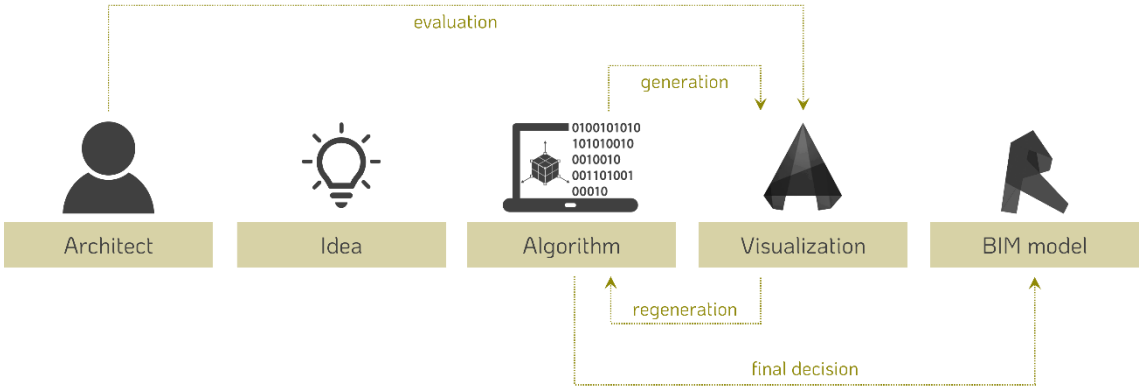


Figure 8.27 AD workflow: the architect presents the design idea to be implemented and, after visualizing the results in the CAD tool, suggests design improvements; this process is repeated until the architect is satisfied with the result, at which point we use the same algorithm to generate its BIM model with the correct BIM families and information.

Figure 8.28. Rendered views of the final model. Left: view from the main street where two of the facades developed are visible; Right: view of the building inside the courtyard with the third AD facade.

In short, we believe that a non-AD approach would not reach the same solution because, firstly, the design exploration process would have been much more tiresome and time-consuming, leaving less time available to test other design options; secondly, the obtained model would hardly allow the iterative incorporation of design changes, hindering the design's iterative improvement; and, lastly, the transition between different design tools would be time consuming and laborious, making it difficult to benefit from their different advantages in the different stages of the design process.

8.4. STUDY 4

This application study is the result of a collaboration with the Portuguese design studio Atelier dos Remédios, which focused on the development of an alternative facade design solution for the residential building of Study 3 made of geometrically different *cobogó* bricks [381]. The choice for this type of hollow brick was due to its air circulation and light penetration control properties and the aim to integrate some local culture in the facade design [382]. Nevertheless, the architects wanted to avoid the use of standardized *cobogó* bricks and take advantage of their different geometric variations to simultaneously improve the building's daylight and ventilation performance and search for an architectural identity. The result of this collaboration was a three-stage process where the framework was used, first, to implement the design intent and explore variations of it; second, to improve the design regarding different criteria; and lastly, rationalize and fabricate the resulting solution.

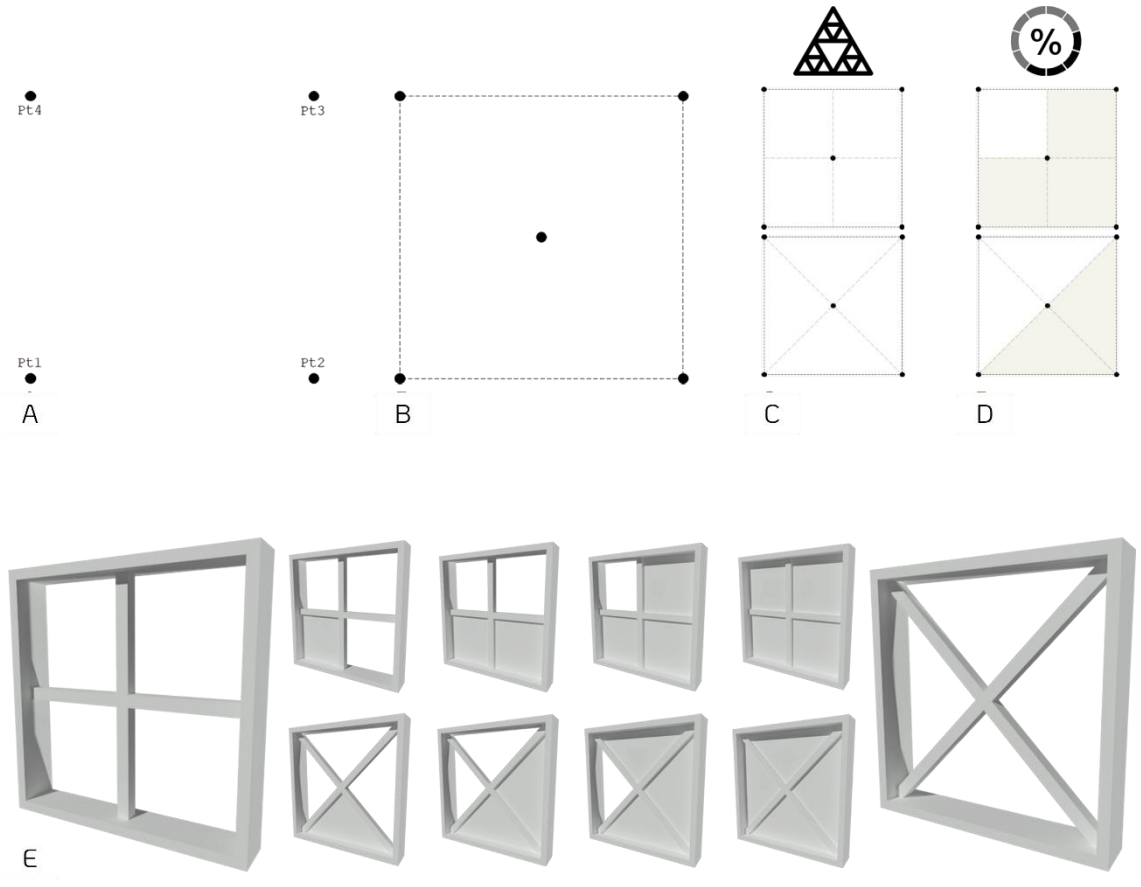


Figure 8.29. Top. the implemented geometric rule: when receiving a set of four points defining the brick's vertices (A), it creates the brick frame and calculates its center (B), while allowing for the selection between two possible rules (C) and different opacity levels (D). Bottom. the range of possible solutions for a given set of four points (E).

8.4.1. DESIGN EXPLORATION

The first stage involved the implementation of the design intent and the geometric exploration of the facade design. It started with the definition of a simple geometric rule that, by varying its parameters,

generated bricks of different shapes and opacity levels (Figure 8.29). Then, to create different facade design solutions, we combined the previous algorithm with those producing different surface grids, obtaining a set of *cobogó* bricks whose shape perfectly fitted the selected grids (Figure 8.30).

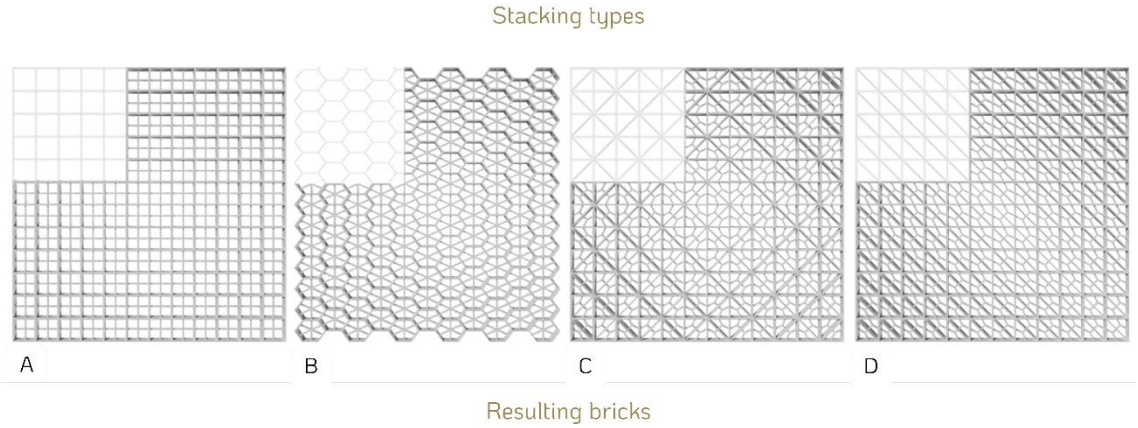


Figure 8.30. Four stacking options tested: the selected grid configuration (top-left grids) and the resulting bricks perfectly adapted to it.

After deciding on the stacking option(s) to explore, we applied gradual geometric variations to the facade design to better respond to the initial design intent, namely (1) using both geometric rules (Figure 8.29-C); (2) exploring different opacities (Figure 8.29-D); and (3) combining two types of stacking (Figure 8.30, A and D), as illustrated in Figure 8.31. At this stage, we could have generated several design solutions to then present, all at once, to the design team. However, given the potentially large set of solutions we could explore in a short amount of time, their presentation would be quite a challenging task that would hinder the design decision-making process. Instead, we opted to gradually explore the design space according to the guidance of the design team, i.e., by iteratively presenting design solutions and using the feedback received to generate new ones until the team was satisfied with the results. In the end, despite not visualizing the entire design space, it was the team that controlled the navigation process through the set of potential solutions and chose the one that most pleased them.

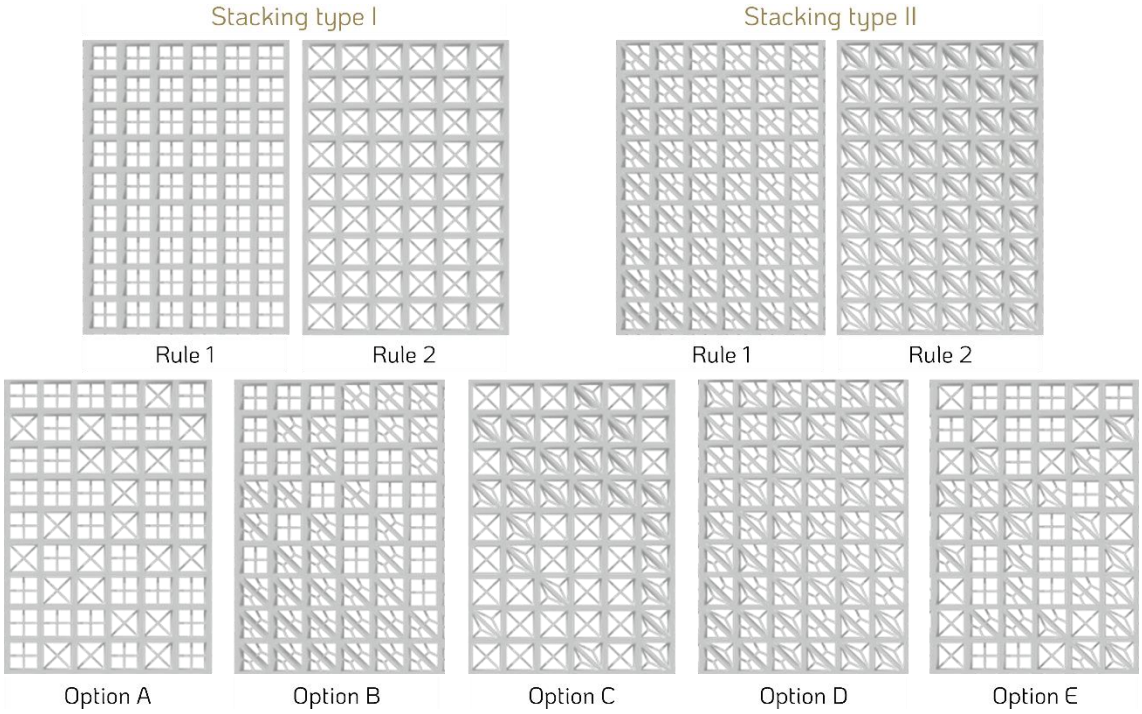


Figure 8.31. Pattern evolution: On top, two stacking types using rule 1 and rule 2; on the bottom, different options explored using both rules and the stacking type I (A), rule 1 and both stacking types (B), rule 2 and both stacking types (C), both rules and the stacking type II (D), and all rules and stacking types (E).

8.4.2. Design Improvement

The following stage addressed the improvement of the application study regarding its privacy, daylight, and natural ventilation levels. To that end, we extended the solution with a set of algorithms that allowed us to control its geometric characteristics and meet the following criteria established by the design team:

- The fraction and size of the voids should be smaller in both private and circulation areas, suggesting a greater use of triangular bricks than squared ones in these areas.
- Living rooms, terraces, and courtyards should have clearer views and greater daylight illumination, meaning that squared and less opaque bricks should prevail in these areas.
- There should be a preference for more opaque bricks in areas exposed to direct sunlight.
- There should be higher opacity levels in the central area of the facade than at its top or base to promote natural ventilation.

These requirements, which were entirely based on the architects' own experience, originated a set of values matching the desired daylight, privacy, and natural ventilation levels for each interior space function (Figure 8.32) that, when assigned to the design variables, created different ratios of squared to triangular elements, as illustrated in Figure 8.33. By assigning this information to our algorithm, we ensured the obtained solutions respected the existing requirements and thus were always within the

range of acceptable designs. This enabled the design team to conscientiously explore the design space and apply iterative design changes that balanced the solutions' performance with the design intent.

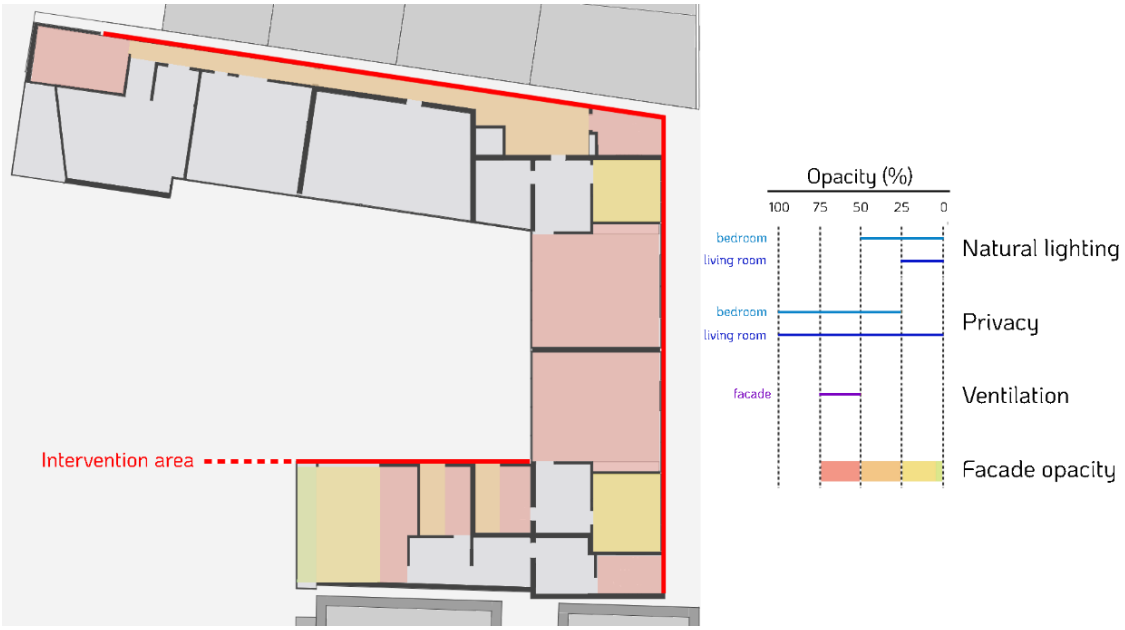


Figure 8.32. On the left: application study plan with red lines representing the facade intervention and colored areas identifying each inside space opacity level; on the right: legend of the different opacity levels with the corresponding daylight, privacy, and ventilation levels.

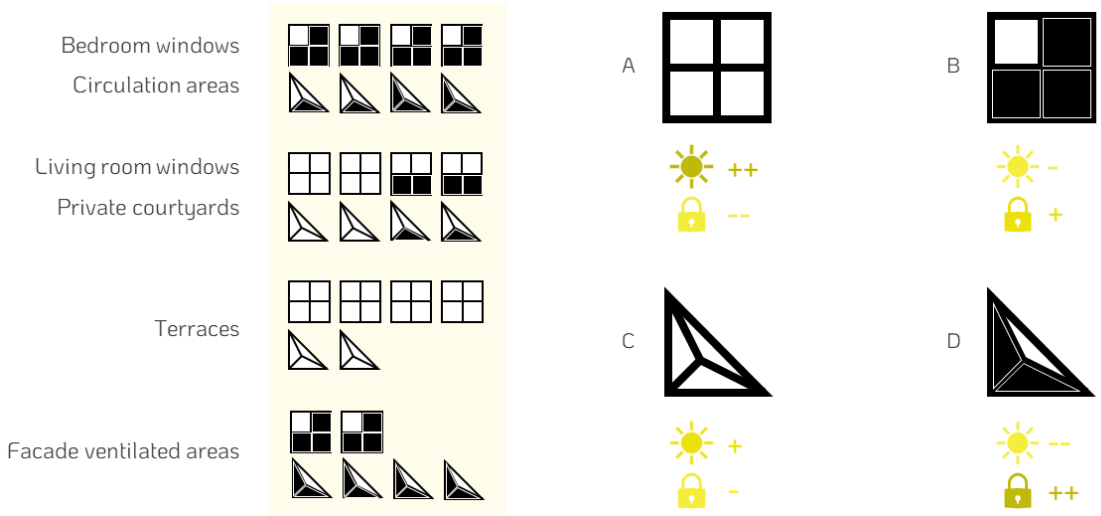


Figure 8.33. Left: the ratios of squared- to triangular-shaped elements of each interior space function; Right: each type of brick privacy (lock symbol) and daylight illumination (sun symbol) levels, with type A corresponding to the highest daylight and lowest privacy levels, types B and C to intermediate levels, and type D corresponding to the opposite.

8.4.3. Design Rationalization

At this stage, we addressed the solution's feasibility in terms of cost, construction time, and resources, selecting the framework's algorithms to control the number and positioning of different facade elements, while searching for the most appropriate manufacturing strategy.

As the initial aim was to manufacture ceramic *cobogó* elements whose unconventional shapes required the production of customized molds, we used the framework's algorithms to automatically obtain the latter's 3D models, together with their frequency of use and list of positions on the facade (Figure 8.34). Based on this information, the design team evaluated the feasibility of using ceramic facade elements, concluding that their manufacturing costs and waste would not be viable since they required the production of eighteen customized molds (Figure 8.34 bottom) and their massiveness and fragility would not suit the need for facade areas that acted as movable window shutters.

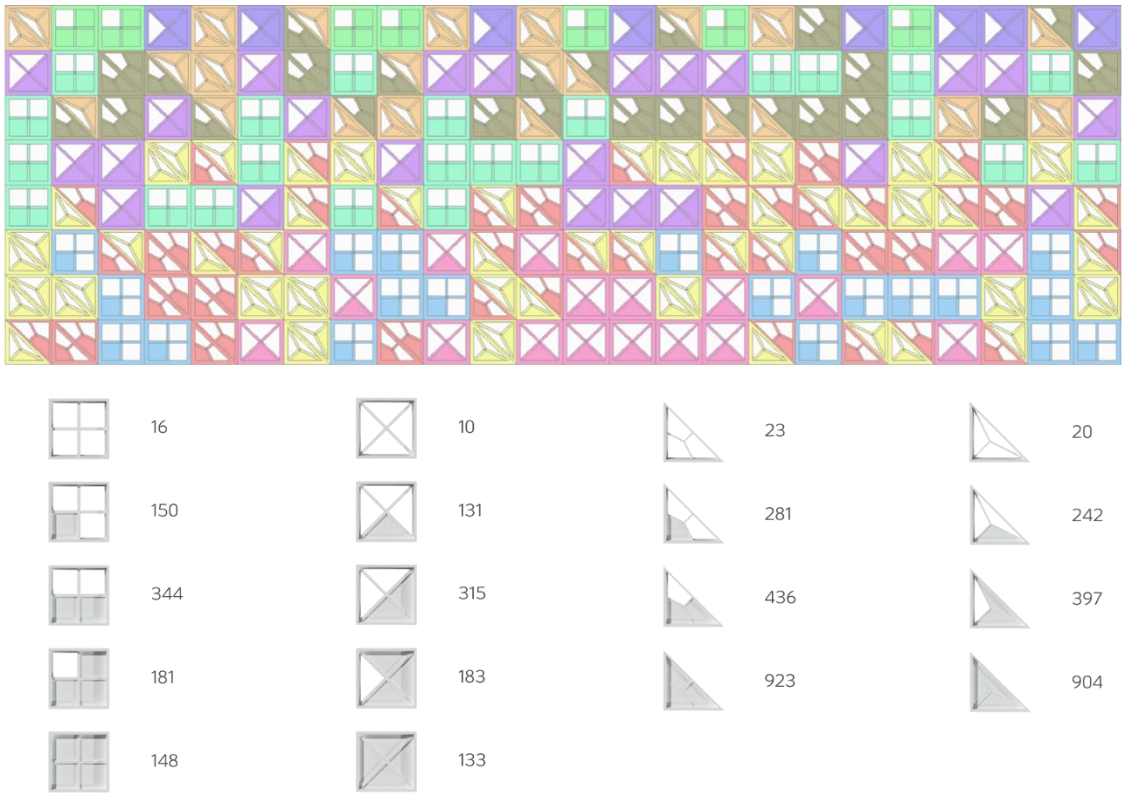


Figure 8.34. Part of the facade 3D model with each type of brick identified with a different color (top) and their frequency of use (bottom).

Given the previous problems, the team selected a different material and fabrication strategy for the facade elements, opting to produce lightweight wood panels that allowed to (1) maintain the geometric diversity and plastic expression of the original solution, (2) eliminate the need for mold production; and (3) obtain facade elements whose weight and robustness were adequate to act as movable window shutters.

To obtain the desired geometric configuration and visual expression, the team decided to manufacture the panels in layers that, when overlapped, originated different thicknesses and shapes (Figure 8.35). During this process, we used the framework to, first, generate wood panels with different geometric compositions and opacity levels and, then, automatically produce the information needed

for their manufacturing. Figures 8.36-7.37 present some of the prototypes developed during this stage and two rendered views of the final solution, respectively.

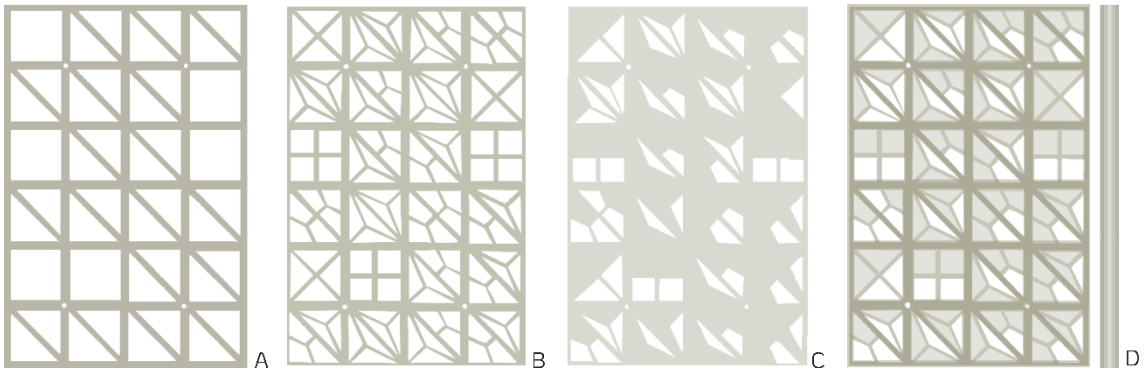


Figure 8.35. A. the outer layers of a wood panel delineating the *cobogó* elements' frame shape; B. its middle layers outlining the *cobogó* elements' interior elements; C. its inner layer producing both the existing voids and opaque areas; D. the resulting wood panel after overlapping the previous layers in a A-B-C-B-A order (front and side views).

8.4.4. Discussion

This application study demonstrated the ability of the framework to support collaborative design processes merging a team of architects with AD skills and a traditional design studio: while the former applied the framework throughout the project, first to geometrically develop the facade elements and then to manufacture them, the latter were responsible for guiding the entire process, proving its adaptability to multiple design scenarios, while also highlighting the growing need of design teams to integrate architects with different backgrounds. It also assessed the framework's ability to (1) convert conceptual design intents into mathematical principles, (2) generate wide design spaces resulting from the same principles, and (3) balance creative intents and functional requirements.

Moreover, by comparing the AD workflow with a non-AD one, we also concluded that it:

- Made it easier to achieve higher levels of design complexity, due to integrating different types of information and constraints in a single workflow.
- Augmented the control over the geometric variations tested, due to providing constant feedback on their success.
- Supported the generation of a wider range of *cobogó* elements, due to requiring less time and effort to explore the design space and facilitating the application of small-to-large design changes.
- Increased the likelihood of achieving better-performing solutions, due to guiding the design exploration process towards the existing requirements.
- Facilitated the balance between the solutions' visual composition, performance, and feasibility, due to providing information about their associated costs and resources.

- Smoothed the transition between design exploration and fabrication stages, due to automating the extraction of manufacturing and construction information.

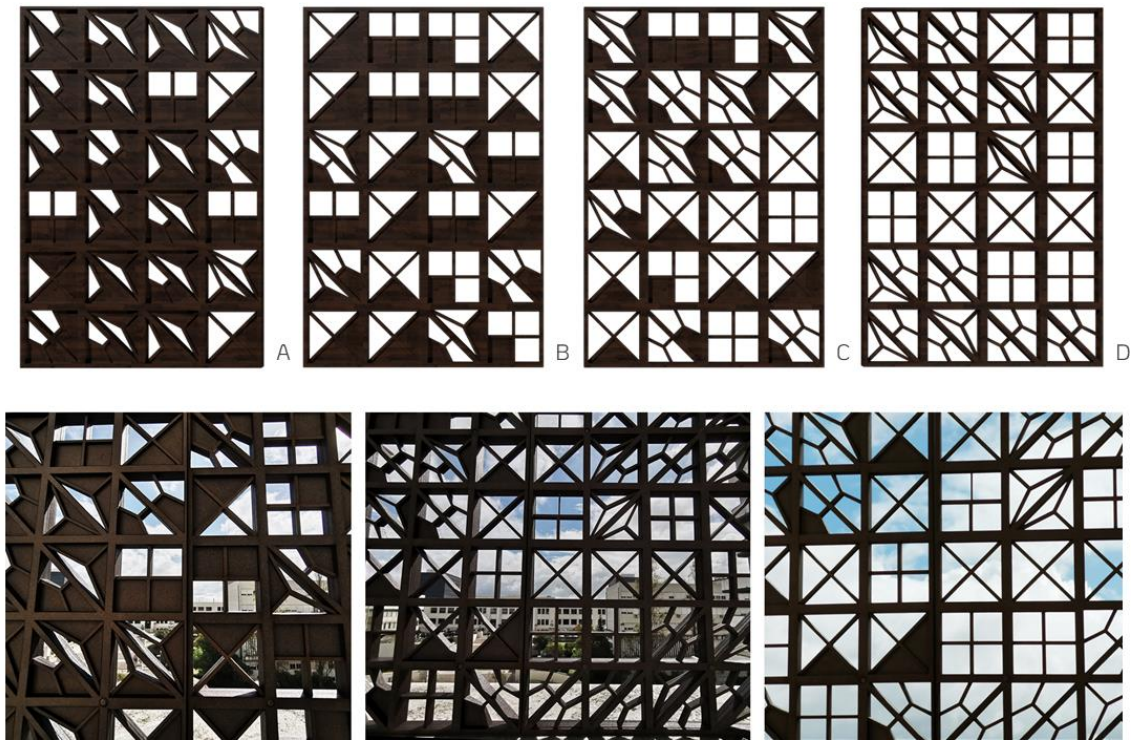


Figure 8.36. Prototypes with different opacities: on top, four typologies with decreasing opacity levels (A to D); at the bottom, three pictures of the produced prototypes.

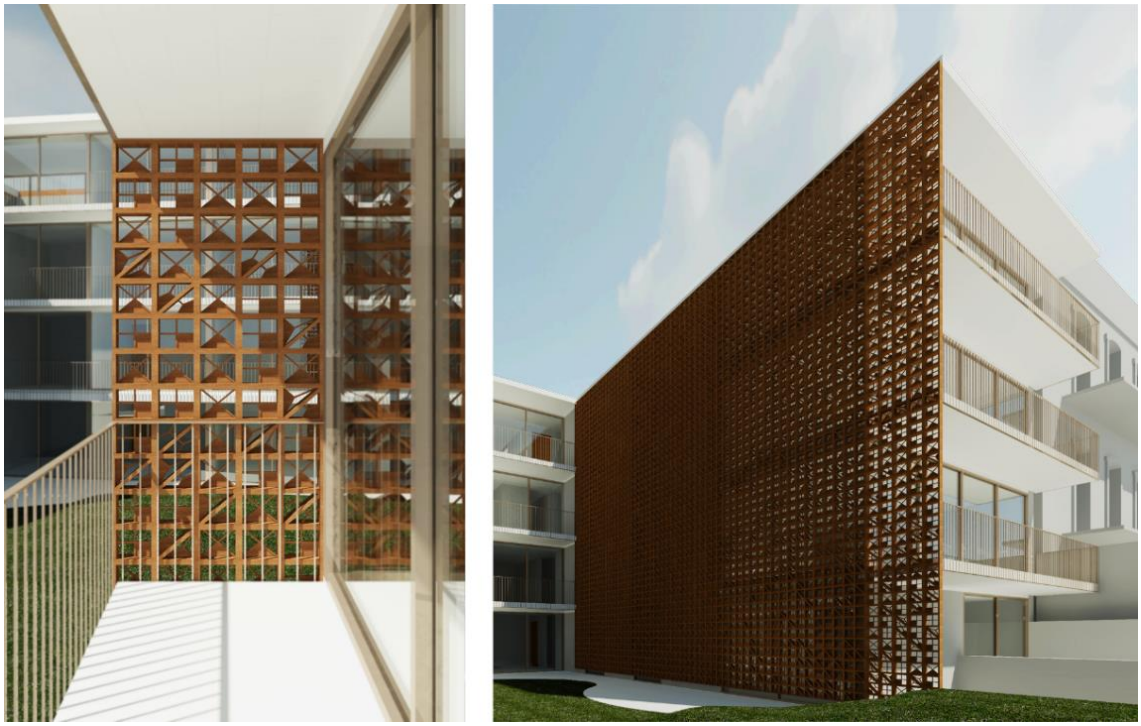


Figure 8.37. Two rendered views of the final facade design in Revit.

In sum, the important lessons drawn from this application study are that the design exploration stage benefits from AD when the design (1) can be geometrically parameterized, making it possible to explore a wider range of solutions; (2) includes both aesthetic and performance requirements, as the initial investment required is recovered in the iterative evaluation and balancing of the different requirements; and (3) is geometrically complex and/or its manufacture is likely to benefit from rationalization processes, allowing the design feasibility to be increased while maintaining its design identity.

8.5. STUDY 5

The last application study demonstrates the ability of the framework to support facade design processes considering multiple construction schemes and their impact on the solutions' visual expression, while facilitating the solutions' manufacturing through different fabrication strategies. This application study results from an attempt to collaborate with the manufacturing industry in order to produce a full-scale model of an unconventional facade shading panel, whose different element shapes, construction details, and surface finishings originated different outcomes and technical documentation. The aim was to understand the challenges and limitations arising from this collaboration.

8.5.1. DESIGN EXPLORATION

The first stage entailed the implementation of (1) the design intent – creating a nonstandard facade design with an irregular triangular configuration that could produce a dynamic three-dimensional visual effect; and (2) the existing performance requirements – responding to different functional, privacy, and daylight needs.

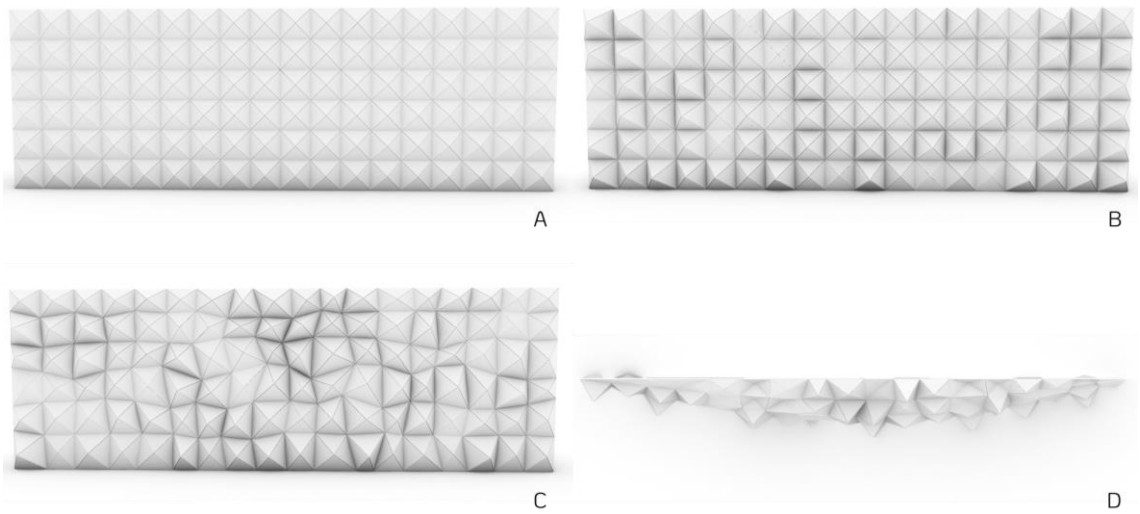


Figure 8.38. Design evolution: creation of squared-based pyramidal elements with (A) equal and (B) random heights, followed by the (C) random deformation of their bases and (D) their distribution on a convex surface together with their heights varying in both directions.

To address the first requirements, we developed a solution based on quadrangular pyramids (Figure 8.38-A), due to their lateral faces creating the desired triangular configuration and their volumetry producing the intended three-dimensional effect. To obtain the desired geometric irregularity and randomness, we made both their height and base shape vary according to different random factors (Figure 8.38, B-C). Finally, to achieve the intended three-dimensional effect, we increased the surface convexity on which these elements were distributed, while making their height vary in both inside and outside directions (Figure 8.38-D).

To satisfy the remaining requirements, we decided to create holes on the pyramids' triangular faces to obtain different levels of permeability (Figure 8.39-A), while randomly controlling their size to meet the desired irregularity (Figure 8.39-B). However, given the specific functional, privacy, and shading requirements of each facade area, it was difficult to ensure the different levels of permeability needed were achieved by simply using a randomly varied factor. Therefore, we increased control over the random factor, thus facilitating the manipulation of the holes' size according to the existing requirements (Figure 8.39-C).

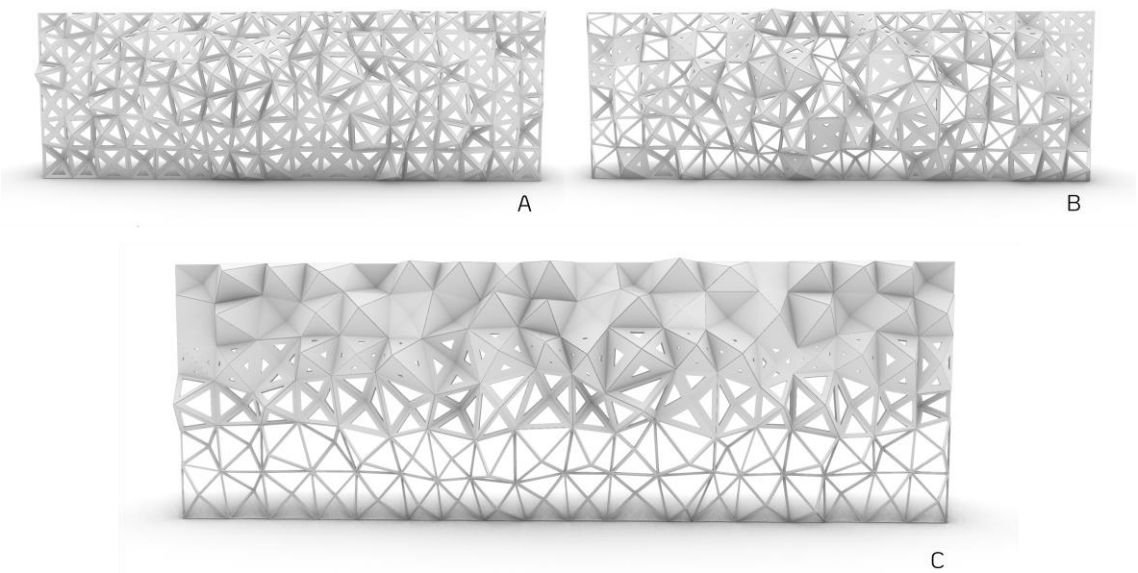


Figure 8.39. Performance-related design variables: obtaining different permeability levels through the creation of panel holes with (A) fixed, (B) random, and (C) gradually changing sizes.

By iteratively assigning different values to the design's parameters, we could easily and quickly test multiple design variations until it successfully balanced the design intent and the existing requirements. Figure 8.40 illustrates this process with four design variations resulting from the same algorithmic description but responding to different functional needs: Example A, for instance, is an attempt to smooth the visual effect of Figure 8.39-C; Example B further accentuates the previous effect by making the holes' size gradually decrease also in the horizontal direction; Example C explores the creation of multiple radial permeable areas; and Example D creates a central permeable area.

8.5.2. MANUFACTURING-RELATED INFORMATION

After the team decided on a solution (Figure 8.40-A) and its material (metal), we addressed its feasibility and preparation for fabrication. To facilitate its manufacturing and assembly on-site, we first focused on discretizing the resulting facade tiling without neglecting its structural stability. Considering its geometric irregularity, we decided to fabricate the triangular panels individually, using the framework to extend them with folded ends that ensured their subsequent connection and fixation into a larger

sound structure. Given their algorithmic nature, the construction extensions were automatically adapted to the panels' ever-changing geometry, while their shape, width, and thicknesses could also be easily and almost instantaneously manipulated by us (Figure 8.41).

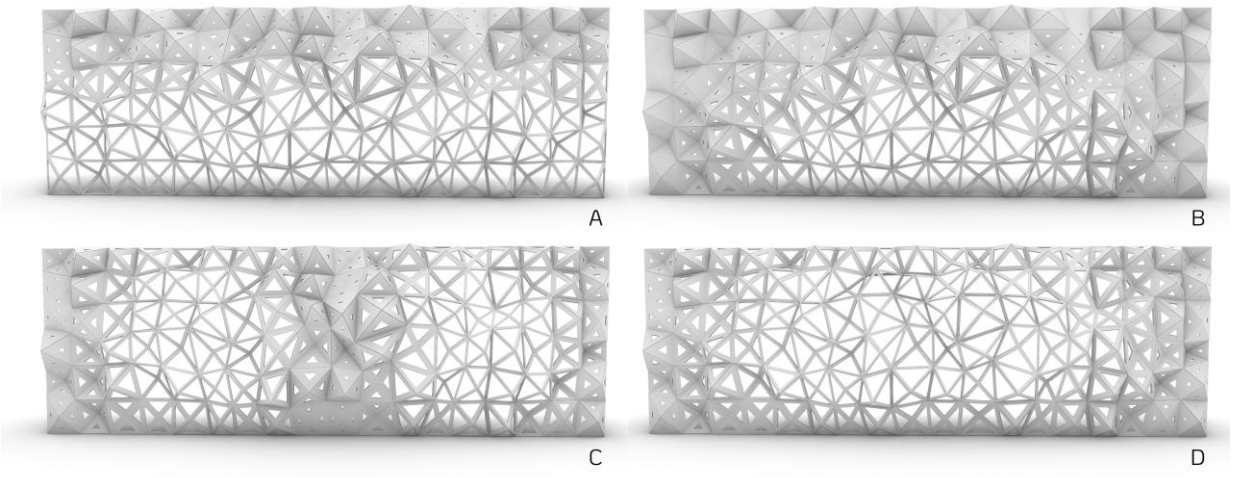


Figure 8.40. Design iterations resulting from different apertures factors.

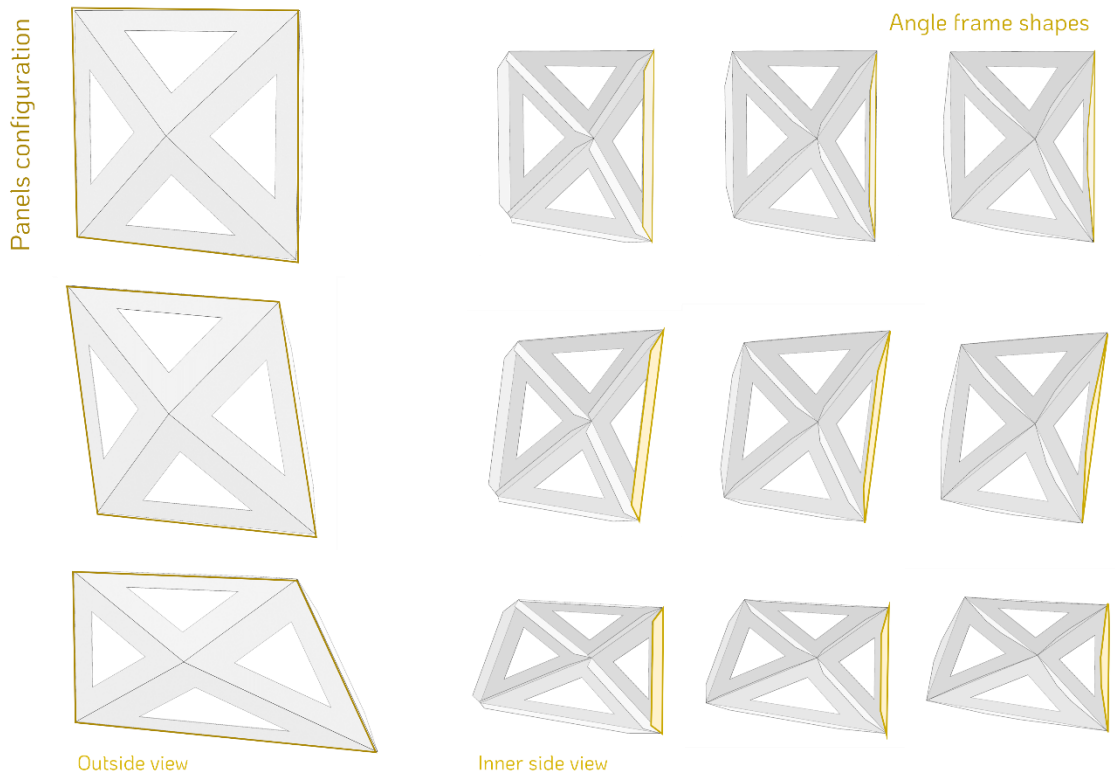


Figure 8.41. Construction details: on the left, two panel configurations and, on the right, their extension with folded ends of varying sizes and shapes.

To obtain smaller, self-supporting parts that could be easily transported and assembled on-site, we then used the framework to divide the solution into three facade modules of 3x3 meters, each composed of 36 quad-based pyramids (or 144 triangular panels) of different sizes and shapes (Figure

8.42-A) and create an outer metal frame grouping them (Figure 8.42-B). Besides enhancing their structural stability, this solution also enabled (1) the subsequent assembly of the different 144 panels at the factory, which was critical for achieving higher levels of production quality and accuracy, and (2) the use of identical linear elements joining the panels and punctual fixing points attaching them to the building structure. It, however, slightly reduced the geometric irregularity of the original solution due to forcing straight alignments to exist at every 3 meters. Nevertheless, considering the resulting manufacturing and construction gains, the team considered this to be acceptable.

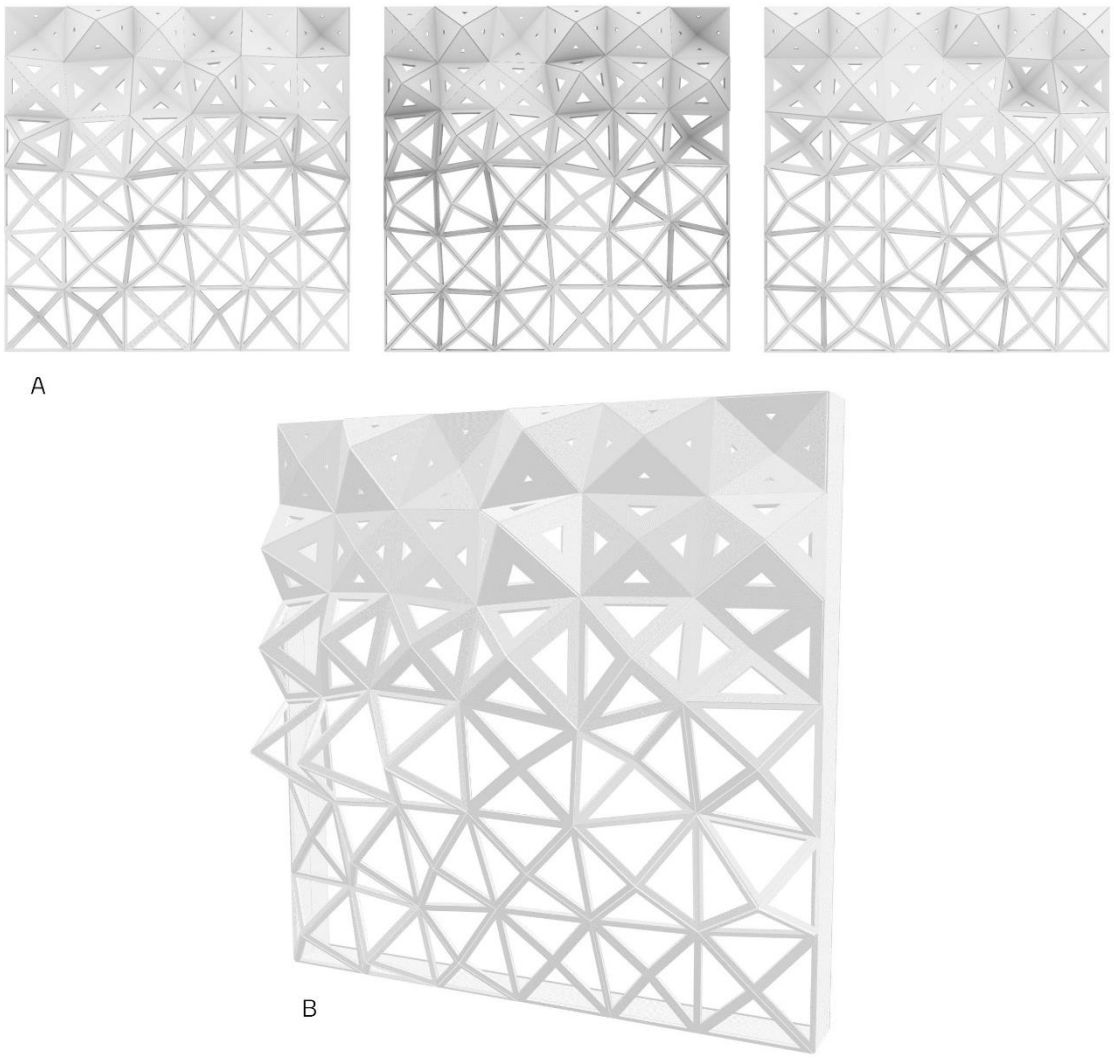


Figure 8.42. The division of the solution into equally sized parts (A) and the creation of an outer frame increasing their structural stability and facilitating their fixation to the building structure (B).

8.5.3. DESIGN PROTOTYPING

Before proceeding with the manufacturing of the final solution, it was important to first test its feasibility on a smaller scale. To that end, the following stage encompassed the technical documentation and prototyping of a single facade module.

Considering both its geometric characteristics and materiality, we selected laser cutting as the manufacturing strategy. Then, we used the framework's fabrication algorithms to:

- Produce the technical drawings required by this strategy, obtaining a set of 144 scaled drawings with the panels' unfolded plans and with different line types identifying the edges to cut or fold (Figure 8.43).
- Identify the different panels according to their installation sequence (Figure 8.44-A), simplifying their challenging and time-consuming assembly process, while minimizing the occurrence of potential errors, we used the existing algorithms.
- Reduce the material waste resulting from their manufacturing, distributing the panels' unfolded representations in the best way by minimizing the gaps between them (Figure 8.44-B).

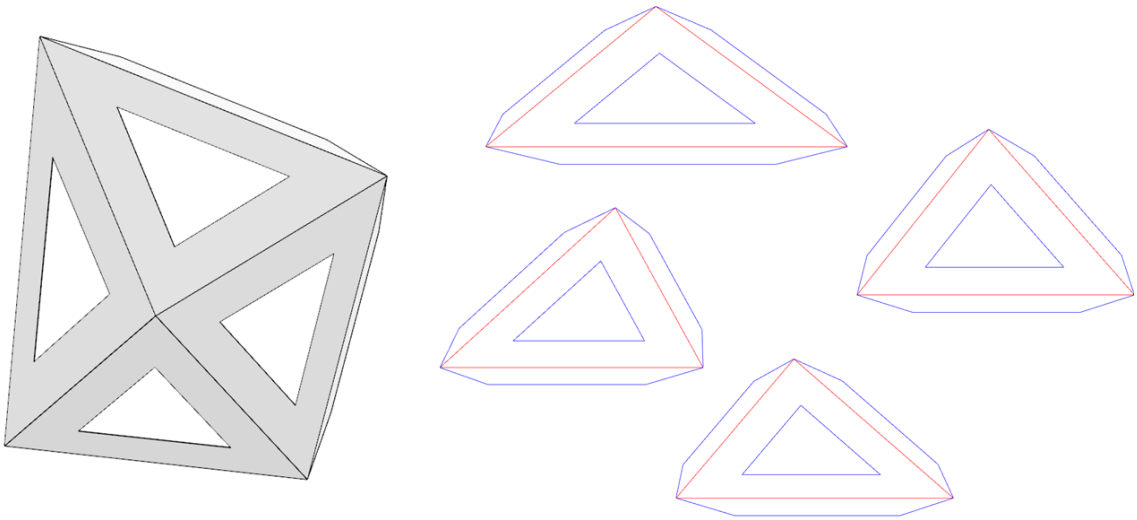


Figure 8.43. Pyramidal element 3D model (left) and corresponding unfolded technical drawings (right).

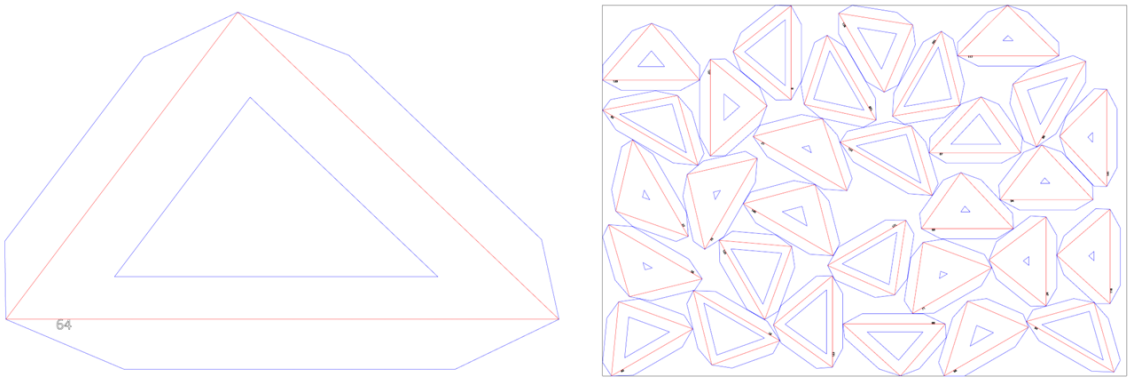


Figure 8.44. A. Triangular panel labelling; B. Nesting of the triangular panels' unfolded drawings.

Finally, to produce the 144 scaled panels, we send the resulting drawings to the laser cutter, which we then fold and assemble according to the labelling instructions (Figure 8.45).

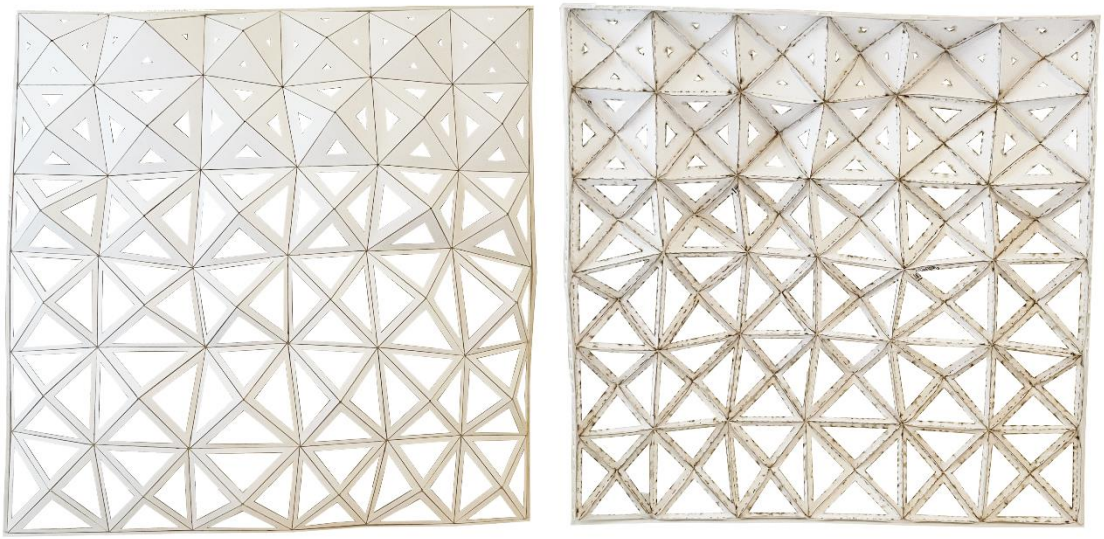


Figure 8.45. Facade panel scaled prototype: outside (left) and inside (right) views.

8.5.4. Aesthetical Consideration

Producing the prototype was important to improve our perception of the solution's outcome, allowing us to realize that, although it satisfied the design intent from an outside perspective, the same did not happen from an inside perspective. This happened because, first, the fluidity of the triangular tiling was broken by the panels' folds and, second, while all the other facade elements had irregular geometric characteristics, these elements did not, thus deviating from the design intent (Figure 8.46-A). Given the flexibility of our proposal, we could quickly and easily test other construction schemes for our solution, while assessing their aesthetic impact. During this process, we could also effortlessly produce the technical documentation needed to prototype other design alternatives.

Since connection elements like these were necessary, we focused on breaking their regularity by dynamically varying their widths, making them either directly or inversely proportional to the panels' opening size (Figure 8.46, examples B-C), while testing different shapes for them, such as rectangular (Figure 8.47-A) or trapezoidal (Figure 8.47-B). We also applied more drastic changes and either produced each pyramidal element as a single foldable panel, whose inner connections were hidden by U-profiles (Figure 8.47-C), or converted the originally two-dimensional panels into three-dimensional elements, whose inner volume adapted to both their thicknesses and opening sizes (Figure 8.47-D).

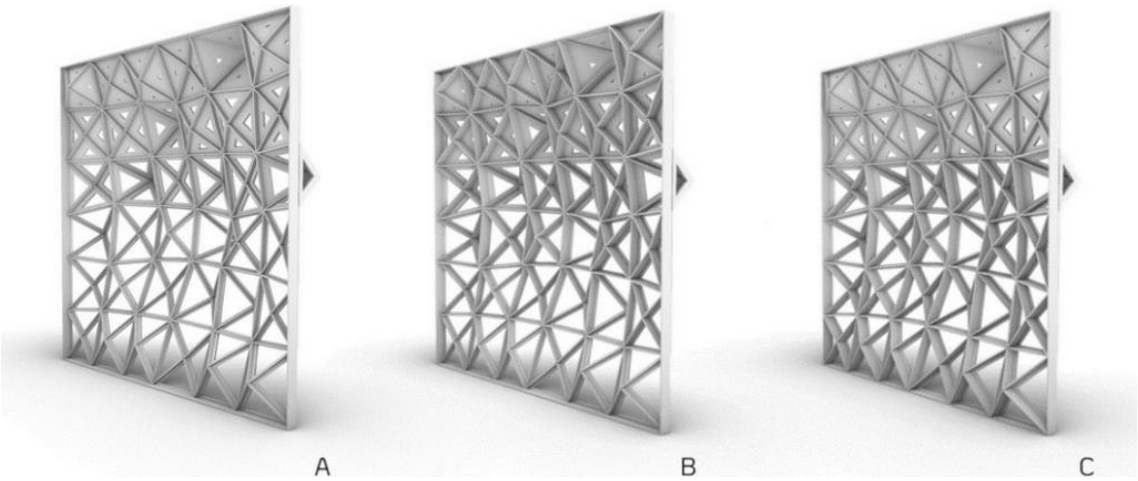


Figure 8.46. Prototype interior view: panels with folded ends with (A) fixed sizes or with their size (B) inversely proportional or (C) directly proportional to the panels' hole.

After deciding on the best solution, the team could proceed to the next stage and, first, produce another prototype to confirm its viability and then, manufacture the final solution. As, during this process, the design's technical documentation was constantly regenerated to accommodate the design changes made and the specificities of the selected manufacturing strategy, not only was it always up to date and accurate, but also the time and effort spent on its production was largely reduced. Lastly, in case we needed to complement the resulting drawings with additional context-specific information and details, such as screw holes, we could select the available manufacturing-related functionalities for that purpose or even combine the framework with external AD libraries when necessary.

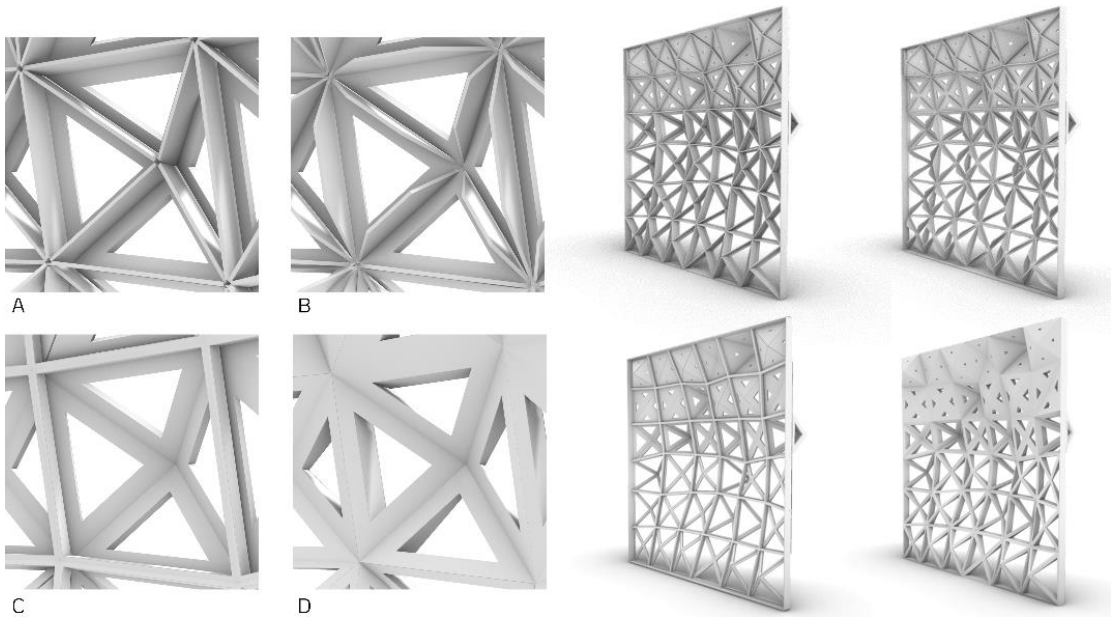


Figure 8.47. Algorithmic exploration of different construction schemes: A. triangular panels with rectangular folds; B. triangular panels with trapezoidal folds; C. pyramidal folded panels with U-profiles hiding their connections; D. three-dimensional triangular panels. On the left, their close-up view and, on the right, the resulting prototype interior view.

8.5.5. DISCUSSION

In a first stage, the framework facilitated the implementation of the design intent, making it easier to apply iterative design changes and integrate different design constraints and requirements, such as the desired geometric irregularity and the different shading and privacy needs, while reducing the time and effort spent with the programming task.

By constantly receiving immediate feedback, the process evolved in a conscious and informed way, increasing our perception of the results and the impact of our changes on them. This therefore allowed satisfactory results to be achieved more quickly, enabling us to spend more time and effort on creative tasks and on the exploration of a wider design space encompassing solutions beyond those initially considered. This also facilitated the balance between the design intent and other existing requirements, such as privacy, shading, and feasibility.

Regarding more advanced stages, the framework reduced the time and effort needed to produce the design's construction details, automatically generating panels with folded ends fitting the design's geometry and selected fabrication scheme. Given its algorithmic nature, we could also easily manipulate the size of the produced details according to context-specific structural and aesthetic requirements, while automatically producing the corresponding technical documentation for the selected manufacturing strategy, including the panels' unfolded drawings with the corresponding line types for cutting, folding, and engraving purposes and labels guiding their assembly process.

Moreover, besides facilitating the materialization of the solution, it also allowed us to minimize material waste by arranging the resulting documentation in an optimized way. Finally, it also smoothed the transition between creative and materialization processes by facilitating the testing of different construction schemes to assess their aesthetic impact, sparing us from the time-consuming and hardworking task of updating the resulting construction details and corresponding technical documentation, minimizing the accumulation of design errors.

9. DISCUSSION

This chapter assesses the suitability of the proposed methodology and framework for architectural practice and its ability to solve complex design problems. It also evaluates the capability of the proposal to integrate different design practices and problems, while improving the flexibility and efficiency of architectural design processes. The next sections elaborate on the findings of the previous chapter, while reflecting on the proposal's applicability and usefulness for practice-based design scenarios, as well as on the merits and limitations of the Algorithmic Design (AD) workflow. Based on the considerations made, the research questions are then answered, and the initially established research goals are addressed.

9.1. APPLYING THE FRAMEWORK

Although the presentation of the previous examples shows a clear transition between design stages, this was not the case in the actual experiences, where the framework allowed for flexible design workflows with fluid and ill-defined stage transitions. The aim of the previous studies presentation was to make the step-by-step application of the framework in different design scenarios clearer, along with its results.

In Study 1, the framework supported the development of an irregular truss-like facade structure considering creative intents and performance criteria since early stages. Based on the results, the framework facilitated:

- The geometric exploration of the truss configuration.
- The integration of aesthetic, structural, and economic constraints since conceptual stages.
- The execution of iterative structural analyses and the interpretation of their results.
- The structural improvement of the solutions' without neglecting its aesthetic quality.
- The search for solutions successfully balancing the aesthetic, structural, and cost constraints.

Then, Studies 2, 3, and 4 applied the framework in different collaborative design contexts involving architects with and without AD skills. The results show that the framework was successful in:

- Materializing both the architects' design intent and aesthetic preferences.
- Augmenting the architects' design freedom and the range of solutions considered.
- Integrating conceptual, functional, and performance requirements since early stages.
- Incrementally refining the solutions according to different design criteria.

- Smoothing the transition between design stages and design tasks.
- Alternating between design tools used according to the task at hand.
- Completing the design process in a short time frame.

Finally, Study 5 evaluated the framework in the development and prototyping of a set of unconventional, visually complex facade shading panels considering both conceptual and construction requirements. The results evidence the ability of the framework to simplify:

- The implementation of the design intent for the facade shading panels.
- Their geometric exploration based on aesthetic, performance, and fabrication criteria.
- The test of different panel materials and surfaces finishes.
- The detailing of the panels according to different construction schemes.
- The extraction of technical drawings according to the selected manufacturing strategy.
- The production of a small-scale prototype.

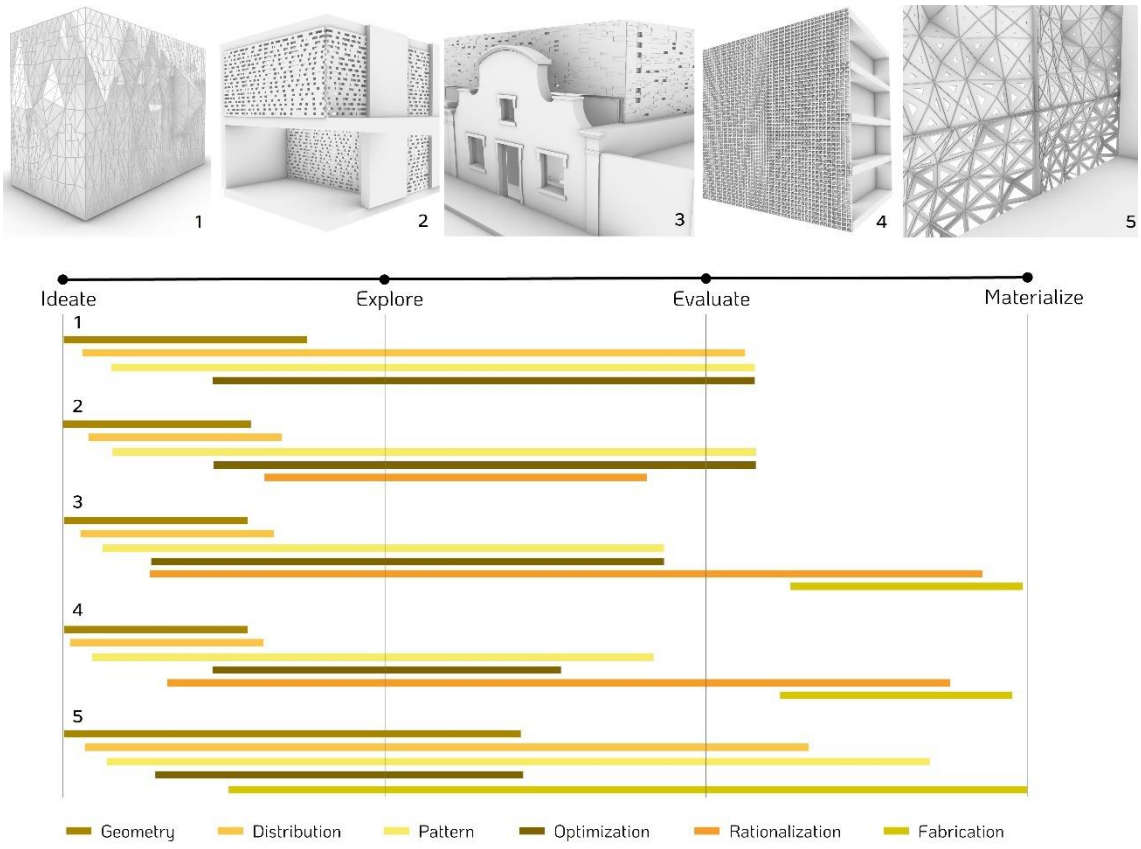


Figure 9.1. Application of the framework's algorithms in the five application studies selected.

Figure 9.1 illustrates the application of the framework in the previous application studies. It represents the different categories of algorithms through a color scheme, demonstrating the framework's ability for handling specific design tasks/problems throughout the design process. Given the mathematical

and parametric nature of the framework's principles, the testing of different design scenarios was facilitated in all application studies, as was the coordination of different design constraints and information. This is visible, for instance, in Study 5, where the framework supported the concurrent combination of algorithms from the Geometry, Distribution, Pattern, and Fabrication categories from early to final design stages, allowing them to directly or indirectly influence each design task performed as well as the architects' decision-making process. As illustrated in Figure 9.2, while the geometry-related algorithms materialize the architects' design intent and aesthetic preferences for the facade panels as well as create different construction schemes and panel details, the manufacturing-related algorithms address the geometric exploration of different visual outcomes and volumetric effects for the panels, as well as their feasibility and preparation for manufacturing.

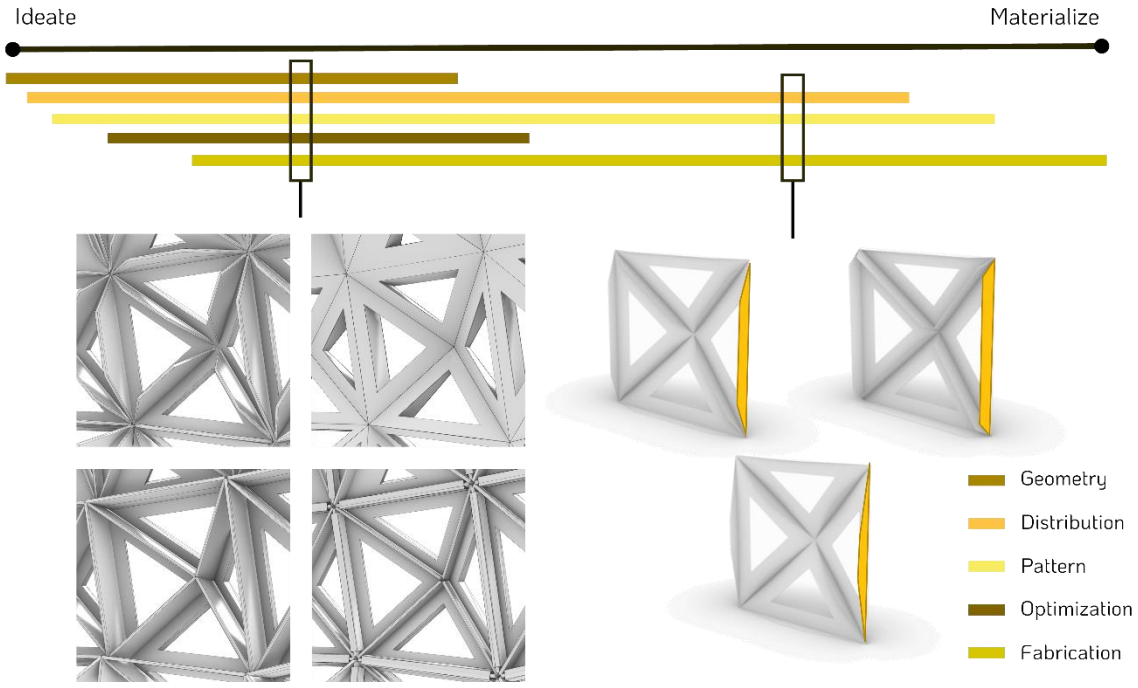


Figure 9.2. Concurrent use of different categories of algorithms in Study 5 and the resulting outcomes.

The application studies also prove the framework's applicability in architectural design, successfully responding to the context-specificity and variability of real-world design practices, while adapting to their different tools and methods. In Study 3, for instance, the framework succeeded in coordinating the existing time, economic, and technical constraints with the design intent of developing a nonconventional, visually complex building facade. Additionally, the framework also allowed the use of the design team's preferred CAD and BIM tools according to the task at hand. Nevertheless, the application of the framework will always depend on the collaboration of an architect with AD skills.

The application studies also show that, in either case, the use of the framework (1) simplifies and facilitates the deployment of AD, by promoting code reuse and thus avoiding repeated work, (2)

reduces the time and effort spent in AD tasks by guiding the selection and combination of its algorithms, and (3) increases the likelihood of achieving better results by leaving more time available for creative and critical thinking tasks. These advantages are evident in Study 4, where the framework allowed the team to consider a wide range of facade design possibilities within the existing time constraints that varied in terms of: (1) geometric composition, exploring several customized bricks of different sizes, shapes, and opacities; (2) facade materiality, by pondering the use of ceramic bricks or wood panels; (3) constructive solution, by considering stacking massive bricks or placing lightwood panels; and (4) manufacturing strategy, by assessing the hypotheses of producing multiple customized bricks or sets of layered panels.

The application studies also evidence an increase in the architects' design freedom, facilitating iterative design changes and the test of multiple solutions and freeing designers from repetitive and time-consuming tasks by automating them, such as those of analysis and optimization processes. By leaving more time available for both creative and design enhancement tasks, the framework also promotes wider design space exploration and the search for improved design solutions. This is visible in Study 1, where the framework was coupled to an optimization routine to automate iterative structural analyses, promoting a gradual balance between the design intent of creating an irregular truss-like facade resembling a crystal and the need to ensure its structural stability and feasibility.

On the other hand, the results also evidenced some challenges deriving from the framework's flexibility, particularly the visualization and analysis of the large set of solutions that potentially results from its use. Although this issue had already been addressed by other authors [383], there is still no widely accepted solution. The application studies tackled this challenge by using the framework to explore the design space in a process entirely guided by the design team, where architects iteratively visualized and changed the solutions according to analysis results, technical constraints, or their personal preferences. By controlling the navigation process through the set of potential solutions, the design team could gradually reduce the design space while ensuring it contained solutions with the intended features. Even in those cases where an optimization routine was applied, as it happened in Study 2, it was the architects who guided the entire design process by defining and redefining, at each iteration, (1) the variables that were allowed to change during the optimization process – in this case, the size and position of the smaller bars – and (2) the variation ranges allowed – in this case, setting the number of possible bar lengths and the maximum distance between them.

Finally, although the proposed principles have been implemented in one specific AD tool, they can be easily implemented in other AD tools, whether visual or textual, or even adapted for other purposes within or outside the architecture field. This is possible because of the principles' mathematical nature and modular structure, which make them universal and independent of specific punctual applications, as well as changeable and adaptable to various scenarios.

9.2. COMPARING AD AND NON-AD WORKFLOWS

The architectural design process starts with the ideation and materialization of design ideas mostly through sketches, handmade schemes, and conceptual models. It then continues with the exploration of the design intent by either using the same approach throughout the entire process or transiting to AD to benefit from its greater design freedom and efficiency, among others. Despite its multiple advantages, the decision of using AD should always consider the initial investment it requires, which may not pay off when dealing with geometrically simple designs that do not need substantial design iterations. In the remaining cases, however, the transition to AD is often advantageous.

Although a non-AD workflow resorting to freehand sketching and physical modeling is more intuitive for design exploration processes than an AD one, when the project is further detailed, design exploration becomes difficult, as it relies on repetitive manual design tasks. Therefore, within the same time frame, not only fewer and less complex designs are produced, but their corresponding models are also less flexible than those produced with AD. In contrast, AD facilitates the application of small-to-large design changes, making it easier to understand their impact and thus efficiently and effectively guide the design process. Nevertheless, identifying the changes we need to apply to the algorithms in order to affect the design in the intended way is not always straightforward. Contrarily, although a design change in a non-AD scenario is more easily tracked because it is directly made to the model, it still requires 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.

When it comes to improving the design performance, a non-AD workflow requires (1) manually modeling each design alternative to then evaluate it; (2) incorporating the evaluation results through manual changes to the model before proceeding to the next iteration; and (3) repeating this process until finding better solutions. As this usually requires multiple iterations, it becomes extremely time-consuming, tedious, and error-prone to improve designs using non-AD approaches. In contrast, in an AD workflow these tasks are automated, increasing the number of solutions evaluated as well as the reliability of the results. Given the latter's flexibility, it is also easier to improve the designs regarding multiple criteria. In contrast, although it is possible to enhance designs with respect to several criteria in a non-AD workflow, and sometimes even more intuitively than in an AD one, the resulting process suffers from the above-mentioned limitations.

Regarding the fabrication of the solutions, an AD workflow facilitates the control of the design's feasibility by allowing for the automatic extraction of construction information and the balance of the resulting manufacturing costs with both conceptual and performance criteria. In a non-AD workflow, however, this scenario would suffer from the same limitations of design exploration and improvement

processes. Moreover, in the adopted AD workflow, both the testing of different construction schemes and the production of technical documentation is facilitated. AD increases the variety of manufacturing scenarios and materials considered, while assessing their impact on the solutions' aesthetics and performance. In contrast, in a non-AD workflow, this ability is dependent on the capabilities of the modeling tool in use, which rarely handles multiple scenarios and strategies efficiently.

By comparing the supported AD workflow with a non-AD one (Figure 9.3), it is possible to conclude that both have advantages and disadvantages for architectural practice. While the non-AD workflow is closer to the architects' creative nature and traditional way of thinking, the AD one is more powerful and, thus, better at handling the complexity and multiplicity of criteria of architectural design practice but it is also less intuitive and more difficult to learn. To promote the dissemination of AD in current design practices, the collaboration between architects with and without AD skills is an increasingly common solution, allowing design teams to benefit from AD's advantages in:

- Creative exploration – increasing the efficiency and flexibility of the process.
- Design development – extending the design space explored.
- Design refinement – achieving improved design solutions.
- Design manufacturing – automating construction-related tasks.

A final but equally important advantage of the supported AD workflow is allowing the previous tasks to be easily coordinated, smoothing the transition between them. This advantage results, on the one hand, from the universality of the framework's principles, which are tool-independent and thus widely applicable; and, on the other hand, from the portability of the AD tool used for its implementation, which ensured interoperability between design tools and thus reduced not only the time and effort spent in the transition between them, but also the error-proneness of these processes.

Design Tasks		non-AD Approach	AD Approach
Design exploration	Design concept definition	Intuitive	Rational
	Design communication	Easy	Hard
	Design space explored	Narrow	Large
	Design manipulation	Hard	Easy
	Design space navigation	Slow and restricted	Efficient and informed
	Model flexibility	Small	Large
	Traceability	Obvious	Non-obvious
Design improvement	Design changes	Error-prone and tedious	Coherent and accurate
	Multi-criteria coordination	Hard	Easy
	Model adaptation	Manual	Automatic
	Analysis results integration	Manual	Automatic
	Iterative analyses	Slow and error-prone	Efficient and accurate
	Design optimization	Manual	Automatic
	Optimized design space	Limited	Extensive
Design production	Design rationalization	Hard	Easy
	Design detailing	Manual	Automatic
	Technical documentation	Manual	Automatic
	Construction information	Tool-dependent	Automatic
	Prototyping	Hard	Easy
	Scenarios considered	Few	Numerous
	Manufacturing control	Limited	Extensive

Figure 9.3. Comparison between a non-AD and an AD workflow on each design stage.

9.3. Answering the Research Questions

This section discusses how the proposed methodology and framework reduces the complexity of AD in solving complex and unconventional facade design problems by answering the research questions

formulated in chapter 1. It starts with the most specific ones, i.e., the research's sub-questions, and then, based on their answers, it responds to the most generic ones, i.e., the main research questions.

SUB-QUESTION 1

How to structure AD according to the specificities of each facade design stage, while smoothing the transition between them?

Building facades have many associated functions and thus their design requires considering several requirements. While some are general, others are context-specific, which makes facade design processes unique and complex. AD has the advantage of facilitating the coordination of such requirements and supporting the search for design solutions that satisfy them.

Nevertheless, AD requires the conversion into algorithms of the geometrical, technical, and performance aspects of facade design. To this end, this thesis analyzed and reproduced a large corpus of contemporary building facades (the design corpus) to better understand the existing requirements and challenges and simultaneously identify relevant design patterns and knowledge. Chapter 7 presented the outcome of this analysis and its systematization in a mathematical methodology tailored for facade design. The premise of such proposal was that providing a framework that organized prior design knowledge in a generic and structured way would allow its reuse in various design contexts, while responding to the specificities of each facade design stage.

To facilitate its application, the methodology was implemented in an algorithmic framework, whose modular structure organizes the available ready-to-use strategies according to their type and function in facade design. Then, chapter 8 applied the framework in five application studies developed in real-world design scenarios with the participation of architects with different levels of AD experience, proving its ability to respond to the challenges and requirements of each design stage and to flexibly coordinate them with those of the other stages. The results showed that the framework successfully systematizes facade design processes, while improving the flexibility of the design workflow.

SUB-QUESTION 2

How to convert conceptual design criteria into algorithmic strategies that respond to different creative intents and design briefs?

Architectural practice has a visual and tactile nature, whereas AD is abstract and less intuitive. While some technical requirements, such as environmental performance, structural stability, and economic viability, can be straightforwardly converted into algorithmic descriptions, since they often involve measurable and tangible quantities, those related to creative aspects, such as aesthetic and conceptual

requirements, are not. For an AD approach to be successful, it is important that the latter requirements, which have a strong visual component, are properly converted into algorithms and adequately combined. To that end, this investigation studied several mathematical strategies to represent geometry-related elements, such as spatial coordinates, curves, surfaces, and solids, as well as to manipulate them in different ways, e.g., by applying geometric transformations.

The seventh chapter of this thesis elaborated on such strategies, placing particular emphasis on how they relate to facade design processes and how they were integrated in the framework. Several conceptual examples were developed to illustrate the strategies considered and their application in facade design. Then, in chapter 8, the previous strategies were applied in realistic design scenarios, demonstrating how they can respond to different creative intents and design practices. The results showed that the framework can successfully integrate different conceptual and aesthetic criteria into an AD workflow, while responding to context-specific creative intents as well as to different real-world design problems and briefs.

SUB-QUESTION 3

How can the provision of a framework reduce the time and effort spent in AD tasks, while handling several design criteria since early design stages?

AD is a complex design approach that requires programming skills. No matter how much experience an architect has in AD, this approach is less intuitive and technically more challenging than traditional ones. Nevertheless, the architectural community using AD is growing, particularly due to the advantages this approach brings to architectural design practice, namely design freedom, efficiency, and accuracy. Previous proposals to smooth the technical complexity of AD mostly focus on visual programming strategies. However, to deal with large-scale design problems, the use of textual programming frequently becomes mandatory due to its higher expressiveness and scalability.

To address the lack of solutions for simplifying AD within the textual programming paradigm, this thesis proposed an algorithmic framework and demonstrated its applicability by targeting the field of facade design. The premise was that the provision of ready-to-use algorithms and strategies would reduce the time and effort spent in AD tasks, by decreasing the amount of code developed and the occurrence of potential errors, leaving more time available for design exploration and thus increasing the design space considered. This is illustrated in the eighth chapter of the thesis through the application of the framework in different design scenarios responding to different creative intents and design briefs. The results showed that the framework facilitated the implementation of the design intent as well as the early-stage integration of different criteria, such as aesthetic, functional, and structural,

allowing the architects to devote more time and energy to both the development and improvement of their facade designs.

Given the complexity of architectural design and the multiple design tools that are currently required in its practice, it is important that the proposed AD strategies are generic and flexible, so as to (1) integrate different design workflows and tools, (2) respond to the context-specificity of the practice, and (3) handle its multiple constraints. Therefore, the proposed methodology adopts the formalism of mathematics to ensure the universal application of its principles, while providing them with the ability to adapt to the variability and specificity of architectural practice. Chapter 7 elaborated on the mathematical formulation of the framework's principles, demonstrating their validity and flexibility by reproducing an already existing corpus of facade designs.

RESEARCH QUESTION

Based on the previous findings, the lessons learned, and the answers to the research sub-questions, this section answers the overarching research question posed by this investigation, namely:

How can we systematize AD to address intricate design problems, particularly those of facade design processes?

AD is a design approach that provides the expressiveness and flexibility needed to handle complex design problems, which are common in facade design processes. To allow architects to benefit from it, this thesis proposed to simplify its use in more complex design problems by providing algorithmic frameworks systematizing specific design tasks. Particularly, this thesis provided an algorithmic framework for the field of facade design, whose mathematical nature ensured its universal application and whose modular structure facilitated its application and adaptation to different design practices and briefs.

Chapter 7 elaborated on both the framework's structure and content, demonstrating its ability to use the collected design knowledge in a flexible way, as well as to simplify the algorithmic reproduction of an already existing corpus of building facades. By systematizing prior design knowledge and guiding its use and combination in new design problems, the framework reduced the time and effort spent in AD tasks, minimizing repeated work and the introduction of errors, while increasing the efficiency of the design process and the probability of achieving better results.

Chapter 8 applied the framework in a set of practice-based studies involving differently skilled architects and tools. The application studies proved, first, the applicability and usefulness of the available algorithmic strategies in an AD context and, second, their flexibility to adapt and respond to different design practices and problems, such as creating an irregular truss-like facade or exploring a set of

facade panels with a random-based configuration and responding to different functional and shading requirements, among others.

The application studies also demonstrated the framework's ability to combine different categories of algorithms in multiple ways and throughout the entire design process, for example using geometry- and analysis-related algorithms to explore different geometric configurations or applying manufacturing-related algorithms to support creative explorations processes, among others. By handling all design stages in a flexible and structured way, the framework allowed the design teams to consider more design possibilities within the limited time available as well as conduct a more comprehensive critical analysis of the designs. Lastly, given the framework's modular structure, strategies that were punctually developed to respond to more specific requirements in each application study could then be easily incorporated in the framework, becoming available for future situations.

Based on the previous considerations, the framework proved to facilitate and simplify the use of AD in tackling diverse complex facades design problems, enhancing the architects' creative exploration and design development processes. Therefore, the thesis answered the main research question. Given the universality and flexibility of the proposed solution, it is expected that these findings and contributions can be generalized to other fields inside or outside architecture and be extended and/or adapted to meet their specific needs. Therefore, this also makes it possible to affirm that the thesis also answered the broader research question of:

How can we reduce the complexity of AD to address intricate design problems?

9.4. Addressing the Research Goals

This section reflects on how the previous findings contributed to reduce the complexity of AD in solving complex and unconventional facade design problems, by fulfilling the initially established research goals of:

Goal 1. Enhancing creative and critical thinking processes.

Goal 2. Articulating different design constraints, tools, and tasks since early stages.

Goal 3. Improving design space exploration and the chance of finding better solutions.

Goal 4. Materializing the solutions through different manufacturing strategies.

Based on the application studies of chapter 8, the use of the framework extended the architects' creative exploration process because it reduced the time and effort spent in the algorithmic implementation of different design ideas, therefore increasing the time available for design exploration. Along with providing AD's general advantages, particularly in terms of design-change propagation and

task automation, the framework allowed the architects to test multiple design alternatives quickly and effortlessly, increasing the range of design possibilities considered and enhancing their creative and critical thinking (Goal 1).

Additionally, the framework also facilitated the early integration of conceptual, functional, environmental, and construction requirements in the design process, as well as the use of different design tools according to the tasks at hand (Goal 2). By facilitating the coordination between different types of data and tasks since initial stages, the architects could navigate the design space in a more informed and conscious way and thus, more easily find design solutions that successfully balance the existing design constraints and requirements (Goal 3).

Finally, the application studies also demonstrated the success of the framework in preparing facade design solutions for fabrication, facilitating their detailing while considering different constructions schemes and automating the production of the corresponding technical documentation according to the selected manufacturing strategy (Goal 4).

V

Conclusion

10. FINAL CONSIDERATIONS

This chapter reflects on the results of this thesis, making some final considerations on its research findings, contributions, and limitations, while suggesting some relevant future work directions.

10.1. RESEARCH OVERVIEW

Architecture is in constant change because it must adapt to the ever-changing nature of society and culture, incorporating the latest technological advancements in its practice to capture the spirit of its time. With the increased sophistication of current representation and fabrication methods, the process of designing and constructing grew in complexity, motivating the use of computation-based design approaches, such as Algorithmic Design (AD), because of their advantages in terms of design flexibility, efficiency, and expressiveness.

Nevertheless, using AD is not trivial, mostly because of its technical complexity and abstraction level, which largely deviate from the visual and tactile nature of architectural practice. With the aim of making AD more accessible to architects, some visual-based AD tools have been proposed but, despite being intuitive and easy to use, they lack the expressiveness and scalability needed to handle large-scale design problems. Unfortunately, there are no successful solutions addressing textual-based AD, which would be critical to deal with such problems.

To address this limitation, this thesis responded to the research question of *How can we reduce the complexity of AD to address intricate design problems?* Given the focus on the AD paradigm, the thesis considered the variability and unpredictability of architectural design in a computation-based perspective. The thesis proposed a mathematical methodology and framework to systematize AD-related tasks and assessed its applicability within the scope of building facades. The aim was to facilitate the use of AD in solving complex and unconventional design problems by providing a set of design principles that could be easily combined in the development of new design solutions. Given the endless variety of design problems, it is only natural that the principles provided do not cover all possibilities and, therefore, the framework was structured and implemented to be continuously extended with additional design knowledge and principles whenever necessary.

To evaluate the viability and usefulness of the proposal for real-world design scenarios, its principles were applied in a set of architectural application studies involving different design practices

and problems. After critically reflecting on the previous results, the thesis identified the research merits and limitations and answered the research question by proving the ability of the proposal to facilitate and simplify the use of AD in tackling diverse complex design problems, while enhancing the architects' creative exploration and design development processes.

10.2. Research Findings

This section summarizes the main research findings of this thesis in terms of usefulness and merits of the proposed methodology and framework and the success of their application in facade design processes.

Mathematical Framework

The framework presented in chapter 7 was developed to systematize AD processes. To assess its applicability, this thesis applied the framework in the context of facade design. However, the framework can also be applied to other fields to structure the corresponding specific tasks. This thesis presented the set of methodological steps to follow in order to create similar frameworks specialized in other tasks.

The framework presents a modular structure classifying the collected design knowledge according to their type and role in facade design processes, providing sets of design strategies that can be differently combined in the development of new solutions. For each category, the most relevant strategies were presented and illustrated in chapter 7 through a set of conceptual examples. The purpose of this modular classification is to facilitate knowledge reuse, guiding the selection and combination of the available principles in future design problems, while increasing the flexibility of the design process. The modular structure also aims to ensure the mutability of the formulated principles and the framework's ability to be continuously extended with additional strategies.

Based on the research findings, the thesis concluded that the framework's modular organization simplifies the matching problem between existing design requirements and possible solutions. The application studies of chapter 8 proved this through (1) the short time spent in the design intent implementation and (2) the diversity and quality of the solutions obtained. Moreover, the application studies also proved the framework's ability to coordinate different types of design data and constraints throughout the design process, such as the architects' design intentions, performance criteria, and specific site constraints.

As illustrated in chapter 7, the framework has a high expressive power, allowing the generation of a wide variety of design possibilities. However, when using the framework in chapter 8, the economic, technical, and time constraints of the application studies narrowed the scope of the strategies used, leading to a more limited range of design variations than the framework allows. This narrowing results

from the application of the framework to real design problems and constraints, proving its ability to adapt to context-specific design constraints and to control design complexity.

Design Workflow

The aim of the mathematical methodology presented in chapter 7 was to ensure that the previous systematization of facade design processes had the widest possible application in architectural practice and, possibly, in other design fields. In this methodology, the collected design knowledge was described through mathematical formalisms, each one representing a relevant design strategy and principle. Chapter 7 elaborated on the structure and combination of some of these formalisms. The aim was to make the collected strategies and principles as universal as possible, allowing them to be easily integrated in different design practices and tools or extended to respond to the specificity and variability of architectural design problems.

Based on the findings of chapter 8, the proposed methodology can be applied in architectural practice, responding to the technical specificities of each design problem, such as performance requirements and construction limitations, as well as to the design teams' different creative intents and methodologies, such as preferred design tools and varying levels of AD experience.

The application studies also proved the methodology's success in supporting design workflows providing greater design freedom and efficiency, enabling designers to explore wider design spaces and thus consider a wider range of design possibilities within a limited time frame. By freeing time for design exploration, the methodology allowed architects to enhance both their creativity and decision-making process, increasing the chances of finding better solutions. Finally, the use of the methodology alongside non-AD design processes also proved that, rather than replacing traditional design processes, the research proposal complements them, particularly when the design complexity demands it.

Algorithmic Implementation

The main goal of this thesis was to structure AD processes in order to reduce their complexity and make their implementation and use more accessible to architects with AD skills. Therefore, the author addressed the variability and unpredictability of architectural design processes in a computation-based perspective, implementing the proposed methodology in an algorithmic framework and applying it in several examples to assess its usefulness and expressiveness. In a first stage, the framework was applied in conceptual and simpler design tests, such as exploring different geometric composition, generating different facade patterns, among others. The goal was to check if the proposed principles could reproduce prior design knowledge in a more efficient and expressive way. In a second stage, the principles were applied in practice-based design problems to assess their suitability and usefulness to support real-world design scenarios.

According to the findings of chapters 7 and 8, the framework's algorithmic implementation was successful in supporting facade design processes because it provided the expressiveness needed to (1) support higher levels of design complexity, (2) quickly and effortlessly explore wider design spaces, (3) coordinate multiple types of design requirements, (4) smoothly transit between design stages and tasks from conceptual exploration to fabrication, (5) incrementally improve the solutions in an interactive and conscious way, and (6) achieve better solutions in terms of aesthetics, performance, and feasibility within a limited time frame.

Furthermore, as the AD tool used to implement the framework (Khepri) is portable between different design tools and AD libraries, it was possible to demonstrate its ability to benefit from other frameworks specialized in, for instance, analysis, optimization, and visualization strategies.

10.3. Thesis Contributions and Merits

The outcomes of this thesis are outlined below according to three criteria: theoretical, methodological, and practical.

Theoretical: Mathematical Framework

Based on an extensive literature review and an analysis of a large set of contemporary facades, this investigation identified common design problems and relevant design strategies, particularly in the field of facade design. The collected design knowledge was then analyzed and organized by type and role in design processes and mathematically formulated in a framework specialized in facade design.

The proposed framework constitutes a theoretical contribution that identifies and structures prior design knowledge, while proposing solutions to formalize it in a universal and flexible way. Additionally, the framework enables the production of novel design knowledge by systematizing the use and combination of the collected design knowledge in new design problems. Lastly, and most importantly, the framework outlines the principles that are necessary for the creation of other specialized frameworks containing different types of strategies and categorical structures.

Methodological: Algorithmic-oriented Methodology

This thesis presented an algorithmic-oriented methodology to support large-scale and unconventional design problems, organizing a set of mathematical formalisms addressing different tasks of facade design processes in a modular way.

The first methodological contribution is the provision of a classification structure that organizes different design processes and their related information and that guides their combination and application in different design scenarios. A second contribution is the proposed modular structure to organize the formulated principles, whose flexibility allows it to adapt to the diversity and variability of

architectural design processes as well as to serve as the basis for future methodologies and applications. The last methodological contribution encompasses the strategy proposed to systematize the complexity of facade design processes through an AD perspective. By presenting the methodological steps that lead to the proposed strategy, the thesis enables the latter's replication in the systematization of other design tasks.

PRACTICAL: SUPPORTING FACADE DESIGN PROCESSES

This thesis assessed the practical contributions of the proposal by, first, implementing the formulated principles in an AD tool for architectural design and then, applying the resulting algorithmic framework in several architectural application studies responding to different design intents and problems.

The findings demonstrated how the framework simplified the use of AD in facade design processes involving architects with and without AD skills and resorting to different design methods and tools. The findings also proved the ability of the framework to address more complex design problems and generate unconventional solutions that would be difficult to achieve otherwise. Lastly, the findings also evidenced an increase in the flexibility and efficiency of the resulting design workflows, which facilitated the transitions between design stages and the coordination of different specific design tasks and requirements. By increasing creative design exploration and freeing time for critical analysis, the design possibilities considered also increased, as did the likelihood of achieving better design solutions.

10.4. LIMITATIONS AND FUTURE WORK

This section outlines some of the research limitations of the thesis and suggests relevant improvements, as well as future research directions.

RESEARCH LIMITATIONS

One of the most obvious limitations is the framework's abstract and non-visual nature. Although the provision of a Graphical User Interface (GUI) would make its application more user-friendly and intuitive, this task was not considered, given the context and time available for this investigation. Furthermore, the goal of this thesis was to lay the foundations for a methodology and framework systemizing AD processes and thus the research focused on ensuring its universal application and continued extension. Moreover, although a GUI could facilitate the use of the framework, it would also limit its widespread application and integration with other specialized tools and frameworks and, above all, it would hamper its expressiveness, flexibility, and scalability.

The second limitation of this investigation stems from the fact that the proposed framework was only implemented in one AD tool. Nevertheless, given the framework's mathematical and modular

nature, its principles can be implemented in other AD tools, whether visual or textual, or even adapted for other purposes within or outside the scope of architecture.

As explained in chapter 7, the process of developing the framework focused, first, on the geometry-related aspects of facade design processes and, only then, on those related with the analysis, rationalization, and fabrication of design solutions. As a result, the proposed categories of principles show different levels of development, the former ones containing a greater diversity of strategies than the latter ones. This imbalance is mainly due to the higher complexity and technical difficulty of the processes covered by the latter categories, particularly optimization and rationalization processes, making it challenging to achieve the level of competence of existing specialized tools, considering the time and human resources involved in this research. Hence, given the modular nature of the framework, it was possible to couple it with other specialized frameworks and libraries, particularly for building performance analysis and optimization (see section 7.3.4). This enabled the combination of the framework's principles with more specific strategies to address particular design problems. Indeed, providing a framework that can be continuously extended whenever the need arises was essential in addressing the research problem.

One limitation of the research evaluation is the absence of a full-scale prototype with a direct application in a construction scenario. Although several attempts have been made to collaborate with the industry to produce some of the solutions developed with the framework, none were successful mostly because of production costs and the lack of availability of the manufacturing companies contacted. Nevertheless, it is still a goal of this research to produce a full-scale prototype and efforts are currently being developed in that direction.

Finally, the lack of quantitative data to evaluate the use of the framework supporting the qualitative analysis made in chapters 8 and 9 is another limitation of this study. For instance, the assessment made in chapter 9 was purely qualitative and based on the options and assumptions of the architects involved in the application studies. However, given the abstract, contextual, and imprecise nature of creative design processes, it would be difficult to measure the quality of the results in a quantitative way. For that reason, this research adopted an evaluation method heavily based on the participants' practical experience and know-how.

Suggestions for Future Research

This thesis addressed the systematization of AD to solve complex design problems, proposing solutions to the research problems tackled, while raising others to address as future work.

The first one regards the framework's geometry-related capabilities for shape generation and manipulation, which can be further improved with additional strategies addressing a wider range of surface shapes, grids, and elements. The same applies to the available rationalization strategies, which,

in their current state, support a limited range of surface shapes, discretization methods, and surface paneling strategies; and to the analysis/optimization strategies, which could integrate additional performance simulations, such as thermal, energy, and acoustic, and metrics, such as construction cost and waste control, among others.

The second suggestion for future work involves the improvement of the framework's manufacturing capabilities. Further research should be done in order to increase the range of construction details and elements allowed, as well as the diversity and quality of the technical documentation produced, e.g., embracing the representation requirements resulting from other fabrication strategies and materials. Additionally, it is important to extend the framework with more manufacturing-related criteria, such as geometric properties, materiality, surface finish, machining time, material waste, and production cost, while increasing its sensibility to the specific manufacturing requirements of each fabrication technology. For example, while in 3D printing and CNC cutting the planning of the printing/cutting path is critical to increase the surface quality and shape accuracy and decrease production time and material expenditure, in CNC cutting 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.

Finally, it would also be relevant to continue using the framework in real-world design scenarios and test its ability to produce different prototypes involving different materials and manufacturing strategies. This is important to improve the framework in terms of structure, content, and application, as well as to identify new design requirements and research directions that may eventually arise. This is also important to evaluate the opportunities its use brings to architectural practice, smoothing the transition between design stages, facilitating design collaboration between different specialists, and thus increasing control over the whole process from design to fabrication.

RELEVANT PUBLICATIONS BY THE AUTHOR

Journal articles

Caetano, I., & Leitão, A. (2019). Integration of an Algorithmic BIM Approach in a Traditional Architecture Studio. Journal of Computational Design and Engineering, 6.

Caetano, I., & Leitão, A. (2020). Architecture Meets Computation: an Overview of the Evolution of Computational Design Approaches in Architecture. Architectural Science Review, 63(2).

Caetano, I., Leitão, A., & Bastos, F. (2020). From Architectural Requirements to Physical Creations. Journal of Façade Design and Engineering, 8(2).

Caetano, I., Santos, L., & Leitão, A. (2020). Computational design in architecture: Defining parametric, generative, and algorithmic design. Frontiers of Architectural Research, 9(2).

Castelo-Branco, R., Caetano, I., & Leitão, A. (2022). Digital representation methods: The case of algorithmic design. Frontiers of Architectural Research, 11(3).

Castelo-Branco, R., Caetano, I., Pereira, I., & Leitão, A. (2022). Sketching Algorithmic Design. Journal of Architectural Engineering, 28(2).

Castelo-Branco, R., Caetano, I., & Leitão, A. (2022). Algorithmic Representation Space. Prospectives Journal, 2.

Caetano, I., Pereira, I., & Leitão A. (2022). Balancing Creativity and Performance Analysis: an algorithmic design approach. A|Z ITU Journal of the Faculty of Architecture (Submitted).

Caetano, I. & Leitão A. (2022). Large Scale Algorithmic Design. Automation in Construction (In preparation).

Book chapters

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

Caetano, I., & Leitão, A. (2022). Behind Algorithmic Geometric Patterns: A Framework for Facade Design Exploration. In P. Lizancos, D. Leite Viana, F. Morais, & J. Vieira Vaz (Eds.), Formal Methods in Architecture: Advances in Science, Technology & Innovation (IEREK Interdisciplinary Series for Sustainable Development). Springer, Cham (To be published).

Caetano, I., Leitão, A., & Bastos, F. (2022). Converting Algorithms into Tangible Solutions: A Workflow for Materializing Algorithmic Facade Designs. In A. Correia, M. Azenha, P. Cruz, P. Novais, & P. Pereira (Eds.) Trends on Construction in Post-Digital Era: ISIC 2022. Lecture Notes in Civil Engineering, vol 306. Springer, Cham.

Caetano, I., & Leitão, A. (2022). The Right Algorithm for the Right Shape: An algorithmic framework for efficient design and conception of building facades. In Barberio, M., Colella, M., Figliola, A., & Battisti, A. (Eds.), Architecture and Design for Industry 4.0: Theory and Practice. Springer (To be published).

Conference proceedings

Caetano, I., & Leitão, A. (2017). Integration of an Algorithmic BIM Approach in a Traditional Architecture Studio. Protocols, Flows and Glitches: Proceedings of the 22nd CAADRIA Conference.

Alfaiate, P., Caetano, I., & Leitão, A. (2017). Luna Moth Supporting Creativity in the Cloud. Disciplines & Disruption: Proceedings of the 37th ACADIA Conference.

Caetano, I., Belém, C., Ilunga, G., Feist, S., Leitão, A., & Bastos, F. (2018). Casos de Estudo sobre a Integração de Processos de Projeto Algorítmico em Fluxos de Trabalho de Projeto em Modelo BIM. 2º Congresso Português de BIM.

Caetano, I., Ilunga, G., Belém, C., Aguiar, R., Feist, S., Bastos, F., & Leitão, A. (2018). Case Studies on the Integration of Algorithmic Design Processes in Traditional Design Workflows. Learning, Adapting and Prototyping - Proceedings of the 23rd CAADRIA Conference.

Caetano, I., & Leitão, A. (2018). Algorithmic Patterns for Facade Design: Merging design exploration, optimization and rationalization. FACADE TECTONICS 2018 World Congress Conference Proceedings.

Sammer, M., Leitão, A., & Caetano, I. (2019). From Visual Input to Visual Output in Textual Programming. Intelligent & Informed - Proceedings of the 24th CAADRIA Conference.

Caetano, I., & Leitão, A. (2019). Weaving Architectural Façades: Exploring algorithmic stripe-based design patterns. "Hello, Culture": Proceeding of the 18th CAADFutures Conference.

Santos, L., Caetano, I., Pereira, I., & Leitão, A. (2020). A Generative System for the Design of High-Performing Shading Devices: Exploring the Daylight Potential of Weaving Patterns. Planning Post Carbon Cities - Proceedings of the 35th PLEA Conference.

Castelo-Branco, R., Caetano, I., Pereira, I., & Leitão, A. (2020). The Collaborative Algorithmic Design Notebook. Imaginable Futures: Design Thinking, and the Scientific Method - Proceedings of the 54th ASA Conference.

Caetano, I., & Leitão, A. (2020). When the Geometry Informs the Algorithm: A hybrid visual/textual programming framework for facade design. Anthropologic - Architecture and Fabrication in the cognitive age: Proceedings of the 38th eCAADe Conference.

Caetano, I., Garcia, S., Pereira, I., & Leitão, A. (2020). Creativity inspired by analysis: An algorithmic design system for designing structurally feasible façades. RE: Anthropocene: Proceedings of the 25th CAADRIA Conference.

Santos, L., Caetano, I., Leitão, A., & Pereira, I. (2021). Uncertainty in daylight simulations of algorithmically generated complex shading screens. Proceedings of the 17th IBPSA Building Simulation Conference.

REFERENCES

1. Horváth, I. (2007). Comparison of Three Methodological Approaches of Design Research. International Conference on Engineering Design, ICED'07, 1–11.
2. Wang, Z., He, W. P., Zhang, D. H., Cai, H. M., & Yu, S. H. (2002). **Creative design research of product appearance based on human–machine interaction and interface**. Journal of Materials Processing Technology, 129(1), 545–550.
3. Kelly, N., & Gero, J. S. (2021). **Design thinking and computational thinking: A dual process model for addressing design problems**. Design Science, 1–15.
4. Alexander, C., Ishikawa, S., & Silverstienm, M. (1977). **A Pattern Language: Towns, Buildings, Construction**. Oxford University Press.
5. Qian, Z. C., Chen, Y. V, & Woodbury, R. (2008). **Developing a simple repository to support authoring learning objects**. International Journal of Advanced Media and Communication, 2(2), 154–173.
6. Johnson, B. & Foote, B. (1988). **Designing Reusable Classes. Journal of object-oriented programming**, 1(2), 22-35.
7. Qian ZC (2009) **Design Patterns: Augmenting Design Practice in Parametric CAD Systems**. Simon Fraser University, Burnaby, Canada.
8. Stiny, G. (1980). **Introduction to shape and shape grammars**. Environment and Planning B, 343–351.
9. Correia, R. (2013). **DESIGNA: A Shape Grammar Interpreter**. Instituto Superior Técnico, Universidade de Lisboa.
10. Castelo-Branco, R., Caetano, I., & Leitão, A. (2022). **Digital representation methods: The case of algorithmic design**. Frontiers of Architectural Research, 11(3), 527–541.
11. Groat, L., & Wang, D. (2013). **Architectural Research Methods**. John Wiley & Sons, Inc.
12. Baper, S. Y., & Hassan, A. S. (2012). **Factors Affecting the Continuity of Architectural Identity**. American Transactions on Engineering & Applied Sciences, 1(3), 2229–1652.
13. Boswell, C. K. (2013). **Exterior Building Enclosures: Design process and composition for innovative facades**. John Wiley & Sons, Inc.
14. Schittich, C. (2006). Building Skins (C. Schittich (Eds.)). Birkhäuser.
15. ElGhazi, Y. S. (2009). **Building Skins in the Age of Information Technology**. Faculty of Engineering at Cairo University.
16. Trubiano, F. (2013). P**erformance Based Envelopes: A Theory of Spatialized Skins and the Emergence of the Integrated Design Professional**. Buildings, 3, 689–712.
17. Picco, M., Lollini, R., & Marengo, M. (2014). **Towards energy performance evaluation in early-stage building design: A simplification methodology for commercial building models**. Energy and Buildings, 76, 497–505.
18. Habraken, A. P. H. W. (2011). **Structural Dynamic Façades**. International Conference on Textile Composites and Inflatable Structures Structural Membranes, 397-408.
19. Knaack, U., & Bilow, M. (2007). Façades: Principles of Construction (1st ed., Vol. 53). Birkhäuser Verlag.
20. El Sheikh, M. M. (2011). Intelligent building skins: Parametric-based algorithm for kinetic facades design and daylighting performance integration. University of Southern California.
21. 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.
22. Herzog, T., Krippner, R., & Lang, W. (2004). Facade Construction Manual. Birkhäuser, Publisher for Architecture.
23. Schulz, C. N. (1971). Existence, space & architecture. Praeger. Stamps.
24. Rapoport, A. (1969). House Form and Culture. Prentice-Hall ,INC.
25. Schimek, H., Stavric, M., & Wiltsche, A. (2008). **The Intelligence of Ornaments: Exploring ornamental ways of Affordable Non-Standard Building Envelopes**. Proceedings of the 13th CAADRIA Conference, 417–425.
26. Stojšić, M. (2017). **(New) Media Facades: Architecture and/as a Medium in Urban Context**. AM Journal of Art and Media Studies, 12, 135–148.
27. Venturi, R., Brown, D. S., & Izenour, S. (1972). Learning from Las Vegas. MIT Press.
28. Alishah, M., Ebrahimi, A., & Ghaffari, F. (2016). **The Role of Buildings Facades of on Urban Landscape**. The Turkish Online Journal of Design, Art and Communication - TOJDAC.
29. Sung, D. (2016). **A New Look at Building Facades as Infrastructure**. Engineering, 2, 63–68.
30. Alberti, L. B. (1452). De re Aedifictoria (On the Art of Building in Ten Books).
31. Serlio, S. (1619). Tutte L'opere D'architettura, Et Prospetiva. Giacomo de Franceschi.
32. Palladio, A. (1570). I quattro libri dell'architettura. Dominico de' Francheschi.
33. Carpo, M. (2011). The Alphabet and the Algorithm. MIT Press.
34. Serlio, S. (1537). Regole generali di architettura.
35. Loos, A. (1913). Ornament and Crime. Cahiers d'aujourd'ui (Issue 5).
36. Corbusier, L. (1923). Vers une architecture (C. Arts (eds.)).

37. Berrett, J. (2018). **Media architecture: Content with purpose for the public**. Proceedings of the Media Architecture Biennale (MAB'18), 28–34.
38. Herr, C. M. (2012). **Non-Trivial Media Façades**. Beyond Codes and Pixels: Proceedings of the 17th International Conference on Computer-Aided Architectural Design Research in Asia, 99–108.
39. Meridith, M., & Sasaki, M. (2008). From Control to Design: parametric/algorithmic architecture. Actar-D.
40. Kolarevic, B., & Parlac, V. (2015). Adaptative, Responsive Building Skins. In B. Kolarevic & V. Parlac (Eds.), Buildings Dynamics: Exploring an architecture of change (p. 70). Routledge.
41. Fortmeyer, R., & Linn, C. (2014). Kinetic Architecture: Design for Active Envelopes. The Images Publishing Group.
42. Vitruvius, M. (1914). The Ten Books on Architecture (M. H. Morgan (Eds.)). Harvard University Press & Oxford University Press.
43. Balik, D., & Allmer, A. (2016). **A critical review of ornament in contemporary architectural theory and practice**. ITU A|Z, 13(1), 157–169.
44. Chevrier, J.-F., & Herzog, J. (2006). Ornament, structure, space: A conversation with Jacques Herzog. El Croquis, 129-130, 22-40.
45. Pell, B. (2010). The Articulate Surface: Ornament and Technology in Contemporary Architecture. Birkhäuser GmbH.
46. Salingaros, N. (2013). The Quest to "Liberate" Architecture from Modernism's Evils: An Interview with Nikos Salingaros.
47. Moussavi, F., & Kubo, M. (Eds.). (2006). The Function of Ornament. Actar.
48. Moughtin, C., Oc, T., & Tiesdell, S. (1995). Urban Design: Ornament and Decoration. ArchitecturePress.
49. Abu-ghazzeh, T. M. (1997). **Signs, Advertising and the Imageability of Buildings: A Perceptual Selection in the View from the Street in Amman, Jordan**. Habitat International, 21(2), 255–267.
50. Askari, A. H., & Dola, K. B. (2009**). Influence of Building Façade Visual Elements on Its Historical Image: Case of Kuala Lumpur City , Malaysia**. Journal of Design and Built Environment Influence, 5(December), 49–59.
51. Pedersen, W. (1987). Intentions. In R. Sonia, D. T. Abramson, & P. Goldberger (Eds.), Kohn Pedersen Fox: Buildings and Projects (pp. 302–303). Rizzoli.
52. Koolhaas, R. (2018). Elements of Architecture. Tashen.
53. Wright, F. L. (1931). Modern Architecture: Being the Kahn Lectures for 1930. Princeton University Press.
54. Gordon, E. (1953). The Threat to the Next America. House Beautiful, April, 126–251.
55. Venturi, R. (1966). Complexity and Contradiction in Architecture (1st ed.). Museum of Modern Art.
56. Jencks, C. (2005). The Iconic Building.
57. Picon, A. (2013). Ornament: The Politics of Architecture and Subjectivity. In AD Primers. John Wiley & Sons Ltd.
58. Riisberg, V., & Munch, A. (2015). **Decoration and Durability: Ornaments and their "appropriateness" from fashion and design to architecture**. ARTIFACT, III(3).
59. McNicholas, M. T. (2006). The relevance and transcendence of ornament: A new public high school for the South Side of Chicago (Issue April). University of Notre Dame.
60. Elrayies, G. M. (2017). Architectural ornaments in the twenty-first century: An analytical study. Cities' Identity Through Architecture and Arts: Proceedings of the International Conference on Cities' Identity through Architecture and Arts (CITAA 2017), Cairo, Egypt, 9–25.
61. Mitrache, A. (2012). **Ornamental art and architectural decoration**. Procedia - Social and Behavioral Sciences, 51, 567–572.
62. Miller, K. (2011). **Organized Crime: The Role of Ornament in Contemporary Architecture**. ACADIA Regional 2011: Parametricism, 67–73.
63. Sağlam, H. (2014). **Re-thinking the Concept of "Ornament" in Architectural Design**. Procedia - Social and Behavioral Sciences, 122, 126–133.
64. Dietterlin, W. (2006). Architecture of Division Symmetry and Proportion of the Five Columns (1598). In T. Nebois (Eds.), Architectural Theory: From the Renaissance to The Present (pp. 520–529). Taschen.
65. Gibbs, J. (1978). A Book of Architecture.
66. Semper, G. (1989). The Four elements of Architecture and Other Writings. Cambridge University Press.
67. Sullivan, L. (1892). Ornament in Architecture. The Engineering Magazine.
68. Touloupaki, E., & Theodosiou, T. (2017). **Performance simulation integrated in parametric 3D modeling as a method for early-stage design optimization - A review**. Energies, 10(637).
69. Dritsas, S. (2012). **Design-Built: Rationalization Strategies and Applications**. International Journal of Architectural Computing, 10(04), 575–594.
70. Boeck, L. De, Verbeke, S., Audenaert, A., & Mesmaeker, L. De. (2015). **Improving the Energy Performance of Residential Buildings: A literature review**. Renewable and Sustainable Energy Reviews, 52, 960–975.
71. Huang, Y., & Niu, J. (2015). **Optimal Building Envelope Design Based on Simulated Performance: History, current status and new potentials.** Energy and Buildings, 117(September), 387–398.
72. Belém, C. G. (2019). **Optimization of Time-Consuming Objective Functions: Derivative-free approaches and their application in architecture**. Instituto Superior Técnico, University of Lisbon.
73. Otani, M., & Kishimoto, T. (2008). **Fluctuating Patterns of Architecture Façade and their Automatic Creation**. Proceedings of the 13th CAADRIA Conference, 375–382.
74. Waseef, A., & El-Mowafy, B. N. (2017). **Towards a new classification for responsive kinetic facades**. Proceedings of the Memaryat International Conference "MIC 2017," March.

75. Velasco, R., Brakke, A. P., & Chavarro, D. (2015). **Dynamic façades and computation: Towards an inclusive categorization of high performance kinetic façade systems**. The next City - New Technologies and the Future of the Built Environment: 16th International Conference CAAD Futures, 172–191.
76. Woodbury, R., Aish, R., & Kilian, A. (2007). **Some Patterns for Parametric Modeling**. Expanding Bodies: Art • Cities• Environment: Proceedings of the 27th Annual Conference of the Association for Computer Aided Design in Architecture, 222–229.
77. Hudson, R. (2010). **Strategies for parametric design in architecture: An application of practice led research**. PhD thesis, University of Bath.
78. Su, H., & Chien, S. (2016). **Revealing patterns: Using parametric design patterns in building façade design workflow**. Living Systems and Micro-Utopias: Towards Continuous Designing, Proceedings of the 21st International Conference on Computer-Aided Architectural Design Research in Asia, 167–176.
79. Caetano, I., Santos, L., & Leitão, A. (2015). **From Idea to Shape, From Algorithm to Design: A Framework for the Generation of Contemporary Facades**. The next City - New Technologies and the Future of the Built Environment: 16th International Conference CAADFutures, 483.
80. Schodek, D., Bechthold, M., Griggs, J. K., Kao, K., & Steinberg, M. (2005). Digital Design and Manufacturing: CAD/CAM Applications in Architecture and Design. John Wiley & Sons, Inc.
81. Mitchell, W. J. (2006). **From Sketchpad to City of Bits: A Story of Shifting Intentions**. Proceedings of the 11th CAADRIA Conference, 1–5.
82. Davis, D. (2013). **Modelled on Software Engineering: Flexible Parametric Models in the Practice of Architecture**. PhD thesis, RMIT University.
83. Woodbury, R. (2010). Elements of Parametric Design. Routledge.
84. Dunn, N. (2012). Digital Fabrication in Architecture (1st ed.). Laurence King Publishing Ltd.
85. Frazer, J. (2016). Parametric Computation: History and Future. AD Magazine: Parametricismo 2.0: Rethinking Architecture's Agenda for the 21st Century, 86(02), 18–23.
86. Kalay, Y. (2004). Architecture's New Media: Principles, Theories, and Methods of Computer-Aided Design. MIT Press.
87. Sutherland, I. (1975). Structure in Drawings and the Hidden-Surface Problem. In N. Negroponte (Eds.), Reflections on Computer Aids to Design and Architecture. Petrocelli/Charter.
88. Eastman, Charles. (1978). The Representation of Design Problems and Maintenance of their Structure. In J. Latombe (Eds.), Application of Artificial Intelligence and Pattern Recognition in Computer-Aided Design, IFIPS Working Conference (pp. 335 – 337). Amsterdam, North-Holland.
89. Leler, W. (1988). Constraint Programming Languages: their specification and generation. Addison-Wesley Longman, Incorporated.
90. Gross, M. D. (1989). **Relational Modeling: A Basis for Computer-Assisted Design**. CAAD Futures Digital Proceedings 1989, 123–136.
91. Rocker, I. M. (2006). When Code Matters. Programming Cultures: Architectural Design and Programming. Architectural Design Magazine, 76(4), 16–25.
92. Tedeschi, A. (2014). Introduction: ADD Algorithms-Aided Design. In A. Tedeschi (Eds.), ADD Algorithms-Aided Design: Parametric Strategies using Grasshopper (pp. 15–32). Le Penseur.
93. Rogers, D. F., & Adams, J. A. (1990). Mathematical Elements for Computer Graphics (2nd ed.). McGraw-Hill, Inc.
94. Brown, Andre. (1986). **A Year's Experience with CATIA and CADAM**. Teaching and Research Experience with CAAD: 4th eCAADe Conference Proceedings.
95. Aish, R., & Bredella, N. (2017). **The evolution of architectural computing: From Building Modelling to Design Computation**. Arq: Architectural Research Quarterly, 21(1), 65–73.
96. Dorst, K., & Dijkhuis, J. (1995). **Comparing paradigms for describing design activity**. Design Studies, 16(2), 261–274.
97. Reffat, R. M. (2006). **Computing in Architectural Design: Reflection and approach to New Generations of CAAD**. ITcon, 11(October 2005), 655–668.
98. McCullough, M. (2006). 20 Years of Scripted Space. Programming Cultures. Architectural Design Magazine, 76(4), 12–15.
99. Asanowicz, A. (1999). **Evolution of Computer Aided Design: three generations of CAD**. In A. Brown, M. Knight, & P. Berridge (Eds.), Architectural Computing: From Turing to 2000 - proceedings of the international eCAADe conference, 94–100.
100. Achten, H. (2009). **Experimental Design Methods - A Review**. International Journal of Architectural Computing, 7(4), 505–534.
101. Terzidis, K. (2006). Algorithmic Architecture (1st ed.). Elsevier Ltd.
102. Burry, Mark. (2011). AD Primers: Scripting Cultures: Architectural Design and Programming. John Wiley & Sons Ltd.
103. Leach, N. (2009). Digital Morphogenesis. Theoretical Meltdown. Architectural Design Magazine, 79(1), 32–37.
104. Celani, G., & Veloso, P. (2015). **CAAD conferences: A brief history**. In Gabriela Celani, D. M. Sperling, & J. M. S. Franco (Eds.), The Next City: 16th International Conference, 47–58.
105. Peters, B., & Peters, T. (2014). Introduction. In Inside Smartgeometry: Expanding the Architectural Possibilities of Computational Design (pp. 8–19). John Wiley & Sons Ltd.
106. Koutamanis, A. (2005). **A Biased History of CAAD: The bibliographic version**. Digital Design - The Quest for New Paradigms: Proceedings of the 23th eCAADe Conference, 629–637.

107. Papamichael, K., & Protzen, J. P. (1993). **The Limits of Intelligence in Design**. Computer-Assisted Building Design Systems: The Fourth International Symposium on System Research, Informatics and Cybernetics, 1–10.
108. Banham, R. (1960). Theory and Design in the First Machine Age (2nd ed.). MIT Press.
109. Alexander, C. (1964). Notes on the Synthesis of Form. Harvard University Press.
110. Negroponte, N. (2011). Towards a Humanism Through Machines (1969). In AD Reader: Computational Design Thinking (pp. 511–512). John Wiley & Sons Ltd.
111. Sutherland, I. (1963). SketchPad: A man-machine graphical communication system. Proceedings of the AFIPS Spring Joint Computer Conference, 23, 323–328.
112. Ahlquist, S., & Menges, A. (2011). Introduction: Computational Design Thinking. In AD Reader: Computational Design Thinking (pp. 10–29). John Wiley & Sons Ltd.
113. Dietz, A. G. H. (1974). Dwelling House Construction (4th ed.). MIT Press.
114. Mitchell, W. J. (1977). Computer-Aided Architectural Design. Van Nostrand Reinhold.
115. Yessios, C. L. (1973). Syntactic Structures and Procedures for Computable Site Planning. Carnegie-Mellon University, Pittsburgh.
116. Akin, O. (1979). Models of Architectural Knowledge - An Information Processing Model of Design. Carnegie-Mellon University, Pittsburgh.
117. March, L., & Steadman, P. (1971). The Geometry of Environment : An Introduction to Spatial Organization in Design. MIT Press.
118. Eastman, C. (Eds.). (1975). Spatial synthesis in computer-aided building design. Applied Science.
119. Lawson, B. (1980). How Designers Think (1st ed.). Architectural Press.
120. Novak, M. (1988). Computational Compositions. In P. J. Bancroft & A. Arbor (Eds.), Computing in Design Education, ACADIA 88' Workshop Proceedings, 5–30.
121. McCullough, M., Mitchell, W. J., & Purcell, P. (1990). The Electronic Design Studio: Architecture, Media and Knowledge in the Computer Era. MIT Press.
122. Mitchell, W. J. (1990). The Logic of Architecture. MIT Press.
123. Mitchell, W. J., & McCullough, M. (1991). Digital Design Media (1st ed.). Van Nostrand Reinhold Company.
124. Eisenman, P. (1992). Visions Unfolding: Architecture in the Age of Electronic Media. Architectural Design Magazine, 62, xvi–xviii.
125. Lynn, G. (1993). Architectural Curvilinearity: The Folded, the Pliant and the Supple. Folding in Architecture. Architectural Design Magazine, 63(March/April), 8–15.
126. Frazer, J. (1995). An Evolutionary Architecture. Architectural Association Publications.
127. Fox, M., & Kemp, M. (2009). Interactive Architecture. Princeton Architectural Press.
128. Lynn, G. (1999). Animate Form. Princeton Architectural Press.
129. Kolarevic, B. (Eds.). (2003). Architecture in the Digital Age: Design and Manufacturing. Spon Press.
130. Terzidis, K. (2004). **Algorithmic Design: A Paradigm Shift in Architecture?**. Architecture in the Network Society: 22nd eCAADe Conference Proceedings, 201–207.
131. Oxman, R., & Oxman, R. (2014). Theories of the Digital in Architecture. Routledge.
132. Oxman, R. (2017). Thinking difference: Theories and models of parametric design thinking. Design Studies, 1–36.
133. McCormack, J., Dorin, A., & Innocent, T. (2004). **Generative design: a paradigm for design research**. Proceedings of Futureground.
134. Kolarevic, B., & Malkawi, A. (2005). Performative Architecture: Beyond Instrumentality (Vol. 60, Issue 1). Spon Press.
135. Oxman, Rivka. (2008). **Performance-based Design: Current Practices and Research Issues**. International Journal of Architectural Computing, 06(01), 1–17.
136. Speaks, M. (2002a). Design Intelligence. A+U: Architecture and Urbanism, 12(387), 10–18.
137. Speaks, M. (2002b). **Theory was interesting... but now we have work: No hope no fear**. Architectural Research Quarterly, 6(3), 209–212.
138. Oxman, Rivka. (2008). **Digital architecture as a challenge for design pedagogy: theory, knowledge, models and medium**. Design Studies, 29(2), 99–120.
139. Picon, A. (2010). Digital Culture in Architecture. Birkhäuser GmbH.
140. Carpo, M. (2012). The Digital Turn in Architecture 1992-2012. In M. Carpo (Eds.), AD Reader. John Wiley & Sons Ltd.
141. Oxman, R., & Oxman, R (2010). **New Structuralism: Design, Engineering and Architectural Technologies**. The New Structuralism: Design, Engineering and Architectural Technologies. Architectural Design Magazine, 80(04), 14-23.
142. Hensel, M. (2013). AD Primers: Performance-Oriented Architecture: Rethinking Architectural Design and the Built Environment. John Wiley & Sons Ltd.
143. Kolarevic, B. (2005). Towards the performative in Architecture. In B. Kolarevic & A. M. Malkawi (Eds.), Performative Architecture: Beyond Instrumentality (pp. 203–214). Spon Press.
144. Tschumi, B. (1996). Architecture and Disjunction. MIT Press.
145. Kronenburg, R. (2007). Flexible: Architecture that Responds to Change. Laurence King Publishing Ltd.
146. Leatherbarrow, D. (2009). Architecture Oriented Otherwise. Princeton Architectural Press.
147. Picon, A. (2012). Architecture as Performative Art. In Y. Grobman & E. Neuman (Eds.), Performalism: Form and Performance in Digital Architecture (pp. 15–19). Routledge.

148. Goethe, J. W. von. (1790). Die Metamorphose der Pftanzen Zu Erklaren. Carl Wilhelm Ettinger.
149. Thompson, D. (1961). On Growth and Form. Cambridge University Press.
150. Migayrou, F. (2003). The Orders of the Non-Standard: Towards a Critical Structuralism. In Architecture non-Standard. Centre Georges Pompidou Service Commercial.
151. Hensel, M., Menges, A., & Weinstock, M. (2004). Emergence in Architecture. Architectural Design Magazine: Emergence: Morphogenetic Design Strategies, 74(3), 6–9.
152. Menges, A. (2006). **Polymorphism**. Techniques and Technologies in Morphogenetic Design. Architectural Design Magazine, 76(2), 78–87.
153. Jabi, W. (2013). Parametric Design for Architecture. Laurence King Publishing Ltd.
154. Schumacher, P. (2009). **Parametric Patterns**. The Patterns of Architecture. Architectural Design Magazine, 79(6), 28–41.
155. Schumacher, P. (2012). The Autopoiesis of Architecture: A new agenda for architecture - Volume I. John Wiley & Sons Ltd.
156. Picon, A. (2004). **Architecture and the Virtual Towards a new Materiality**? Praxis: Journal of Writing Building, 6, 114–121.
157. Scheurer, F. (2010). **Materialising Complexity**. The New Structuralism: Design, Engineering and Architectural Technologies. Architectural Design Magazine, 80(4), 86–93.
158. Bechthold, M. (2014). Design Robotics: A New Paradigm in Process-Based Design. In R Oxman (Eds.), Theories of the Digital in Architecture. Abingdon: Routledge/Taylor & Francis.
159. Oxman, Rivka. (2012). **Informed tectonics in material-based design**. Design Studies, 33, 427–455.
160. Gramazio, F., & Kohler, M. (Eds.). (2014). **Made by Robots: Challenging Architecture at a Larger Scale**. Architectural Design Magazine, 84(3).
151. Iwamoto, L. (2009). Digital fabrications - Architectural and Material Techniques. Princeton Architectural Press.
162. Willmann, J., Gramazio, F., Kohler, M., & Langenberg, S. (2012). Digital by Material: Envisioning an extended performative materiality in the digital age of architecture. In S. Brell-Cokcan & J. Braumann (Eds.), Robotic Fabrication in Architecture, Art, and Design (pp. 12–27). Springer-Verlag/Wien.
163. Oxman, R. (2012). Novel Concepts in Digital Design. In N. Gu & X. Wang (Eds.), Computational Design Methods and Technologies: Applications in CAD, CAM and CAE Education (pp. 18–33). Information Science reference.
164. Oxman, N. (2011). **Variable property rapid prototyping: Inspired by nature, where form is characterized by heterogeneous compositions, the paper presents a novel approach to layered manufacturing entitled variable property rapid prototyping**. Virtual and Physical Prototyping, 6(1), 3–31.
165. Oxman, N., Ortiz, C., Gramazio, F., & Kohler, M. (2015). **Material ecology**. Computer-Aided Design, 60, 1–2.
166. Beesley, P., Hirosue, S., & Ruxton, J. (2006). Toward Responsive Architectures. In Philip Beesley, S. Hirosue, J. Ruxton, M. Trankle, & C. Turner (Eds.), Responsive Architectures: Subtle Technologies (pp. 3–11). Architectural Press.
167. Oosterhuis, K. (2011). Towards a new kind of building: tag, make, move, evolve. NAi Publishers.
168. Pottmann, H. (2010). **Architectural Geometry as Design Knowledge**. The New Structuralism: Design, Engineering and Architectural Technologies. Architectural Design Magazine, 80(4), 72–77.
169. Andrade, D., Harada, M., & Shimada, K. (2017). **Framework for automatic generation of facades on free-form surfaces**. Frontiers of Architectural Research, 6(3), 273–289.
170. Eigensatz, M., Deuss, M., Schiftner, A., Kilian, M., Mitra, N., Pottmann, H., & Pauly, M. (2010). **Case Studies in Cost-Optimized Paneling of Architectural Freeform Surfaces**. In C. Ceccato, L. Hesselgren, M. Pauly, H. Pottmann, & J. Wallner (Eds.), Advances in Architectural Geometry 2010 (pp. 47–72). Springer.
161. Eigensatz, M., Kilian, M., Schiftner, A., Mitra, N., Pottmann, H., & Pauly, M. (2010). **Paneling Architectural Freeform Surfaces**. ACM Transactions on Graphics, 29(4).
172. Flöry, S., & Pottmann, H. (2010). **Ruled Surfaces for Rationalization and Design in Architecture**. LIFE in:Formation, On Responsive Information and Variations in Architecture: Proceedings of the 30th Annual Conference of the Association for Computer Aided Design in Architecture, 103–109.
173. Fu, C., & Cohen-or, D. (2010). **K-set Tilable Surfaces**. ACM Transactions on Graphics, 29(4), 1–6.
174. Son, S., Fitriani, H., Kim, J. T., Go, S., & Kim, S. (2017). **Mathematical Algorithms of Patterns for Free-Form Panels**. Proceedings of the 2nd World Congress on Civil, Structural, and Environmental Engineering (CSEE'17), 1–8.
175. Larsen, N. M. (2012). **Generative Algorithmic Techniques for Architectural Design**. PhD thesis, Aarhus School of Architecture.
176. Chien, Sheng-fen, Su, H., & Huang, Y. (2015). **PARADE: A pattern-based knowledge repository for parametric designs**. 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.
177. Yu, R., & Gero, J. S. (2015). **An empirical foundation for design patterns in parametric design**. In Y. Ikeda, C. M. Herr, D. Holzer, S. Kaijima, M. J. Kim, & S. M, A (Eds.), Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia, 551–560.
178. Ebel, K.-H., & Ulrich, E. (1987). **Social and Labour Effects of CAD/CAM. IFAC Proceedings Volumes**, 20(5), 291–297.
179. Tamke, M., Nicholas, P., & Zwierzycki, M. (2018). **Machine learning for architectural design: Practices and infrastructure**. International Journal of Architectural Computing, 16(2), 123–143.
180. Belém, C. G., Leitão, A. M., Santos, L., & Leitão, A. M. (2019). **On the Impact of Machine Learning: Architecture without Architects?**. "Hello, Culture": Proceedings of the 18th Computer-Aided Architectural Design Futures Conference, 247–293.

181. Caetano, I., Santos, L., & Leitão, A. (2020). **Computational design in architecture: Defining parametric, generative, and algorithmic design**. Frontiers of Architectural Research, 9(2), 287–300.
182. Alfaris, A. (2009). **Emergence Through Conflict the Multi-Disciplinary Design System (MDDS)**. PhD thesis, Massachusetts Institute of Technology.
183. Knight, T., & Stiny, G. (2015). **Making grammars: From computing with shapes to computing with things**. Design Studies, 41, 8–28.
184. Stiny, G., & March, L. (1981). **Design Machines. Environment and Planning B: Planning and Design**, 8(3), 245–255.
185. Albayrak, C. (2011). **Performative Architecture as a Guideline for Transformation of the Defence Line of Amsterdam**. Master thesis, Middle East Technical University and Delft University of Technology.
186. Peters, B. (2013). **Introduction: Computation Works: The building of algorithmic thought**. Computation Works: The building of algorithmic thought. Architectural Design Magazine, 83(2), 8–15.
187. Cagan, J., Campbell, M. I., Finger, S., & Tomiyama, T. (2005). **A Framework for Computational Design Synthesis: Model and Applications**. Journal of Computing and Information Science in Engineering, 5(3), 171.
188. Humppi, H. (2015). **Algorithm-Aided Building Information Modeling: Connecting Algorithm-Aided Design and Object-Oriented Design**. Master thesis, Tampere University of Technology.
189. Oxman, R. (2006). **Theory and design in the first digital age**. Design Studies, 27(3), 229–265.
190. Otto, F., & Rasch, B. (1996). Finding Form: Towards an Architecture of the Minimal. Edition Axel Menges.
191. Burry, Mark. (1993). The Expiatory Church of the Sagrada Família. Phaidon Press.
192. Moretti, L. (1971). Ricerca Matematica in Architettura e Urbanisticâ. Moebius, IV(1), 30–53.
193. Kalay, Y. E. (1989). Modeling Objects and Environments (Principles of Computer Aided Design). Wiley-Academy.
194. Alfaris, A., & Merello, R. (2008). **The Generative Multi-Performance Design System**. Silicon + Skin: Biological Processes and Computation: Proceedings of the 28th Annual Conference of the Association for Computer Aided Design in Architecture, 448–457.
195. Burry, M. (2003). Between Intuition and process: Parametric Design and Rapid Prototyping. In B. Kolarevic (Eds.), Architecture in the Digital Age: Design and Manufacturing (pp. 149-162.). Taylor&Francis.
196. Nassar, K., Thabet, W., & Beliveau, Y. (2003). **Building assembly detailing using constraint-based modeling.** Automation in Construction, 12(4), 365–379.
197. Barrios, C. (2005). **Transformations on Parametric Design Models: A Case Study on the Sagrada Familia Columns**. Computer Aided Architectural Design Futures: Proceedings of the 11th International Conference on Computer Aided Architectural Design Futures, 393–400.
198. Aish, R., & Woodbury, R. (2005). **Multi-level interaction in parametric design**. Smart Graphics. SG 2005. Lecture Notes in Computer Science, 3638, 151–162.
199. Hernandez, C. (2006). **Thinking parametric design: introducing parametric Gaudi**. Design Studies, 27, 309–324.
200. Eggert, R. J. (2004). Engineering Design. Prentice Hall.
201. Schumacher, P. (2008). **Parametricism as Style: Parametricist Manifesto**. The Darkside Club, 11th Architecture Biennale, Venice.
202. Zboinska, M. A. (2015). **Hybrid CAD/E Platform Supporting Exploratory Architectural Design**. CAD Computer Aided Design, 59, 64–84.
203. Elghandour, A., Saleh, A., Aboeineen, O., & Elmokadem, A. (2016). **Using Parametric Design to Optimize Building's Façade Skin to Improve Indoor Daylighting Performance**. Proceedings of the 3rd IBPSA-England Conference BSO, 353–361.
204. Szalapaj, P. (2001). CAD Principles for Architectural Design. Architectural Press.
205. Eastman, Chuck, Teicholz, P., Sacks, R., & Liston, K. (2008). BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors (1st ed.). John Wiley & Sons, Inc.
206. Marin, P., Blanchi, Y., & Janda, M. (2015). **Cost Analysis and Data Based Design for Supporting Programmatic Phase**. Real Time: Proceedings of the 33rd International Conference on Education and Research in Computer Aided Architectural Design in Europe, Volume 1, 613–618.
207. Monedero, J. (1997). **Parametric Design. a Review and Some Experiences**. Challenges of the Future: Proceedings of the 15th eCAADe Conference.
208. Kensek, K. M., & Noble, D. E. (Eds.). (2014). Building Information Modeling: BIM in Current and Future Practice (1st ed.). John Wiley & Sons.
209. Jabi, W., Soe, S., Theobald, P., Aish, R., & Lannon, S. (2017). **Enhancing parametric design through non-manifold topology**. Design Studies, 52, 1–19.
210. Gerber, David Jason, & Pantazis, E. (2016). **A Multi-Agent System for Facade Design: A design methodology for Design Exploration, Analysis and Simulated Robotic Fabrication**. POSTHUMAN FRONTIERS: Data, Designers, and Cognitive Machines: Proceedings of the 36th Annual Conference of the Association for Computer Aided Design in Architecture, 12–23.
211. Fischer, T., & Herr, C. M. (2001). **Teaching Generative Design**. Proceedings of the 4th International Conference on Generative Art. Generative Design Lab, DiAP, Politechnico di Milano University.
212. Frazer, J., Frazer, J., Xiyu, L., Mingxi, T., & Janssen, P. (2002). **Generative and Evolutionary Techniques for Building Envelope Design**. International Conference on Generative Art, 1–16.

213. Zhang, P., & Xu, W. (2018). **Quasicrystal Structure Inspired Spatial Tessellation in Generative Design**. Learning, Adapting and Prototyping: Proceedings of the 23rd CAADRIA Conference, Volume 1, 143–152.
214. Mitchell, W. J. (1975). **The theoretical foundation of computer-aided architectural design**. Environment and Planning B, 2(2), 127–150.
215. Fasoulaki, E. (2008). **Integrated Design: A Generative Multi-Performative Design Approach**. Master thesis, Massachusetts Institute of Technology.
216. Shea, K., Aish, R., & Gourtovaia, M. (2003). **Towards integrated performance-based generative design tools**. Digital Design: Proceedings of the 21st eCAADe Conference.
217. Chase, S. C. (2005). **Generative design tools for novice designers: Issues for selection**. Automation in Construction, 14(6), 689–698.
218. Leitão, A., Lopes, J., & Santos, L. (2014). **Illustrated Programming**. Design Agency: Proceedings of the 34th Annual Conference of the Association for Computer Aided Design in Architecture, 291–300.
219. Humppi, H., & Österlund, T. (2016). **Algorithm-Aided BIM**. Complexity & Simplicity: Proceedings of the 34th eCAADe Conference, Volume 2, 601–609.
220. Bukhari, F. a. (2011). **A Hierarchical Evolutionary Algorithmic Design (HEAD) System for Generating and Evolving Building Design Models.** PhD thesis, Queensland University of Technology.
221. Bernal, M., Haymaker, J. R., & Eastman, C. (2015). **On the role of computational support for designers in action**. Design Studies, 41, 163–182.
222. Zee, A. v. d., & Vrie, B. d. (2008). Design by Computation.
223. Puusepp, R. (2011). **Generating Circulation Diagrams for Architecture and Urban Design Using Multi-Agent Systems.** PhD thesis, University of East London.
224. Abdelmohsen, S. (2013). **Reconfiguring Architectural Space using Generative Design and Digital Fabrication: A Project Based Course**. Proceedings of the 17th Conference of the Iberoamerican Society of Digital Graphics.
225. Caldas, L. (2008). **Generation of energy-efficient architecture solutions applying GENE_ARCH: An evolution-based generative design system.** Advanced Engineering Informatics, 22(1), 59–70.
226. Chaszar, A., & Joyce, S. C. (2016). **Generating freedom: Questions of flexibility in digital design and architectural computation**. International Journal of Architectural Computing, 14(2), 167–181.
227. Garber, R. (2014). The Digital States and Information Modelling. In R. Garber (Eds.), BIM Design: Realising the Creative Potential of Building Information Modelling (pp. 122–131). John Wiley & Sons Ltd.
228. Terzidis, K. (2003). Expressive Form: A Conceptual Approach to Computational Design. Spon Press.
229. Queiroz, N., Carlos, N., Dantas, N., & Vaz, C. (2015). **Designing a Building envelope using parametric and algorithmic processes**. Proceedings of the 19th Conference of the Iberoamerican Society of Digital Graphics, 797–801.
230. Leitão, A., & Santos, L. (2011). **Programming Languages for Generative Design: Visual or Textual?**. Respecting Fragile Places: 29th eCAADe Conference Proceedings, 549-557.
231. Janssen, P. (2014). **Visual Dataflow Modelling: Some thoughts on complexity**. Fusion: Proceedings of the 32nd eCAADe Conference, Volume 2, 547-556.
232. Kaurel, H. G. (2016). **Easing the Transition from Visual to Textual Programming**. Master thesis, Norwegian University of Science and Technology.
233. Autodesk. (2019). The Dynamo Primer: for Dynamo v2.0.
234. Sutherland, I. E. (1963). **Sketchpad: A man-machine graphical communication system**. PhD thesis, Massachusetts Institute of Technology.
235. Janssen, P., Li, R., & Mohanty, A. (2016). **Möbius: A parametric modeller for the web**. 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, 157–166.
236. Cristie, V., & Joyce, S. C. (2019). **'GHShot': a collaborative and distributed visual version control for Grasshopper parametric programming**. Architecture in the Age of the 4th Industrial Revolution: Proceedings of 37th eCAADe and XXIII SIGraDi Joint Conference, 35–44.
237. Harding, J. E., & Shepherd, P. (2017). **Meta-Parametric Design**. Design Studies, 52, 73–95.
238. Wortmann, T., & Tunçer, B. (2017). **Differentiating Parametric Design: Digital workflows in contemporary architecture and construction**. Design Studies, 52, 173–197.
239. Nezamaldin, D. (2019). **Parametric Design with Visual Programming in Dynamo with Revit: The conversion from CAD models to BIM and the design of analytical applications**. Master thesis, KTH Skolan för arkitektur och samhällsbyggnad.
240. Feist, S., Ferreira, B., & Leitão, A. (2017). **Collaborative Algorithmic-based Building Information Modelling**. Protocols, Flows and Glitches: 22nd International Conference on Computer-Aided Architectural Design Research in Asia, 613–622.
241. Leitão, A., Santos, L., & Lopes, J. (2012). **Programming Languages for Generative Design: A Comparative Study**. International Journal of Architectural Computing, 10(1), 139–162.
242. Celani, G., & Vaz, C. (2012). **CAD Scripting and Visual Programming Languages for Implementing Computational Design Concepts: A Comparison from A Pedagogical Point of View**. International Journal of Architectural Computing, 10(1), 121–138.
243. Noone, M., & Mooney, A. (2018). **Visual and textual programming languages: a systematic review of the literature**. Journal of Computers in Education, 5(2), 149–174.

244. Aish, R. (2011). **DesignScript: Origins, Explanation, Illustration**. Design Modelling Symposium.
245. Castelo-Branco, R., & Leitão, A. (2020). **Visual meets Textual: A Hybrid Programming Environment for Algorithmic Design**. {RE}: Anthropocene - Design in the Age of Humans: Proceedings of the 25th International Conference of the Association for Computer-Aided Architectural Design Research in Asia, Volume 1, 375–384.
246. Sammer, M., Leitão, A., & Caetano, I. (2019). **From Visual Input to Visual Output in Textual Programming**. Intelligent & Informed: Proceedings of the 24th International Conference of the Association for Computer-Aided Architectural Design Research in Asia, Volume 1, 645–654.
247. Lin, C.-J. (2018). **The STG-framework: a pattern-based algorithmic framework for developing generative models of parametric architectural design at the conceptual design stage**. Computer-Aided Design and Applications, 15(5), 653–660.
248. Krull, F. N. (1994). **The Origin of Computer Graphics within General Motors**. IEEE Annals of the History of Computing, 16(3), 40–56.
249. Mutic, P., Moldovan, S., & Moldovan, I. (2010). **Advanced 3D Modeling in ArchiCAD. Basic GDL scripting**. Proceedings of the 8th International Symposium: Computational Civil Engineering 2010, 346–357.
250. Leitão, A., & Proença, S. (2014). **On the Expressive Power of Programming Languages for Generative Design: The Case of Higher-Order Functions**. Fusion: Proceedings of the 32nd eCAADe Conference, Volume 1, 257–266.
251. Tibbits, S., Harten, A. van der, & Baer, S. (2011). Python for Rhinoceros 5.
252. Wortmann, T. (2017). **OPOSSUM: Introducing and Evaluating a Model-based Optimization Tool for Grasshopper**. Protocols, Flows and Glitches: Proceedings of the 22nd International Conference of the Association for Computer-Aided Architectural Design Research in Asia, 283–292.
253. Menges, A. (2015). **Fusing the Computational and the Physical: Towards a Novel Material Culture**. Material Synthesis: Fusing the Physical and the Computational. Architectural Design Magazine, 85(05), 8-15.
254. Dent, A., & Sherr, L. (2014). Material Innovation: Architecture. Thames & Hudson.
255. Fernando, R., Drogemuller, R., Salim, F. D., & Burry, J. (2010). **Patterns, heuristics for architectural design support: Making use of evolutionary modelling in design**. New Frontiers: Proceedings of the 15th CAADRIA Conference, 283–292.
256. Qian, C. Z., Chen, V. Y., & Woodbury, R. F. (2007). **Participant observation can discover design patterns in parametric modeling**. Expanding Bodies: Art, Cities, Environment - Proceedings of the ACADIA 2007 Conference, 230–241.
257. Globa, A., Moloney, J., & Donn, M. (2015). **Urban Codes: Abstraction and Case-Based Approaches to Algorithmic Design and Implications for the Design of Contemporary Cities**. The next City - New Technologies and the Future of the Built Environment: 16th International Conference CAAD Futures, 112–123.
258. Yu, R., & Gero, J. (2015). **Design patterns from empirical studies in computer-aided design**. The next City - New Technologies and the Future of the Built Environment: 16th International Conference CAAD Futures, 527, 493–506.
259. Kilov, H. (2004). **Using RM-ODP to bridge communication gaps between stakeholders**. Workshop on ODP for Enterprise Computing 2004.
260. Naur, P. (1968). Software Engineering. Report on a Conference sponsored by the NATO Science Committee.
261. Gabriel, R. (1996). Patterns of Software: Tales from the Software Community. Oxford University Press.
262. Khwaja, S., & Alshayeb, M. (2013). **Towards design pattern definition language**. Software - Practice and Experience, 43(7), 747–757.
263. Sousa, M., & Paio, A. (2020). **Pattern-driven Design for Small Public Spaces An analysis of pattern books and toolboxes**. Anthropologic: Architecture and Fabrication in the Cognitive Age - Proceedings of the 38th eCAADe Conference, Volume 2, 491–498.
264. Anton, I., & Tănase, D. (2016). **Informed Geometries. Parametric Modelling and Energy Analysis in Early Stages of Design**. Energy Procedia, 85, 9–16.
265. Zuk, W., & Clark, R. H. (1970). Kinetic Architecture. Van Nostrand Reinhold.
266. Ruiz-Geli, E. (2015). It is all about particles. In B. Kolarevic & V. Parlac (Eds.), Building Dynamics: Exploring Architecture of Change (p. 268). Routledge.
267. Tibbits. (2015). Self-Assembly and Programmable Materials. In B. Kolarevic & V. Parlac (Eds.), Building Dynamics: Exploring Architecture of Change (p. 146). Routledge.
268. European Commission (2020). Energy efficiency in buildings. European Commission, Department: Energy. Brussels.
269. Ciardiello, A., Rosso, F., Dell'Olmo, J., Ciancio, V., Ferrero, M., & Salata, F. (2020). **Multi-objective approach to the optimization of shape and envelope in building energy design**. Applied Energy, 280(October).
270. Evins, R. (2013). **A review of computational optimisation methods applied to sustainable building design**. Renewable and Sustainable Energy Reviews, 22, 230–245.
271. UNFCCC, U. N. F. C. on C. C. (1997). Kyoto Protocol.
272. Machairas, V., Tsangrassoulis, A., & Axarli, K. (2014). **Algorithms for optimization of building design: A review**. Renewable and Sustainable Energy Reviews, 31, 101–112.
273. Larson, G. W., & Shakespeare, R. A. (1998). Rendering with Radiance the Art and Science of Lighting Visualization. Morgan Kaufmann Publishers.
274. Marsh, K. (2014). Autodesk Robot Structural Analysis Professional 2015: Essentials. Marsh API LLC.
275. Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning (1st ed.). Addison-Wesley Longman Publishing Co., Inc.

276. Wortmann, T., & Nannicini, G. (2017). Introduction to Architectural Design Optimization. In A. Karakitsiou, A. Migdalas, P. M. Pardalos, & S. Rassia (Eds.), City Networks. Springer Optimization and Its Applications (Issue 128). Springer, Cham.
277. Henriksson, V., & Hult, M. (2015). **Rationalizing Freeform Architecture: Surface discretization and multi-objective optimization**. Master thesis, Chalmers University of Technology.
278. 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.
279. Lin, S.-H., & Gerber, D. J. (2014). **Evolutionary energy performance feedback for design: Multidisciplinary design optimization and performance boundaries for design decision support**. Energy and Buildings, 84, 426–441.
280. Stevanović, S. (2013). **Optimization of passive solar design strategies: A review**. Renewable and Sustainable Energy Reviews, 25, 177–196.
281. Yang, X. S., Koziel, S., & Leifsson, L. (2013). **Computational optimization, modelling and simulation: Recent trends and challenges**. Procedia Computer Science, 18 (International Conference on Computational Science, ICCS 2013 Computational), 855–860.
282. Wortmann, T., & Fischer, T. (2020). **Does architectural design optimization require multiple objectives? A critical analysis**. RE: Anthropocene, Design in the Age of Humans: Proceedings of the 25th International Conference on Computer-Aided Architectural Design Research in Asia, Volume 1, 365–374.
283. Emmerich, M. T. M., & Deutz, A. H. (2018). **A tutorial on multiobjective optimization: fundamentals and evolutionary methods**. Natural Computing, 17, 585–609.
284. Wetter, M. (2001). **GenOpt - a generic optimization program**. Seventh International IBPSA Conference, 601-608.
285. von Buelow, P. (2012). **Paragen: Performative Exploration of Generative Systems**. Journal of the International Association for Shell and Spatial Structures, 53(4), 271–284.
286. Chantrelle, F. P., Lahmidi, H., Keilholz, W., Mankibi, M. El, & Michel, P. (2011). **Development of a multicriteria tool for optimizing the renovation of buildings**. Applied Energy, 88(4), 1386–1394.
287. Cichocka, J. M., Migalska, A., Browne, W. N., & Rodriguez, E. (2017). **SILVEREYE - The Implementation of Particle Swarm Optimization Algorithm in a Design Optimization Tool**. Future Trajectories of Computation in Design: 17th International Conference CAAD Futures 2017 Proceedings, 151–169. Springer Singapore.
288. Rahmani Asl, M., Stoupine, A., Zarrinmehr, S., & Yan, W. (2015). **Optimo: A BIM-based Multi-Objective Optimization Tool Utilizing Visual Programming for High Performance Building Design**. Real Time: Proceedings of the 33rd eCAADe Conference, Volume 1, 673–682.
289. Ma, W., Wang, X., Wang, J., Xiang, X., & Sun, J. (2021). **Generative Design in Building Information Modelling (BIM): Approaches and Requirements**. Sensors, 21(5439).
290. Wortmann, T. (2019). **Genetic evolution vs. function approximation: Benchmarking algorithms for architectural design optimization.** Journal of Computational Design and Engineering, 6(3), 414–428.
291. Pereira, I., & Leitão, A. (2020). **More is more: The no free lunch theorem in architecture**. Proceedings of the International Conference of Architectural Science Association, 765–774.
292. Baños, R., Manzano-Agugliaro, F., Montoya, F. G., Gil, C., Alcayde, A., & Gómez, J. (2011). **Optimization methods applied to renewable and sustainable energy: A review**. Renewable and Sustainable Energy Reviews, 15(4), 1753–1766.
293. Hopfe, C. J., & Hensen, J. L. M. (2011). **Uncertainty analysis in building performance simulation for design support**. Energy and Buildings, 43(10), 2798–2805.
294. Loonen, R., Favoino, F., Hensen, J., & Overend, M. (2016). **Review of current status, requirements and opportunities for building performance simulation of adaptive facades**. Journal of Building Performance Simulation, 10(2), 205–223.
295. Chen, J.-Y., & Huang, S.-C. (2016). **Adaptive building facade optimisation: An integrated Green-BIM approach**. 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, 259–268.
296. Han, T., Huang, Q., Zhang, A., & Zhang, Q. (2018). **Simulation-based decision support tools in the early design stages of a green building-A review**. Sustainability, 10(10).
297. D'Oca, S., Hong, T., & Langevin, J. (2018). **The human dimensions of energy use in buildings: A review**. Renewable and Sustainable Energy Reviews, 81, 731–742.
298. Heiselberg, P., Brohus, H., Hesselholt, A., Rasmussen, H., Seinre, E., & Thomas, S. (2009). **Application of sensitivity analysis in design of sustainable buildings**. Renewable Energy, 34(9), 2030–2036.
299. Pisello, A. L., Castaldo, V. L., Rosso, F., Piselli, C., Ferrero, M., & Cotana, F. (2016). **Traditional and Innovative Materials for Energy Efficiency in Buildings**. Key Engineering Materials, 678, 14–34.
300. Kheiri, F. (2019). **Optimization of building fenestration and shading for climate-based daylight performance using the coupled genetic algorithm and simulated annealing optimization methods**. Indoor and Built Environment, 30(2).
301. Turrin, M., Von Buelow, P., & Stouffs, R. (2011). **Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms**. Advanced Engineering Informatics, 25(4), 656–675.
302. Shi, X. (2010). **Performance-based and performance-driven architectural design and optimization**. Frontiers of Architecture and Civil Engineering in China, 4, 512–518.
303. Schlueter, A., & Thesseling, F. (2009). **Building information model-based energy/exergy performance assessment in early design stages**. Automation in Construction, 18(2), 153–163.

304. Petersen, S., & Svendsen, S. (2010). **Method and simulation program informed decisions in the early stages of building design.** Energy and Buildings, 42(7), 1113–1119.
305. Attia, S., Gratia, E., De Herde, A., & Hensen, J. (2012). **Simulation-based decision support tool for early stages of zero-energy building design**. Energy and Buildings, 49, 2–15.
306. Lin, S. E., & Gerber, D. J. (2013). **Designing-in performance: evolutionary energy performance feedback for early stage design**. Proceedings of Building Simulation 2013: 13th Conference of IBPSA, 386–393.
307. 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.
308. Lin, B., Chen, H., Yu, Q., Zhou, X., Lv, S., He, Q., & Li, Z. (2021). **MOOSAS – A systematic solution for multiple objective building performance optimization in the early design stage**. Building and Environment, 200, 107929.
309. Ampanavos, S., & Malkawi, A. (2022). **Early-Phase Performance-Driven Design using Generative Models**. Design Imperatives: The Future is Now. CAAD Futures 2021. Communications in Computer and Information Science, vol 1465. Springer.
310. López, M., Rubio, R., Martín, S., & Croxford, B. (2017). **How plants inspire facades. From plants to architecture: Biomimetic principles for the development of adaptive architectural envelopes**. Renewable and Sustainable Energy Reviews, 67, 692–703.
311. Ochoa, C., & Capeluto, I. (2009). **Advice tool for early design stages of intelligent facades based on energy and visual comfort approach**. Energy and Buildings, 41(5), 480–488.
312. Bouchlaghem, N. (2000). **Optimizing the design of building envelopes for thermal performance**. Automation in Construction, 10(1), 101–112.
313. Wang, W., Zmeureanu, R., & Rivard, H. (2005). **Applying multi-objective genetic algorithms in green building design optimization**. Building and Environment, 40(11), 1512–1525.
314. Gagne, J., & Andersen, M. (2012). **A generative facade design method based on daylighting performance goals**. Journal of Building Performance Simulation, 5(3), 141–154.
315. Gagne, J. M. L., & Andersen, M. (2010). **Multi-Objective Optimization for Daylighting Design Using a Genetic Algorithm**. Proceedings of SimBuild 2010 - 4th National Conference of IBPSA-USA, 9.
316. Ko, W. H., Schiler, M., Kensek, K., & Simmonds, P. (2012). **Tilted Glazing in Building Simulations and its Effect on Form-refinement of Complex Facades**. Proceedings of SimBuild Conference 2012: 5th National Conference of IBPSA-USA, 361–368.
317. Kasinalis, C., Loonen, R., Cóstola, D., & Hensen, J. (2014). **Framework for assessing the performance potential of seasonally adaptable facades using multi-objective optimization**. Energy and Buildings, 79, 106–113.
318. Jin, Q., & Overend, M. (2014). **A prototype whole-life value optimization tool for façade design**. Journal of Building Performance Simulation, 7(3), 217–232.
319. Gamas, A., Konis, K., & Kensek, K. (2014). A Parametric Fenestration Design Approach for Optimizing Thermal and Daylighting Performance in Complex Urban Settings. 43rd ASES National Solar Conference 2014, 87–94.
320. Pantazis, E., & Gerber, D. (2018). **A framework for generating and evaluating façade designs using a multi-agent system approach**. International Journal of Architectural Computing, 16(4), 248–270.
321. Gerber, D.., Pantazis, E., & Wang, A. (2017). **A multi-agent approach for performance-based architecture: Design exploring geometry, user, and environmental agencies in façades**. Automation in Construction, 76, 45–58.
322. Bertagna, F., D'Acunto, P., Ohlbrock, P. O., & Moosavi, V. (2021). **Holistic Design Explorations of Building Envelopes Supported by Machine Learning**. Journal of Facade Design and Engineering, 9(1), 31–46.
323. Laing, R. (2019). Digital Participation and Collaboration in Architectural Design. Routledge: Taylor & Francis Group.
324. Self, J. A. (2019). **Communication through design sketches: Implications for stakeholder interpretation during concept design**. Design Studies, 63, 1–36.
325. Station, S. S., & Street, S. (2006). **The new Southern Cross Station: The iconic redevelopment of Melbourne's Spencer Street Station**. Steel Australia, 11–14.
326. Rothenthal, G., Ziegler, R., & Spasic, D. (2018). Oasis of light – Manufacturing the cladding of the Louvre Abu Dhabi. In L. Hesselgren, A. Kilian, S. Malek, K.-G. Olsson, O. Sorkine-Hornung, & C. Williams (Eds.), AAG 2018: Advances in Architectural Geometry (pp. 274–285). Klein Publishing GmbH (Ltd.).
327. Imbert, F., Frost, K. S., Fisher, A., Witt, A., Tourre, V., & Koren, B. (2012). Concurrent Geometric, Structural and Environmental Design: Louvre Abu Dhabi. In L. Hesselgren, S. Sharma, J. Wallner, N. Baldassini, P. Bompas, & J. Raynaud (Eds.), Advances in Architectural Geometry 2012 (pp. 77–90). Springer Vienna.
328. Piermarini, E., Nuttall, H., May, R., Janssens, V. M., Manglesdorf, W., & Kelly, T. (2018). **Morpheus Hotel, Macau – a paradigm shift in computational engineering**. Steel Construction, 11(3), 218–231.
329. Blandini, L., & Nieri, G. (2020). **Kuwait International Airport Terminal 2: Engineeirng and Fabrication of a Complex Parametric Megastructure**. In J. Burry, J. Sabin, B. Sheil, & M. Skavara (Eds.), FABRICATE 2020: Making Resilient Architecture (pp. 84–91). UCLPress.
330. Kolarevic, B. (2003). Information Master Builders. In Architecture in the Digital Age: Design and Manufacturing (pp. 57–62). Spon Press.
331. Austern, G., Capeluto, I. G., & Grobman, Y. J. (2018). **Rationalization Methods in Computer Aided Fabrication: A critical review**. Automation in Construction, 90, 281–293.

332. Hill, J. (2005). **Building the Drawing**. Design through Making. Architectural Design Magazine, 75(4), 13–21.
333. Austern, G., Elber, G., Capeluto, I. G., & Grobman, Y. J. (2018). Adapting Architectural Form to Digital Fabrication Constraints. In L. Hesselgren, A. Kilian, S. Malek, K.-G. Olsson, O. Sorkine-Hornung, & C. Williams (Eds.), AAG 2018: Advances in Architectural Geometry (pp. 10–33). Klein Publishing GmbH (Ltd.).
334. Overall, S., Rysavy, J. P., Miller, C., Sharples, W., Sharples, C., Kumar, S., Vittadini, A., & Saby, V. (2020). **Direct-to-Drawing: Automation in extruded terracotta fabrication**. In J. Burry, J. Sabin, B. Sheil, & M. Skavara (Eds.), FABRICATE 2020: Making Resilient Architecture. UCLPress.
335. Castañeda, E., Lauret, B., Lirola, J. M., & Ovando, G. (2015). **Free-form Architectural Envelopes: Digital processes opportunities of industrial production at a reasonable price**. Journal of Facade Design and Engineering, 3(1), 1–13.
336. Lee, G., & Kim, S. (2012). **Case Study of Mass Customization of Double-Curved Metal Façade Panels Using a New Hybrid Sheet Metal Processing Technique**. Journal of Construction Engineering and Management, 138(11), 1322–1330.
337. Aksamija, A. (2016). AD Smart04: Integrating Innovation in Architecture - Design, methods and technology for progressive practice and research. John Wiley & Sons Ltd.
338. Soar, R., & Andreen, D. (2012). **The Role of Additive Manufacturing and Physiomimetic Computational Design for Digital Construction**. Material Computation: Higher Integration in Morphogenetic Design. Architectural Design Magazine, 82(2), 126–135.
339. Paio, A., Eloy, S., Rato, V. M., Resende, R., & de Oliveira, M. J. (2012). **Prototyping Vitruvius, New Challenges: Digital Education, Research and Practice**. Nexus Network Journal: Architecture and Mathematics, 14, 409–429
340. Jančič, L. (2016). Implications of the Use of Additive Manufacturing in Architectural Design. PhD Thesis. Univerza v Ljubljani.
341. Kolarevic, B. (2008). The (Riscky) Craft of Digital Making. In Manufacturing material effects: Rethinking design and making in architecture. Routledge.
342. Afify, H. M. N., & Elghaffar, Z. (2007). **Advanced Digital Manufacturing Techniques (CAM) in Architecture**. Em'Body'Ing Virtual Architecture: The 3rd International Conference of the Arab Society for Computer Aided Architectural Design, 67–80.
343. Bayram, A. K. Ş. (2021). Digital Fabrication Shift in Architecture. In Architectural Sciences and Technology (Vol. 42, pp. 173–193). Livre de Lyon.
344. Barkow F (2008). Cut to Fit. In B. Kolarevic & K. Klinger (Eds.), Manufacturing Material Effects: Rethinking Design and Making in Architecture. Routledge.
345. Suare, R. (2008). Innovation Through Accountability in the Design and Manufacturing of Material Effects. In B. Kolarevic & K. Klinger (Eds.), Manufacturing Material Effects: Rethinking Design and Making in Architecture. Routledge.
346. Simmons, M. (2008). Material Collaborations. In B. Kolarevic & K. Klinger (Eds.), Manufacturing Material Effects: Rethinking Design and Making in Architecture. Routledge.
347. Kolarevic, B., & Klinger, K. R. (2008). Manufacturing / Material / Effects. In B. Kolarevic & K. Klinger (Eds.), Manufacturing Material Effects: Rethinking Design and Making in Architecture. Routledge.
348. Peters, B., & Peters, T. (2013). Mind the Gap: Case Stories of Exchange. In AD Smart01: Inside Smartgeometry (pp. 206–217). John Wiley & Sons, Ltd.
349. Mesnil, R., Douthe, C., Baverel, O., Léger, B., & Caron, J. F. (2015). **Isogonal moulding surfaces: A family of shapes for high node congruence in free-form structures**. Automation in Construction, 59, 38–47.
350. Hesselgren, L., Charitou, R., & Dritsas, S. (2007). **The Bishopsgate Tower Case Study**. International Journal of Architectural Computing, 5(1), 61–81.
351. Whitehead, H. (2003). Laws of form. In B. Kolarevic (Eds.), Architecture in the Digital Age: Design and Manufacturing (pp. 116–148). Spon Press - Taylor & Francis Group.
352. Pottmann, H., Eigensatz, M., Vaxman, A., & Wallner, J. (2015). **Architectural geometry**. Computers and Graphics, 47, 145–164.
353. Ceccato, C. (2012). **Material Articulation: Computing and Constructing Continuous Differentiation**. Material Computation: Higher Integration in Morphogenetic Design. Architectural Design Magazine, 82(2), 96–103
354. Bates, D. (2008). Different Differences. In B. Kolarevic & K. Klinger (Eds.), Manufacturing Material Effects: Rethinking Design and Making in Architecture. Routledge.
355. Sharples, C. (2009). **Unified frontiers: Reaching out with BIM**. Closing the Gap. Architectural Design Magazine, 79(2), 42–47.
356. Schittich, C., & Lenzen, S. (Eds.). (2015). Fassaden/Facades: Best of Detail. Herausgeber.
357. Romero, F., & Ramos, A. (2013). **Bridging a culture: The design of museo soumaya**. Computation Works: The buliding of algorithmic thought. Architectural Design Magazine, 83(2), 66–69.
358. Leon, A. P. De. (2012). **Two Case-Studies of Freeform-Facade Rationalization**. Digital Physicality: Proceedings of the 30th eCAADe Conference, Volume 2, 491–500.
359. Beorkrem, C. (2013). Material Strategies in Digital Fabrication. Routledge.
360. Brownell, B. (2012). Material Strategies: Innovative Applications in Architecture. Princeton Architectural Press.
361. Bennett, C., Fabrizio, M., & Kehoe, M. (2011). 100 11th Avenue: Ateliers Jean Nouvel.
362. Caetano, I., & Leitão, A. (2016). **DrAFT: an Algorithmic Framework for Facade Design**. Complexity & Simplicity: Proceedings of the 34th eCAADe Conference, Volume 1, 465–474.

363. Leitão, A. (2014). **Improving generative design by combining abstract geometry and higher-order programming**. Rethinking Comprehensive Design: Speculative Counterculture - Proceedings of the 19th International Conference on Computer- Aided Architectural Design Research in Asia, 575–584.
364. Barendregt, H., & Barendsen, E. (1994). Introduction to Lambda Calculus.
365. Coxeter, H. S. M. (1973). Regular Polytopes (3rd edition). Methuen, Dover Publications.
366. Grünbaum, B., & Shephard, G. (1977). **Tilings by Regular Polygons**. Mathematics Magazine, 50(5), 227–247.
367. Kaplan, C., & Salesin, D. (2004). **Islamic star patterns in absolute geometry**. ACM Transactions on Graphics, 23(2), 97–119.
368. Nasri, A., Benslimane, R., & El, A. (2017). **Geometric rosette patterns analysis and generation**. Journal of Cultural Heritage, 25, 65–74.
369. Grünbaum, B., & Shephard, G. (1987). Tilings and patterns. New York : W.H. Freeman.
370. Emery, I. (2009). The Primary Structures of Fabrics: An Illustrated Classification (4th edition). Thames & Hudson.
371. Grünbaum, B., & Shephard, G. (1980). **Satins and Twills: An Introduction to the Geometry of Fabrics**. Mathematics Magazine, 53(3), 139–161.
372. Grünbaum, B., & Shephard, G. (1986). **An extension to the catalogue of isonemal fabrics**. Discrete Mathematics, 60, 155–192.
373. Jiang, J., & Ma, Y. (2020). **Path planning strategies to optimize accuracy, quality, build time and material use in additive manufacturing: A review**. Micromachines, 11(7).
374. Evins, R., Joyce, S. C., Pointer, P., Sharma, S., Vaidyanathan, R., & Williams, C. (2012). **Multi-objective design optimisation: Getting more for less**. Proceedings of the Institution of Civil Engineers: Civil Engineering, 165(5), 5–10.
375. Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). **A fast and elitist multiobjective genetic algorithm: NSGA-II**. IEEE Transactions on Evolutionary Computation, 6(2), 182–197.
376. Carlucci, S., Cattarin, G., Causone, F., & Pagliano, L. (2015). **Multi-objective optimization of a nearly zero-energy building based on thermal and visual discomfort minimization using a non-dominated sorting genetic algorithm (NSGA-II)**. Energy and Buildings, 104, 378–394.
377. Barraza, M., Bojórquez, E., Fernández-González, E., & Reyes-Salazar, A. (2017). **Multi-objective optimization of structural steel buildings under earthquake loads using NSGA-II and PSO**. KSCE Journal of Civil Engineering, 21(2), 488–500.
378. Nabil, A., & Mardaljevic, J. (2006). **Useful daylight illuminances: A replacement for daylight factors**. Energy and Buildings, 38(7), 905–913.
379. Shapiro, A. (2003). Monte Carlo Sampling Methods. In Handbooks in Operations Research and Management Science (pp. 353–425). Elsevier.
380. McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). **A Comparison of Three Methods for Selecting Value of Input Variables in the Analysis of Output from a Computer Code**. Technometrics, 21(2), 239–245.
381. Rytel, G. (2013). The Influence of Climate on the Forms of Brazilian Modernist Architecture in the Years 1925-1960. Kwartalnik Architektury I Urbanistyki, 4, 57–78.
382. Sousa, S., & Batista-Bastos, M. (2015). O tempo e a Diferença: Análise e Readaptação num Edifício em Lisboa. Cadernos de Arquitectura e Urbanismo, 22(33), 58.
383. Erhan, H., Wang, I. Y., & Shireen, N. (2015). **Harnessing design space: A similarity-based exploration method for generative design**. International Journal of Architectural Computing, 13(2), 217–236.