http://periodicos.sbu.unicamp.br/ojs/index.php/parc
https://doi.org/10.20396/parc.v11i0.8657350

GENERATING NURBS CLADDING AND STRUCTURES WITH PARAMETRIC PROGRAMMING AND BIM

GERANDO VEDAÇÕES E ESTRUTURAS EM NURBS UTILIZANDO PROGRAMAÇÃO PARAMÉTRICA E BIM

Neander Furtado Silva [1]
University of Brasília, DF, Brazil, neander@unb.br

Lilian Maciel Furtado Silva [2]
University of Brasília, DF, Brazil, lilianmfs91@gmail.com

Igor Lacroix [3]
University of Brasília, DF, Brazil, igorlacroixarquitetura@gmail.com

Abstract

The process of designing and building curvilinear architectures is still challenging. The use of multiple applications with distinctive design paradigms is unlikely to disappear. The interoperability used here was not only the conventional one. It was also 'live,' in 'real-time', with two of the applications involved opened and running simultaneously. A design workflow based on the use of form-forming applications connected via parametric programming to building information modeling, BIM, was proposed. The main objective was to facilitate designing and building curvilinear architectures and their supporting structures simultaneously using two different design paradigms. The tools needed in our research can be summarized as follows: NURBS Lofting for surface creation, contouring for modular slicing and structural axis grid definition, sweeping along axes for surface creation of the curved beams of I-profile and paneling for the subdivision of curved surfaces into planar fractions. Parametric programming was used to automate sweeping along axes to generating curved I-beams and paneling to subdivide the NURBS surfaces into planar fractions. To the best of our knowledge, our major contribution resides in defining a workflow and developing new algorithms for facilitating designing NURBS surfaces and corresponding supporting structures through 'live' interoperability among different applications.

Keywords: Parametric programing. NURBS. Cladding. Structures. BIM.

Resumo

Os processos de projetação e construção de arquiteturas curvilíneas são ainda desafiadores. O uso de múltiplos aplicativos baseados em paradigmas distintos provavelmente não irá desaparecer. A interoperabilidade utilizada aqui foi não apenas a convencional. Foi também 'ao vivo', em tempo real, com os aplicativos abertos e rodando simultaneamente. Um fluxo de projeto baseado no uso aplicativos orientados para a criação da forma conectados via programação paramétrica e BIM foi proposto. O principal objetivo foi facilitar a projetação e construção de arquiteturas curvilíneas e suas estruturas de suporte utilizando simultaneamente dois diferentes paradigmas de projeto. As ferramentas ou operações necessárias nesta pesquisa foram essencialmente as seguintes: NURBS Lofting para a criação da superfície, contouring para o fatiamento modular da superfície e definição dos eixos geradores da estrutura em grelha, varredura ao longo de eixo para a criação de vigas perfil I curvas e panelização para a subdivisão das superfícies curvas em frações planares para viabilizar a fabricação e construção. No melhor do nosso conhecimento, a principal contribuição desta pesquisa reside na definição e desenvolvimento de um fluxo de trabalho e no desenvolvimento de novos algoritmos para facilitar a projetação de superfícies NURBS e suas estruturas de suporte correspondentes por meio de interoperabilidade em tempo real entre diferentes aplicativos.

Palavras-chave: Programação paramétrica. NURBS. Vedações. Estruturas. BIM.

How to cite this article:

SILVA, N. F.; SILVA, L. M. F.; LACROIX, ÍGOR. Gerando vedações e estruturas em NURBS utilizando programação paramétrica e BIM. **PARC Pesquisa em Arquitetura e Construção**, v. 11, p. e020016, 17 ago. 2020. DOI:https://doi.org/10.20396/parc.v11i0.8657350

Received in 30.10.2019 - accepted in 18.05.2020 – published 17.08.2020

Introduction

The concepts of parametric models and parametric programming are not always clearly stated in many discussions, often leading to misunderstanding and misinformation. For this reason, it is essential to provide a clear understanding of the meaning attributed to those terms in this paper.

It is important to notice that the parametric concept is quite old and precedes by several centuries the advent of modern computing. It goes back to at least Ancient Greece. The Renaissance studies of the Greek orders, particularly those involving the proportions of columns and their smaller components, are remarkable examples of early parametric representation. They frequently show drawings of classical architectural elements in which dimensions do not have fixed values. Instead, they show expressions defining these values as functions of other variables such as the column's diameter (TZONIS; LEFAIVRE, 1986, p. 48-51).

The notion of parametric models is best understood by contrasting their representational properties with those that fall outside this category. Kolarevic describes some of those distinctive features::

> *Parametrics can provide for a powerful conception of architectural form by describing a range of possibilities, replacing in the process stable with variable, singularity with multiplicity…In parametric design, it is the parameters of a particular design that are declared, not its shape. By assigning different values to the parameters, different objects or configurations can be created. (KOLAREVIC, 2003, p. 17)*

One straightforward way to understand Kolarevic assertions and what a parametric model is and its implications is as a non-parametric object is represented with fixed properties. For example, the dimensions of this object would be defined by fixed numbers. This object may be a rectangle measuring 2 x 4 meters. There are no variables in its representation. Besides, these dimensions are everything that is known or registered about this object. Its dimensions are presented in an unrelated way, that is, nowhere there is a record telling that the longest side of the rectangle is twice the size of the shortest one.

On the order hand, if a rectangle is represented by related variables, such as H for height and L for length, then, a proportion may be attributed through the formulas L=Hx2 or H=L/2. However simplistic or naive this example may seem, the consequences of such type of representation are tremendous: one may derive from this an infinite number of rectangles, all of them following the same proportions parameters.

Parametric representations are not restricted to the relations between an object's dimensions. A vast range of things and properties may be parametrized. An example of parametrization that may look foreigner to an architect, but it is ubiquitous nowadays is the spreadsheet. If we contrast it to a simple table of values, this last is found to be static. No variations can be produced from it. However, from a spreadsheet, with each cell programmed, an infinite number of instances can be produced (WOODBURRY, 2010, p. 11).

Relations among objects and their parts can also be parameterized, such as distances, angles, and various rules as attached to, parallel to or distance from (EASTMAN et al., 2008, p. 29).

Parametric should not be confused with the concept of building information modeling (EASTMAN et al., 2008), BIM. While all BIM systems must be parametric, not all parametric systems fit into a BIM description. There are different types of parametrization in architecture.

We argue that BIM systems parametrize objects according to a construction-function-oriented paradigm. This leads to the representation of objects such as slabs, columns, beams, walls, doors, windows, etc.

On the other hand, most parametric systems that do not fit into a BIM description are based on a form-forming-process paradigm. This paradigm uses form creation processes such as extrusion, revolving, twisting, tapering, bulging, morphing, rounding, sweeping, NURBS lofting, subdivision, triangulating, etc.

We also argue that neither of these two paradigms is better than the other. It is not the case that one provides less information than the other. Sometimes it is assumed that form-forming-process systems lack something, while construction-function-oriented systems have everything, which is not true. They contain different information.

Both paradigms are essential for designing and building the forms we will describe later in this paper. Indeed, none of the mainstream architectural design applications is based solely on one of these two paradigms. Systems that were originally based predominantly in one paradigm have gradually, though often modestly, incorporated resources that are based on the other one.

Parametric programming (TERZIDIS, 2009, p.19-22) may be understood as the design process that takes place using parametric representations and its properties' controls as nodes in propagation-based systems (AISH; WOODBURY, 2005; WOODBURY, 2010, p. 12-22). Some authors use parametric programming as a synonym to parametric modeling or parametric design (WOODBURY, 2011, p. 12-15). However, for the sake of clarity, we will use only parametric programming in this paper.

Visual programming is a very important implementation of parametric programming, with interfaces directly derived from graph theory. The Rhinoceros-Grasshopper programming environment is one of them, and we will describe how it was used in the experiment reported in this paper.

We want to stress that parametric models and parametric programming should not be confused with parametricism. This expression was coined by Patrik Schumacher and meant a new architectural style in succession to the post-modernism, the modernism, the art nouveau, the historicism, the baroque, etc. (SCHUMACHER, 2011, p. 35).

This might suggest that the use of parametric programming leads necessarily to one style. However, this sounds restricting, since parametric techniques can lead to greater diversity, not to conversion. Also, any architectural language can be modeled in a parametric system, not just the contemporary ones. However, we will not go deeper into this discussion, but only stress that we do not use parametric programming as a synonym to parametricism.

Another important concept used in this paper is the algorithm. Berlinski uses a simple way to define what is an algorithm:

> *An algorithm is a finite procedure, written in a fixed symbolic vocabulary, governed by precise instructions, moving in discrete steps, 1,2,3, ... , whose execution requires no insight, cleverness, intuition, intelligence or perspicuity, and that, sooner or later, comes to an end. (BERLINSKI, page 9, 2000)*

An algorithm is computer language independent. Although an algorithm is the product of some sort of programming is not supposed to be confused with computer programing. Programing is understood here as the instantiation of one or more algorithms into a specific computer language, be either a command-line one, such as Java, Phyton, C++, etc. or a visual language such as Grasshopper from McNeel or Dynamo from Autodesk.

Research problem

The process of designing and building curvilinear architectures (KRAUEL et al., 2010) is still challenging (KOLAREVIC, 2003, p. 6-7). Digital technologies have contributed to turning those processes more feasible, but no single application is suitable for all design and building tasks involved. The use of multiple applications with distinctive design paradigms is a common practice, and it is unlikely to disappear. We recognize the relevance of integration into standalone applications, but society's diverse nature and software development make a unified platform undesirable. Therefore, our research focuses on augmenting interoperability to facilitate designing and build curvilinear architectures.

The design process often moves gradually from a high level of ambiguity into a high level of specification (GOEL, 1995, p. 193-195). Curvilinear architectural forms are better produced in systems providing a high degree of ambiguity, where the relationship between what is known and what can be produced is loosely defined. This results in flexibility and significant freedom. On the other hand, as the design moves into a higher specification, the stricter the rules must become, reducing the range of options and increasing the level of detail. Therefore, the two paradigms described earlier in the introduction, the form-forming one and the BIM systems, are essential for designing and building the curvilinear architectures.

In many cases, curvilinear architectures are continuous. Their structure and behavior do not fit into categories such as columns, beams, and slabs, as predominantly found in BIM systems. Although BIM systems have gradually introduced some form-forming oriented tools, these are still rudimentary.

On the other hand, BIM systems represent building components according to their construction function. This representation allows the straightforward creation of construction drawings at different levels of detail. Those remain consistent with the three-dimensional model whenever changes are made during the design process. It also provides several other types of information, such as bills of quantity that facilitate rigorous construction management. BIM systems change key processes involved in putting a building together (KHEMLANI, 2011).

Therefore, we arrive at our research questions. If the use of applications based on two different design paradigms is necessary, is there a way to significantly improve interoperability between them? How can this scheme of interoperability be described and implemented? How could it be used to design a curvilinear architecture and its supporting structure?

Hypothesis

The scientific method’s hypothetical-deductive nature has already been extensively demonstrated (POPPER, 1980, p. 74-94). Therefore, it is assumed as a given presupposition, and it will not be discussed here.

We believe the simultaneous use of a form-forming oriented application and a BIM system, connected through parametric programming, in a ‘live’ or ‘real time’ interoperability, may improve the design process and construction curvilinear architecture and their supporting structures. We believe that this improvement will be expressed through a greater level of automation in the processes of form-forming and their direct translation into BIM elements.

Objectives

The main objective was to facilitate designing and building curvilinear architectures and their supporting structures using simultaneously two design paradigms connected via parametric programming.

The specific objectives were:

a) To improve the interoperability between a form-forming application and a BIM system by implementing a 'live' or 'real-time' connection.

b) To develop and test a specific set of prototypical algorithms for designing a curved cladding surface and its supporting structure.

Research method

The type of method used here was predominantly problem-solving research with some testing-out research (PHILLIPS; PUGH, 1987, p. 45).

Within this framework, prototyping (SOMMERVILLE, 2011, p. 44-46, 53, 109, 111) had a leading rule. Prototyping is what Woodbury (2010, p. 36-37) referred to as 'throw code away.' Professional programmers in the software industry seek a stable code that can be distributed safely and repeatedly reused as it is. On the other hand, designers do design, not media. In each design project, they tend to rebuild rather than reuse. If designers use programming, they do not seek the stable code at each new design, but they rather copy, paste, and modify previous ones or start completely from scratch.

Therefore, the research described here, and its experimental procedure relied significantly on writing a 'throw-away code' and observing its immediate results as designed objects. A 'throw-away code' does not mean something that will not be ever useful in a different situation. It simply means that it will not be used without modification and adaptation. Modification and incremental changes are at the core of any design process (LAWSON, 2005, p. 197-198).

We argue that interoperability is an important design media, not the objective. Since interoperability will result in various communication and data transfer issues, a design workflow based on the use of parametric programming to connect a form-forming application and building information modeling, BIM (Eastman et al., 2008) was proposed for prototyping and testing.

The type of interoperability used here was not only the conventional one, involving the asynchronous use of two applications at different times with data being transferred via mutually readable files. The interoperability used here was also 'live,' in 'real-time', with two of the applications involved opened and running simultaneously.

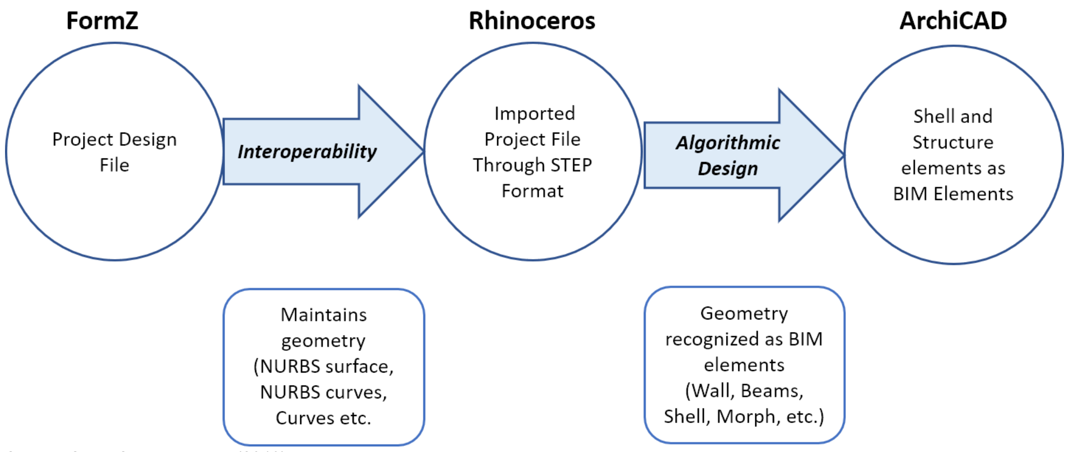
In this paper, we describe a design experiment requiring the development of a curvilinear roof. The curvilinear roof was represented as a surface based on a set of Non-Uniform Rational B-Splines, NURBS (MITCHELL; MCCULLOUGH, 1995, p. 193-196, 199-200). This surface was generated by sweeping curved profiles along a path also curved.

Our starting point was three commercial applications: FormZ (AUTODESSYS INC., 2020), Rhinoceros-Grasshopper (ROBERT MCNEEL & ASSOCIATES, 2020) and ArchiCAD (GRAPHISOFT, 2020). Central to our research was the Grasshopper-ArchiCAD Live

Connection. The connection between FormZ and Rhinoceros is not live, and the FormZ file was imported to Rhinoceros. In an asynchronous interoperability scheme, the use of a third application was purely pragmatical to speed up the design process due to our research time constraints and some limitations of Rhinoceros in some specific tools. However, the connection Rhinoceros-Grasshopper-ArchiCAD is 'live,' simultaneous and bi-directional. This connection means that any change in the Rhino-Grasshopper model results in changes in the ArchiCAD model and vice-versa.

A workflow involving FormZ-Rhinoceros-Grasshopper-ArchiCAD interoperability was proposed (Figure 1).

Figure 1 - Interoperability chosen path based on the existing path


FormZ
Rhinoceros
ArchiCAD
Project Design File
Interoperability
Imported Project File Through STEP Format
Algorithmic Design
Shell and Structure elements as BIM Elements
Maintains geometry (NURBS surface, NURBS curves, Curves etc.
Geometry recognized as BIM elements (Wall, Beams, Shell, Morph, etc.)

Source: Silva, Silva e Lacroix (2019).

The interoperability between FormZ and Rhinoceros was achieved through a mutually recognizable format. The interoperability in Rhinoceros-Grasshopper-ArchiCAD was in real-time and helped transform abstract geometry into recognized BIM elements.

At this point, it is important to establish a distinction with other research developed in recent years that differ from what is proposed here. Those researches deal with different issues and objectives. Ashour and Kolarevic (2015) test multiple solutions within a design workflow with the limitations of the software used to explore the best performance and quality of results. However, the software used is different and does not include BIM.

Plotnikov (2016) integrates Grasshopper with GIS, instead of a BIM system to bridge the gap between the physical and digital design methodologies.

Khalili-Araghi and Kolarevic (2016) used a plug-in within Revit. However, their goal is a standalone platform rather than interoperability.

Sharah used programming in Rhino-Grasshopper using UV Mesh and Quadrangulation (SHARAH, 2017). However, the objective is visualizing the modeled object in a virtual reality headset.

On an urban scale, Fink (2019) investigates the potential towards a holistic, digital, urban design process aimed at the development of a practical methodology using Rhinoceros 3D and Grasshopper.

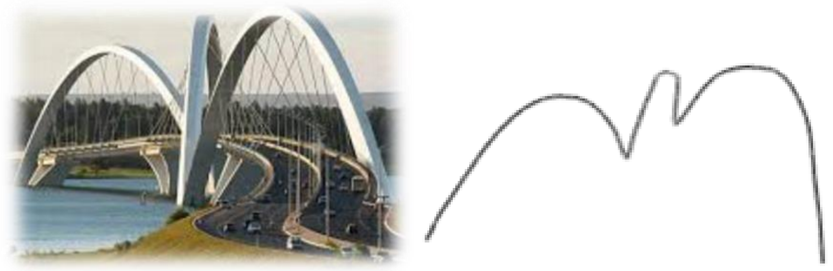
Therefore, these authors have not experimented with a workflow like the one we proposed above, enabling the use of an external file imported into Rhinoceros and connected to ArchiCAD via Grasshopper.

Experimental procedures

The form-finding method used in defining this project's shape was inspired by that used in the Church of St. Francis of Assisi of Pampulha, Brazil, and in the Riverside Museum in Glasgow, UK, but with important modifications that we will describe next. The geometric and computational technique used was that of a NURBS type surface ruled by variable profiles along its path.

Our initial source of inspiration in defining our project's shape was silhouette profiles of the JK bridge in Brasilia, Brazil, viewed at a particular angle, as shown in the following Figure 2.

Figure 2 - JK Bridge and initial profile



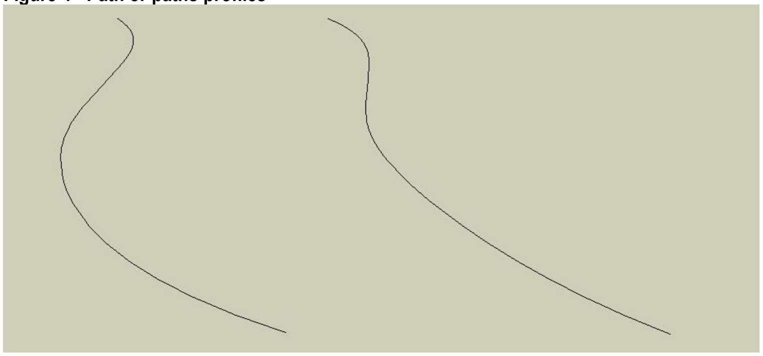
Source: the authors.

We adopted as initial profile as the silhouette shown in Figure 2.

This silhouette was slightly modified and reshaped throughout the design process. The initial profile represented a silhouette obtained at a significantly obtuse angle. Our final design's silhouette is a more discreet reference with a more lateral viewing angle of the bridge and softened sinuosity.

Regarding the paths that constrain the path's profile, we defined two very smooth, undulated curves shown in figure 4 below.

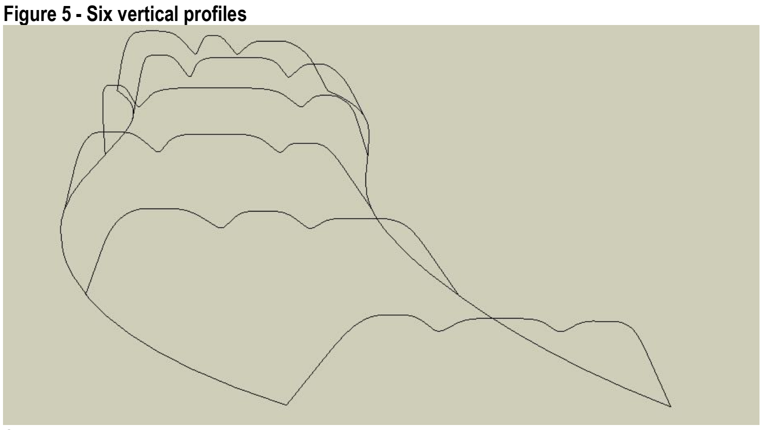
Figure 4 - Path or paths profiles



Source: the authors.

Initially, there was only a single curved covering surface of our project. Later, we realized the need to split the roof surface into three parts: two opaque surfaces, one on each side of the building, and one translucent central. This decision made it possible to obtain a higher incidence of natural light through the translucent central cover. Figure 5 below shows the building's lateral boundary paths and the six vertical profiles defining the roof surface.

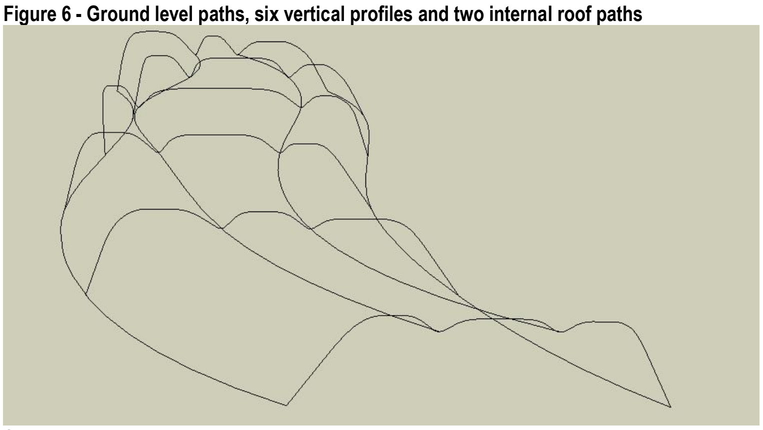
Figure 5 - Six vertical profiles



Source: the authors.

Due to the need to split the roof into three parts, two more paths above the floor were set at a high height to secure the translucent central part's inner boundaries. Figure 6 below shows the two floor-level side paths, the six vertical path profiles, and the two roof-level inner paths.

Figure 6 - Ground level paths, six vertical profiles and two internal roof paths



Source: the authors.

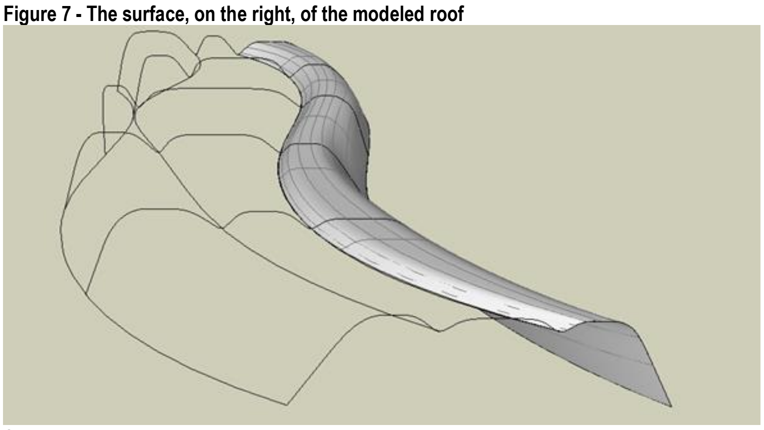
Figure 7 below shows, in addition to the elements mentioned above, the surface of the right part of the roof. Figure 8-a shows the two opaque surfaces of the roof. Finally, Figure 8-b shows the three parts of the roof.

Our project is like defining the technical shape of the Church of St. Francis of Assisi in Pampulha. Both designs use a variable shape curved profile along the swept path. The formal differences, however, varied. First, only in one part of the Church of St. Francis of Assisi does the profile vary. Besides, the path is always straight. In our project, we used multiple profiles while at St. Francis of Assisi, there was only one at the beginning and the end.

Our design resembles the Riverside Museum's form-finding technique in that both used a NURBS like surface ruled by varying profiles along their course. However, in addition to creating a significantly different number of profiles (there are 12 in Riverside and only 6 in our project), our profiles are diverse.

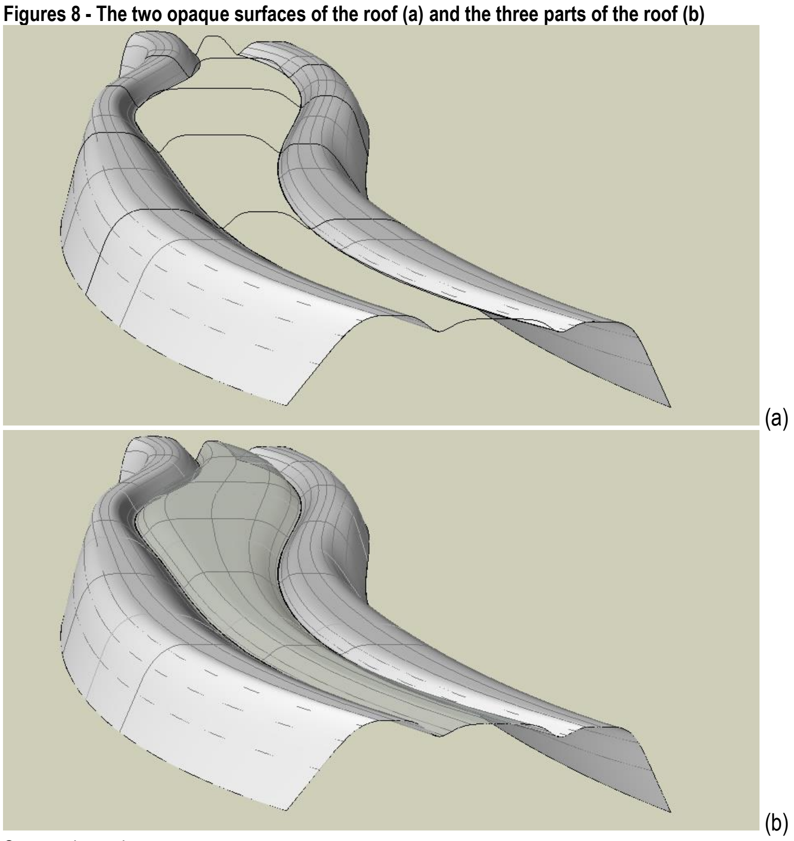
On Riverside, all profiles are topologically identical; they are composed of the same number of straight segments and points. In our project, the segments are not topologically identical. Also, our project profiles are true curves, whereas, in Riverside, they are straight-line sequences. Finally, our project's roof is made up of three adjoining contiguous surfaces, whereas Riverside's is monolithic and hermetically sealed on the top.

Figure 7 - The surface, on the right, of the modeled roof



Source: the authors.

Figures 8 - The two opaque surfaces of the roof (a) and the three parts of the roof (b)



Source: the authors.

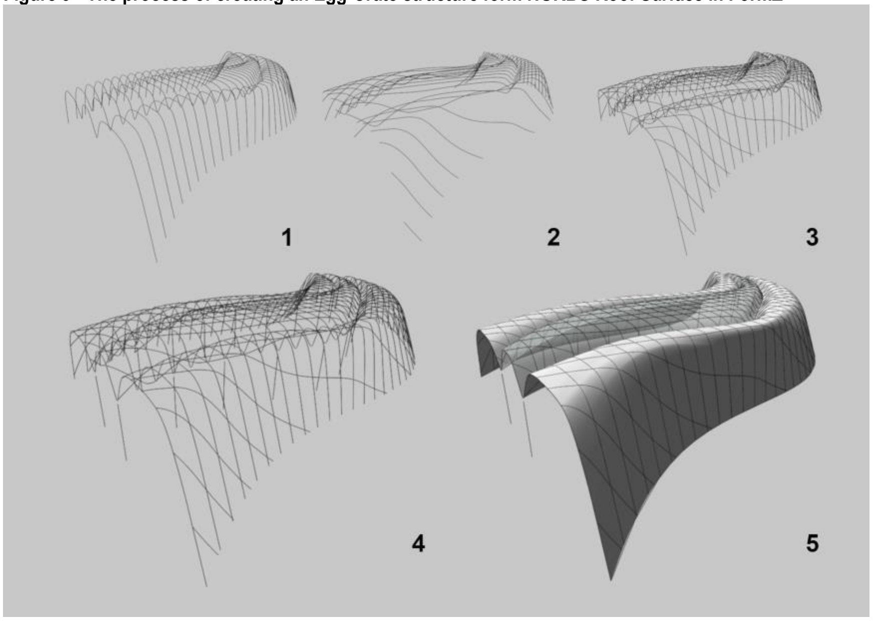
Considering the large spans of the proposed project, a structural solution was adopted in some respects, similar to those used in the Memorial Darcy Ribeiro in Brasilia (by João Filgueiras Lima) and the Experience Music Project in Seattle (by Frank Gehry). In other words, I-beams (with web and lower and upper flanges) have been proposed that continuously propagate under the cladding surface following its curvature.

Unlike the Darcy Ribeiro Memorial, where the structure is radially distributed, and the Experience Music Project, where the grid's distribution is not always regular, this project adopted a regular spacing of 2 meters in two perpendicular directions. The I-beams profiles have a 35 centimeters high web and 21 centimeters wide in the bottom and top flanges.

We designed a surface for testing our hypothesis. The surface was initially modeled in FormZ because of its easy to use graphical interface without resorting to programming. Due to certain shortcomings and lack of Rhinoceros' intuitiveness, particularly regarding direct graphical modeling, we adopted in the early stages of design development the application FormZ. A NURBS surface was created to form the roof and cladding.

Firstly, A surface was created from six contours with a NURBS lofting tool. The cladding was modeled with a 30 cm offset below the roof. The axes contours of the modeled structural beams were implemented using the contour tool to create an egg-crate structure. Figure 9 illustrates part of this process.

Figure 9 - The process of creating an Egg-Crate structure form NURBS Roof Surface in FormZ



Source: Silva, Silva e Lacroix (2019).

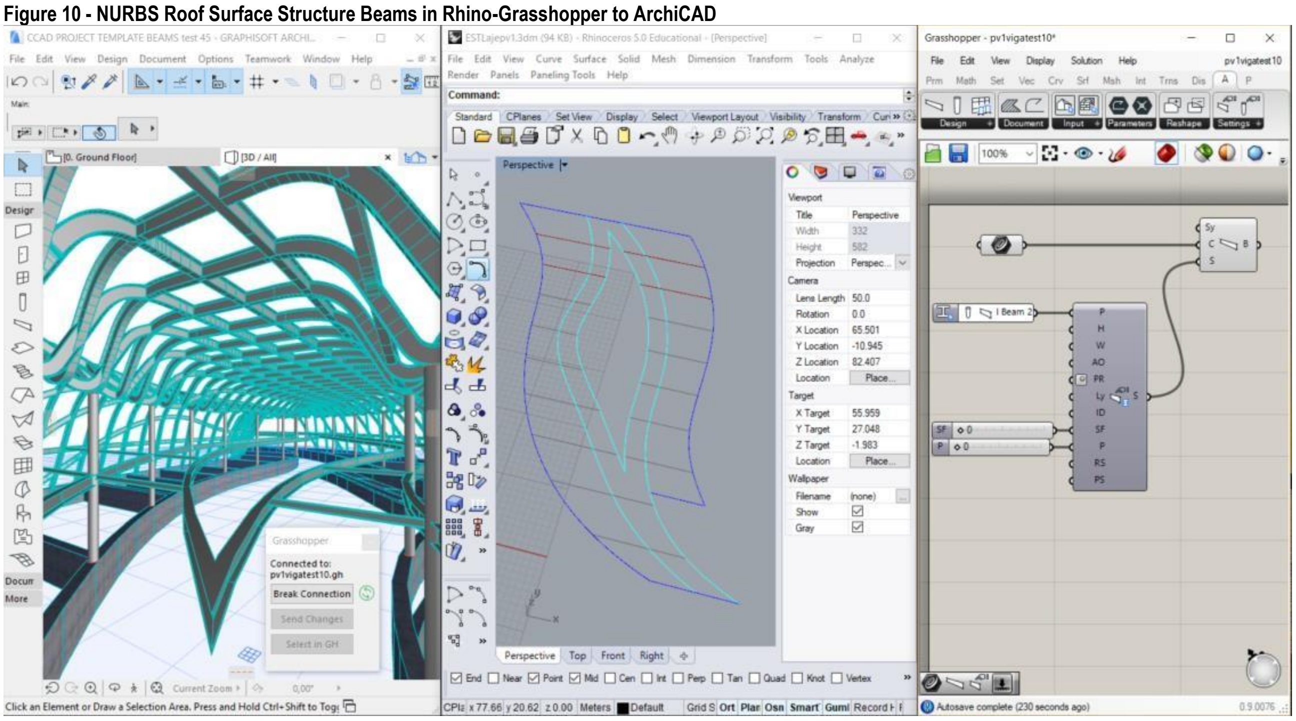
The project file was then exported to Rhinoceros through a STEP file. This format maintains the object's original geometry properties.

The main reason for choosing FormZ to define the surface was that it would have taken more steps and time to create the surface in Rhinoceros graphic window. For example, it took approximately four steps in FormZ to model it, while in Rhino, it would have taken seven steps.

Again, we would like to stress that using multiple applications in the same design process is a very common situation in practice that must be dealt with as a research problem. Designers must and do resort to different applications, considering the best tool for each design task at hand.

Secondly, the roof, cladding, and structural axes imported from FormZ had to be further developed. The Grasshopper programming and the Rhino-Grasshopper-ArchiCAD Live Connection were used for this purpose. This connection recognizes the Grasshopper generated objects as real BIM elements in ArchiCAD. One example is the steel beams adopted in this design experiment, which had an I profile shape. Figure 10 illustrates one of these processes.

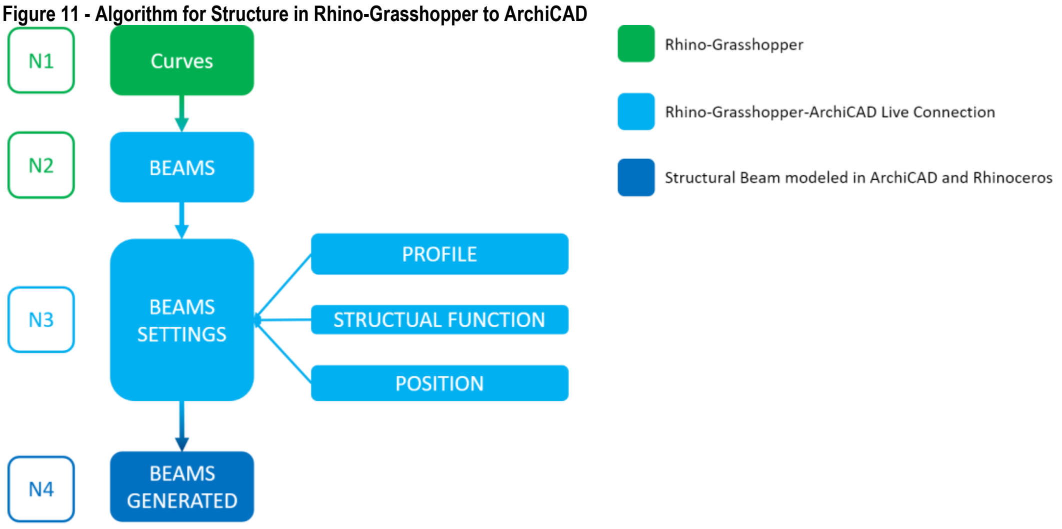
Figure 10 - NURBS Roof Surface Structure Beams in Rhino-Grasshopper to ArchiCAD


CCAD PROJECT TEMPLATE BEAMS test 45 - GRAPHISOFT ARCHI...
[0. Ground Floor]
[3D / All]
Grasshopper
Connected to: pv1vigatest10.gh
Break Connection
Send Changes
Select in GH
ESTLajepv1.3dm (94 KB) - Rhinoceros 5.0 Educational - [Perspective]
Perspective
Top
Front
Right
Grasshopper - pv1vigatest10*
I Beam 2
Autosave complete (230 seconds ago)

Source: Silva, Silva e Lacroix (2019).

Thirdly, algorithms were elaborated in Grasshopper for generating steel beams from the imported axes. Structural components were created by sweeping the I profile through the axes imported from FormZ. Figure 11 provides a general overview of the programming produced in Grasshopper for generating the structural beams. The authors created the algorithms shown here on Grasshopper's interface with the tools and connections made available by Robert McNeel & Associates (Rhinoceros-Grasshopper) and Graphisoft (ArchiCAD). They were used on one notebook computer only. No other computer was used in this study, nor were there changes in hardware.

Figure 11 - Algorithm for Structure in Rhino-Grasshopper to ArchiCAD

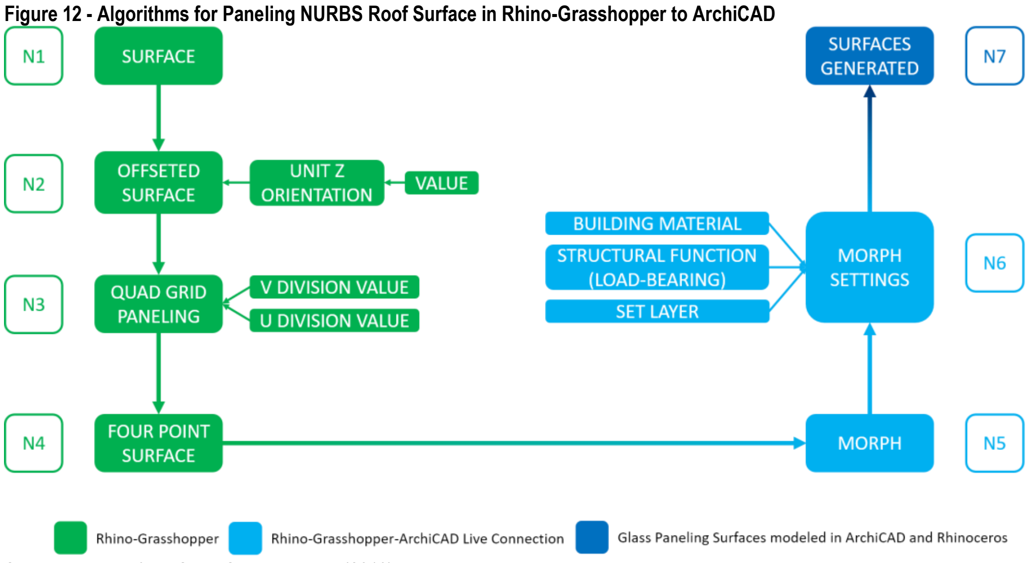


Source: adapted from Silva, Silva e Lacroix (2019).

Node 1 (N1), shown in Figure 11, represents structural contours/curves used as a path to generate the beams along with them in node 2. A node 2 (N2) procedure creates a beam in ArchiCAD defined by curve from the selected structural contours/curves contained in node 1. Node 3 (N3) allows the user to define beams settings. It was used as a complex I-beam shape. This node contains setting properties of the type of profile beam to be selected and defines the beams' structural function to be specified; 0 (zero) for load-bearing or 1 for non-load bearing. A load-bearing structure was specified, and the user-defined height position of the beam structure about information stored in N1. This height position is defined by 0 (zero) for exterior (above) or 1 for interior (below). In this case, it was specified 0 (zero) for the exterior. Node 4 (N4) is the result of the algorithm generating the curvilinear beams.

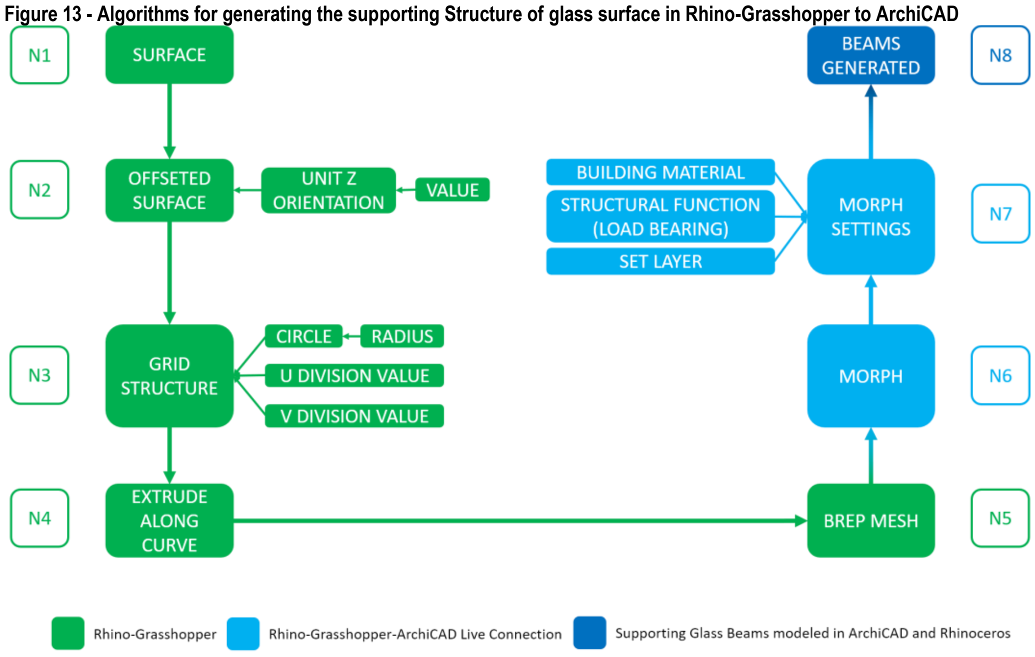
Fourthly, an algorithm was applied to the imported surfaces to generate panels of approximately two by two meters with Grasshopper plug-in LunchBox and Rhino-Grasshopper-ArchiCAD Live Connection. The curved surface was divided into subfractions of four-sided coplanar panels to make the construction feasible. Figures 12 and 13 illustrate in more detail the programming involved in these processes.

Figure 12 - Algorithms for Paneling NURBS Roof Surface in Rhino-Grasshopper to ArchiCAD



Source: adapted from Silva, Silva e Lacroix (2019).

Figure 13 - Algorithms for generating the supporting Structure of glass surface in Rhino-Grasshopper to ArchiCAD



Source: adapted from Silva, Silva e Lacroix (2019).

In the Rhinoceros-Grasshopper interface, the nodes/boxes represent the surface (shown in Figures 7-9) and the procedures. Node 3 (N3) places points on the surface and connects the points with lines forming polygons. Node 4 separates the polygons dividing it into a mesh UV mapping. This separates the surface into smaller ones. Once installed, the Rhino-Grasshopper-ArchiCAD Plugin/Connection, a new set of nodes/boxes options within the tab ARCHICAD, appears.

These new nodes receive the information from the Mesh UV mapping algorithm of Grasshopper through node 6 in the diagram shown in Figure 12. Node 5 (N5) collects the information of smaller surfaces as Morph surfaces in ArchiCAD with the defined Morph Properties Settings in Node 6 (N6). Then it translates, in this case, a geometry only object into a Morph surface with BIM Components properties. In this research, this included the material properties of Zinc Roof panels. The node 6 (N6), Morph settings, contains information of building material user-defined, specification of in which layer the Morph Surface will be stored, and specification of a load non-load-bearing structure. Node 7 (N7) is the Glass roof surface generated by an algorithm created by the authors. A similar algorithm was used for generating the supporting structure of the glass cladding surface, as shown in Figures 7-9.

Node 1 (N1) stores the glass surface selected by the user. Node 2 (N2) generates an offset surface by defining the offset Z-axis direction and distance value it offsets from the original surface. Node 3 (N3) defines the offset surface's grid structure, contains information to define the U and V grid division respectively, and tubular profile shape of the grid structure. The supporting structure for the glass surface was composed of tubular parts. Node 4 (N4) extrudes the circular profile along structural grid lines. Node 5 (N5) generates the tubular Mesh surface based on the extrusion produced in 4. Node 6 (N6) receives Mesh information from 5 and sends it to ArchiCAD. Node 7 (N7), called Mesh settings, contains user-defined properties such as building material, layer for information stored and non-load, and load-bearing structural function. Node 8 (N8) results from the algorithm of creating load-bearing tubular beams structure to support the glass roof.

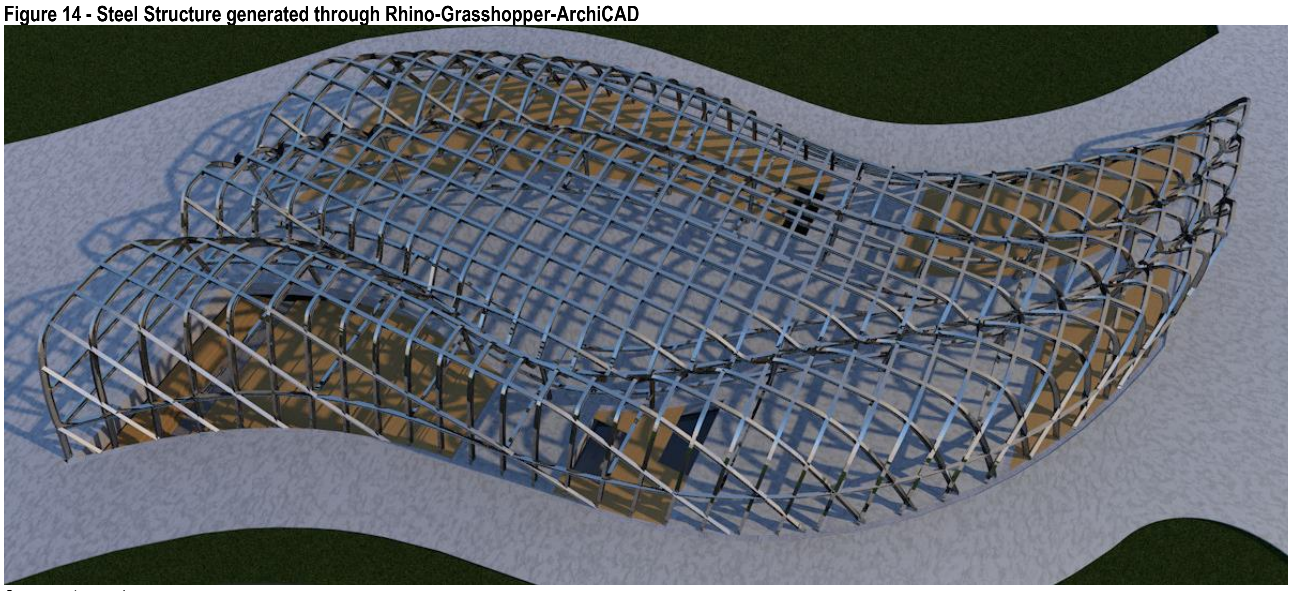
Results and observation

The beams were generated in ArchiCAD directly as BIM elements from the profile axis imported into Rhino from FormZ. The total time spent to generate the elements between finishing programming in Grasshopper and seeing ArchiCAD's results was 12 minutes and 48 seconds. The proposed workflow, that is, FormZ-Rhinoceros-Grasshopper-ArchiCAD connection, made it possible the desired flexibility at the beginning of the design process, as well as precision and speed in the detailed specification of the final designed building. The use of a programming interface such as Grasshopper was essential to create detailed structural beams and cladding panels and translate the elements generated in Rhinoceros as BIM recognized components in ArchiCAD. The completion of the process in a BIM system also made possible the precise extraction of construction drawings. The representation of the project as a 3D model in a BIM system was would also make it possible that the production of 2D nesting cutting plans, but this was beyond the scope of the research described here.

Figures 14 and 15 below show aerial views of the steel structure and metallic cladding for the roof generated through Rhino-Grasshopper-ArchiCAD Live Connection.

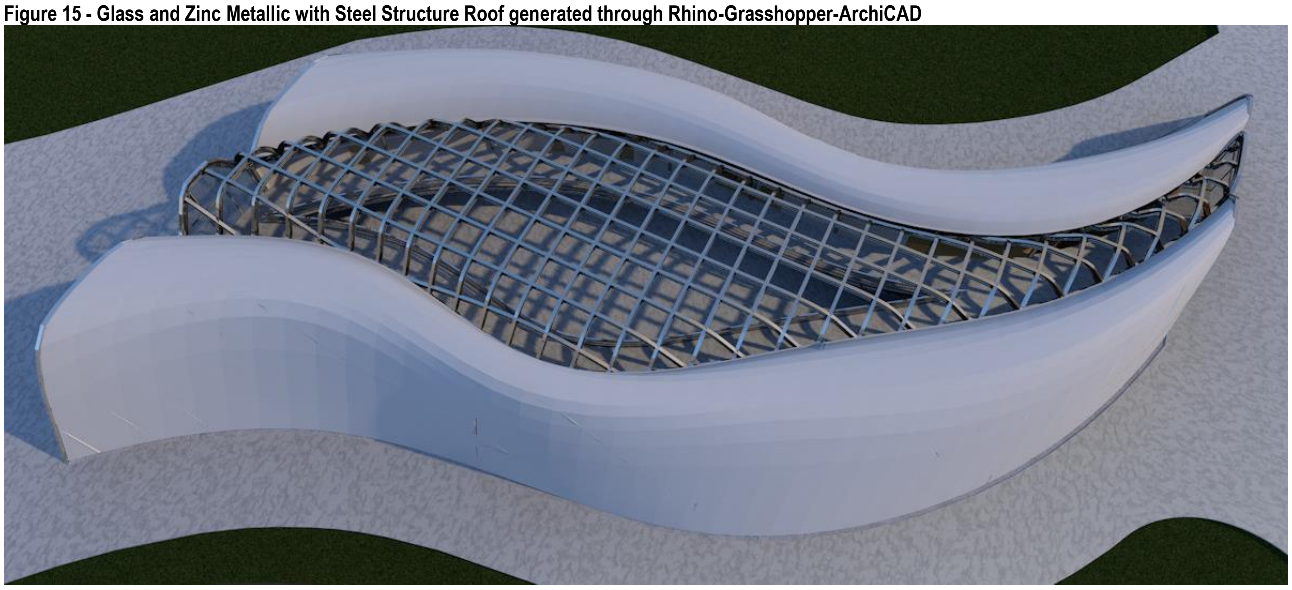
Figures 16 and 17 show partial views of construction drawings produced from the BIM model in ArchiCAD. These views are partly due to the limitations of file size.

Figure 14 - Steel Structure generated through Rhino-Grasshopper-ArchiCAD



Source: the authors.

Figure 15 - Glass and Zinc Metallic with Steel Structure Roof generated through Rhino-Grasshopper-ArchiCAD

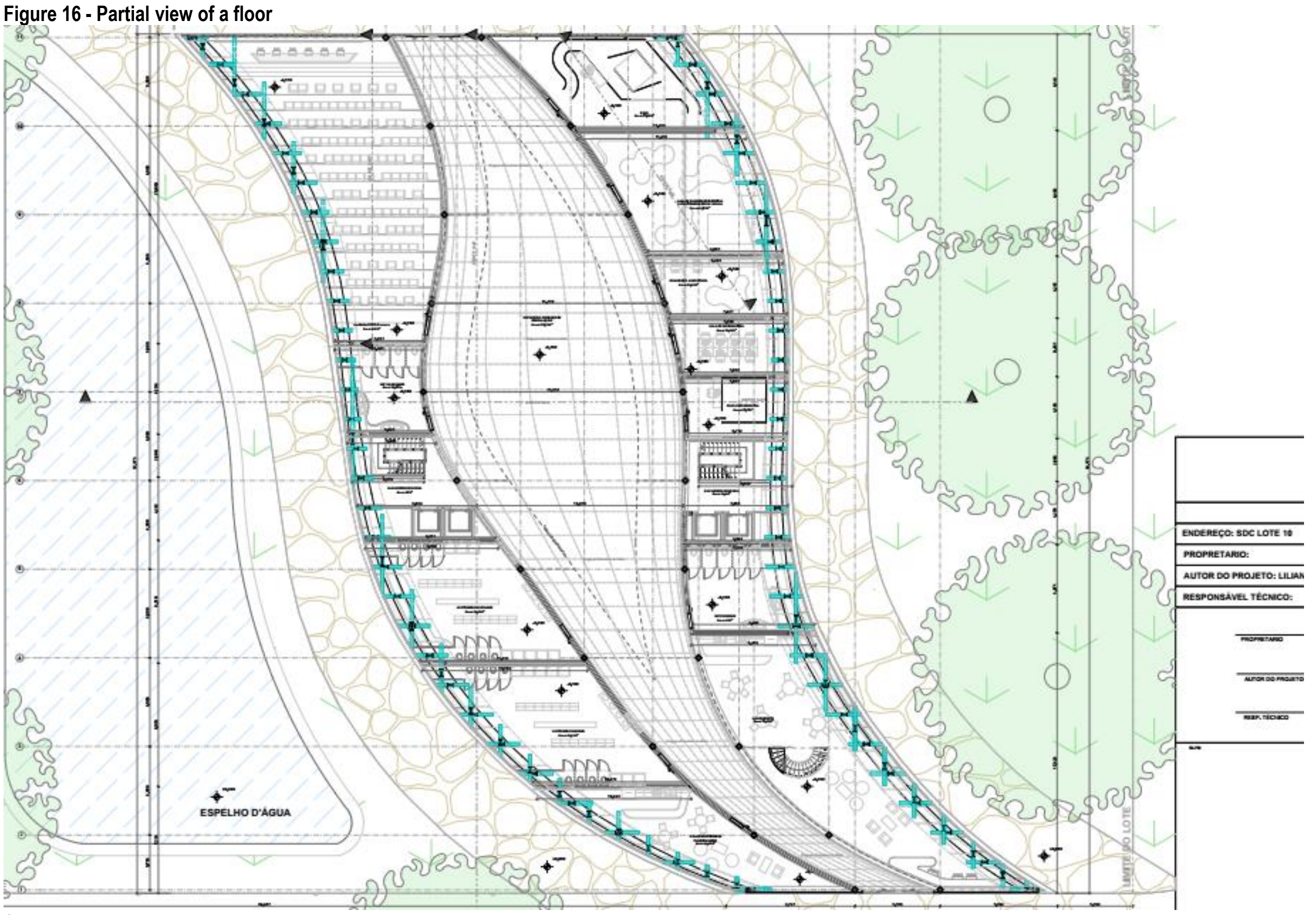


Source: the authors.

Discussion

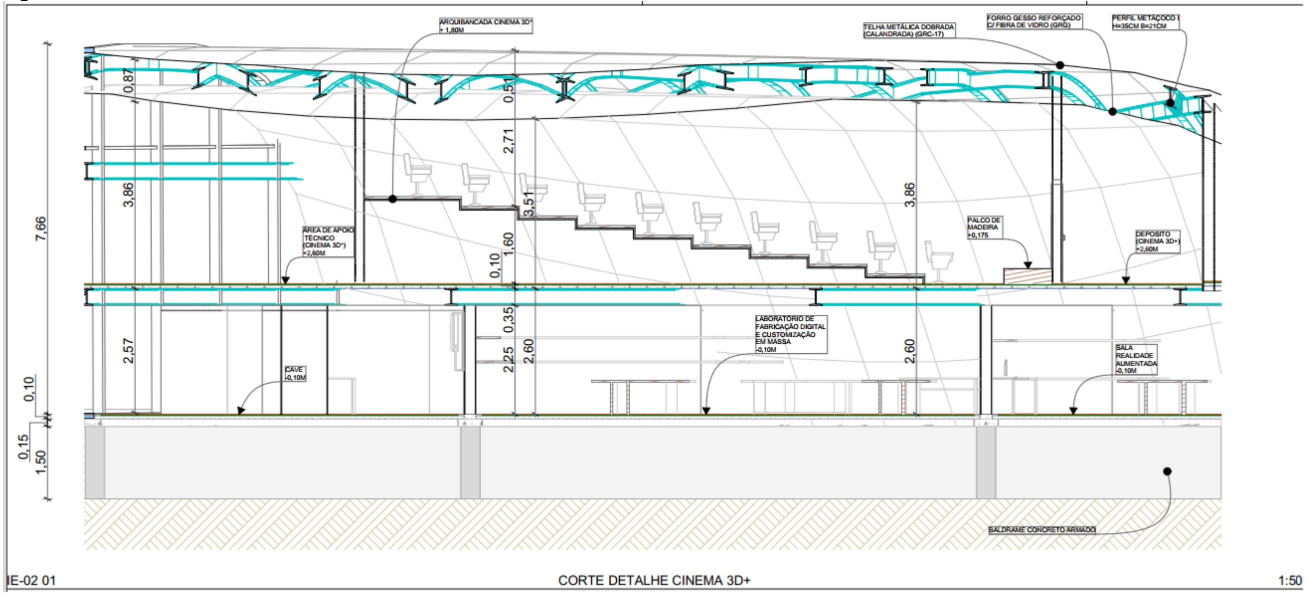
The ‘live’ connection between Grasshopper and ArchiCAD (Grasshopper-ArchiCAD Live Connection) was the most important resource in this research. The connection between FormZ and Rhinoceros was not ‘live’ or in real-time, but it was based on the conventional interoperability, i.e., non-simultaneous. A program is used to generate a file in native format, which converts to another format to be exported to another program where it should be recognized. The program that generates the file is then closed, and the destination application opened. The file exported by the former is opened and re-saved in another native format by the destination application. There, this new file will continue to be developed, modified, and often reworked to recover missing or lost data in the conventional interoperability transfer process. Although the decision to use FormZ was initially pragmatical as explained earlier, it also allowed to contrast with the ‘live’ interoperability used in the rest of the proposed workflow and better perceive its advantages.

Figure 16 - Partial view of a floor



Source: the authors.

Figure 17 - Partial view of a section


ARQUIBANCADA CINEMA 3D+ +1,80M
TELHA METÁLICA DOBRADA (CALANDRADA) (GRC-17)
FORRO GESSO REFORÇADO C/ FIBRA DE VIDRO (GRG)
PERFIL METAÇOCO I H=35CM B=21CM
ÁREA DE APOIO TÉCNICO (CINEMA 3D+) +2,60M
PALCO DE MADEIRA +0,175
DEPOSITO (CINEMA 3D+) +2,60M
LABORATORIO DE FABRICAÇÃO DIGITAL E CUSTOMIZAÇÃO EM MASSA -0,10M
CAVE -0,10M
SALA REALIDADE AUMENTADA -0,10M
BALDRAME CONCRETO ARMADO
7,66
3,86
2,57
0,87
0,51
2,71
3,51
1,60
0,10
0,35
2,25
2,60
0,15
1,50
IE-02 01
CORTE DETALHE CINEMA 3D+
1:50

Source: the authors.

The interoperability provided by Rhinoceros-Grasshopper-ArchiCAD Live Connection represents considerable progress because it is simultaneous, real-time, and bidirectional. This connection means that both programs are open and remain open simultaneously and that any model change in the Rhino-Grasshopper platform results in the model change in ArchiCAD. This type of real-time interoperability has been central to this research and is our future research projects.

In this research, it was possible to develop an architectural project of curvilinear and complex form that required a series of computational programming procedures that would make it constructively feasible. This study was an experiment through which it was possible to observe several gaps or deficiencies in the existing computational tools.

Despite the various tools available, the process of creating such an architecture and structure is still very laborious, involving many manual activities that increase the probability of propagating inefficiency, inaccuracy, and errors. The integration of existing tools through further programming would bring greater security to the design process of complex curvilinear buildings by reducing such risks.

In this research project, the file containing the NURBS surfaces and structural axes was then exported to Rhinoceros using the STEP format. This format retained the original geometric and topological properties of the object. In this set of procedures, it was possible to observe a reduced integration level between the different operations needed to create the shapes, with a significant number of tasks that needed to be done manually, particularly in the transition between one tool and another.

Although the procedures for creating the NURBS surfaces and the axes profiles for the egg-crate structure are slightly more automated in FormZ than in Rhinoceros graphic workspace, in both cases it involves a significant degree of manual tasks, such as choosing the surface, modulating slicing, and orienting it to generate structural profiles. The manual tasks were one of the shortcomings identified and could be reduced by creating new algorithms in Grasshopper. The algorithm would link in the continuous workflow the operations as mentioned earlier and increase the level of automation in Rhinoceros without resorting to FormZ or other platforms that do not support 'live' interoperability with a BIM system.

Conclusion

To the best of our knowledge, our major contribution resides in defining a workflow and creating a set of new algorithms, as modifiable and adaptable codes, for designing NURBS surfaces and the corresponding supporting structures through enhancing the interoperability by using parametric programming. For this reason, we believe our hypothesis is promising, and we achieved our objectives.

Applying FormZ into the existing workflow during the conceptual design process helped create more fluid forms within an easier interface. Although it does not offer the programming capabilities of Rhinoceros-Grasshopper, FormZ provided a clear understanding of the set of tools needed to realize the intended building design. It helped to see at an individual level the types of tools that could be integrated into a single workflow in the future by programming for 'live' interoperability with a BIM system.

The interoperability provided by the Rhinoceros-Grasshopper-ArchiCAD connection helped in solving the issues with geometry not recognized as BIM elements such as the axis as a path for an I-beam profile structure.

The tools needed in our research can be summarized as follows: NURBS Lofting for surface creation, contouring for modular slicing and structural axis grid definition, sweeping along axes for surface creation of the curved beams of I-profile and paneling for the subdivision of curved surfaces into planar fractions. Therefore, future research should focus on integrating these existing tools in Rhinoceros, in a continuous workflow, through the programming of new algorithms implemented through

Grasshopper, raising the automation of the surface and structures production described above.

As future research, the algorithms developed here could be further improved for their specific application in other situations by modifying and adapting code to create structural curved I-beams and automate paneling of curved surfaces. These could result in specific plugins within Grasshopper.

Note

This article is an extended version of the article "INTEGRATING PARAMETRIC MODELING WITH BIM THROUGH GENERATIVE PROGRAMMING FOR THE PRODUCTION OF NURBS SURFACES AND STRUCTURES" (SILVA; SILVA; LACROIX, 2019) by the authors presented in e 24th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) in 2019 in Wellington, New Zealand.

References

AISH R., WOODBURY R. Multi-level Interaction in Parametric Design. In: BUTZ A., FISHER B., KRÜGER A., OLIVIER P. (eds) **Smart Graphics. SG 2005 Lecture Notes in Computer Science**, Berlin, Heidelberg, v. 3638, p. 151-162, 2005. DOI:https://doi.org/10.1007/11536482_13

ASHOUR, Y.; KOLAREVIC, B. Heuristic Optimization in Design. *In*: ANNUAL CONFERENCE OF THE ASSOCIATION FOR COMPUTER AIDED DESIGN IN ARCHITECTURE, 35th, 2015, Cincinnati, USA. **Proceedings [...].** Cincinnati: ACADIA, October 2015, p. 357-369.

AUTODESSYS INC. **FormZ**: 3D modeler, version 8.6. Available at: http://www.formz.com/fzsite1219/index.html. Accessed in: 29 mar. 2020.

BERLINSKI, David. **The Advent of the Algorithm – The Idea that Rules the World**. New York: Harcourt, INC., 2000. ISBN 0-151-00338-6

EASTMAN, C.; TEICHOLZ, P.; SACKS, R.; LISTON, K. **BIM Handbook: A guide to Building Information Modeling for owners, managers, designers, engineers, and contractors**. Hoboken, New Jersey: John Wiley & Sons, 2008. ISBN 978-0-470-18528-5

FINK, T.; KOENIG, R. Integrated Parametric Urban Design in Grasshopper / Rhinoceros 3D – Demonstrated on a Master Plan in Vienna. *In*: ANNUAL CONFERENCE OF EDUCATION AND RESEARCH IN COMPUTER AIDED ARCHITECTURAL DESIGN IN EUROPE, 37th; ANNUAL CONFERENTE OF THE IBEROAMERICAN SOCIETY OF DIGITAL GRAPHICS, 23rd, 2019, Porto, Portugal. **Proceedings [...].** Portugal: eCAADe & SIGraDi, 2019, v. 3, p. 313-322.

GOEL, V. **Sketches of Thought**. Cambridge, Massachusetts: The MIT Press, 1995. ISBN 0-262-07163-0

GRAPHISOFT. **ArchiCAD**: BIM software. Version 21. Available at: https://myarchicad.com/Default.aspx. Accessed in: 29 mar. 2020.

KHALILI-ARAGHI, S.; KOLAREVIC, B. Captivity or Flexibility: Complexities in a Dimensional Customization System. *In*: ANNUAL CONFERENCE OF EDUCATION AND RESEARCH IN COMPUTER AIDED ARCHITECTURAL DESIGN IN EUROPE, 34th, 2016, Oulu. **Proceeding [...].** Oulu: eCAADe, 2016, v.2, p. 633-642.

KHEMLANI, L. Foreword, *in*: **BIM Handbook: A guide to Building Information Modeling for owners, managers, designers, engineers, and contractors**. Hoboken, New Jersey: John Wiley & Sons, 2011. ISBN 978-0-470-54137-1

KOLAREVIC, B. **Architecture in the Digital Age – Design and Manufacturing**. New York: Taylor & Francis, 2003. ISBN 0-415-27820-1

KRAUEL J., GEORGE W.; NODEN J. **Contemporary Digital Architecture – Design & Techniques**, Barcelona: Links Books, 2010.

LAWSON, B. **How Designers Think – The Design Process Demystified**. Oxford: Elsevier, fourth edition, 2005.

MITCHELL, W. J.; MCCULLOUGH, M. **Digital Design Media**. New York: Van Nostrand Reinhold, 1995.

PHILLIPS, E. M.; PUGH, D. S. **How to Get a PhD**. Buckingham, England, UK: Open University Press, 1987.

PLOTNIKOV, B.; SCHUBERT, G.; PETZOLD, F. Tangible Grasshopper: A method to combine physical models with generative, parametric tools. *In*: ANNUAL CONFERENCE OF EDUCATION AND RESEARCH IN COMPUTER AIDED ARCHITECTURAL DESIGN IN EUROPE, 34th, 2016, Oulu. **Proceeding [...].** Oulu: eCAADe, 2016. v. 2, p. 127-136.

POPPER, K. **The Logic of Scientific Discovery**. New York: Routledge, fourth English edition, 1985.

ROBERT MCNEEL & ASSOCIATES. **Rhinoceros-Grasshopper: 3D modeler and parametric programing**. Version 5. Available at: https://www.rhino3d.com/. Accessed in: 29 mar. 2020.

SCHUMACHER, P. **The Autopoiesis of Architecture**. Chichester, UK: John Wiley and Sons Ltd., 2011.

SHARAH, L.; ESCALANTE, E.; FABBRI, A.; GUILLOT, R.; HAEUSLER, M. H. Streamlining the Modelling to Virtual Reality Process. *In*: ANNUAL CONFERENCE OF ASSOCIATION FOR COMPUTER-AIDED ARCHITECTURAL DESIGN RESEARCH IN ASIA, 22nd, 2017, Suzhou. **Proceedings [...].** Suzhou: CAADRIA, 2017. p. 53-62.

SILVA, L.; SILVA, N.; LACROIX, I. Integrating parametric modeling with BIM through generative programming for the production of NURBS surfaces and structures. In: ANNUAL CONFERENCE OF THE INTERNATIONAL CONFERENCE OF THE ASSOCIATION FOR COMPUTER-AIDED ARCHITECTURAL DESIGN RESEARCH IN ASIA, 24th, 2019, Wellington. **Proceedings [...].** Wellington: CAADRIA, v. 1, 2019. p. 635-644.

SOMMERVILLE, I. **Software Engineering**. New York: Addison-Wesley, 2011.

TERZIDIS, K. **Algorithmic Architecture**. Amsterdam: Elsevier/Architectural Press, 2009.

TZONIS, A.; LEFAIVRE, L. **Classical Architecture – The Poetics of Order**. Cambridge, Massachusetts: The MIT Press, 1986.

WOODBURRY, R. **Elements of Parametric Design**. London: Routledge, 2010.

---

[1] **Neander Furtado Silva**
Architect. Ph.D. in Architecture from Strathclyde University, United Kingdom. Associate Professor at the University of Brasília, School of Architecture and Urbanism, Laboratory of Digital Fabrication and Mass Customization - LFDC. Postal Address: Universidade de Brasília, Campus Universitário Darcy Ribeiro, ICC Norte, Ala A, Sala ASS-589/9, Brasília, DF, Brasil, CEP 70919-970.

[2] **Lilian Maciel Furtado Silva**
Architect. Graduated in Architecture and Urbanism at the University Center of Brasília. Master's student at the University of Brasília, School of Architecture and Urbanism, Graduate Program. Postal Address: Universidade de Brasília, Campus Universitário Darcy Ribeiro, ICC Norte, Ala A, Sala ASS-589/9, Brasília, DF, Brasil, CEP 70919-970.

[3] **Igor Lacroix**
Architect. Ph.D. in Architecture and Urbanism from the University of Brasília. Post-Doctoral Researcher at the University of Porto, School of Architecture, FAUP, Center for Architecture and Urbanism Studies - CEAU, Digital Fabrication Laboratory - DFL. Postal Address: Rua do Campo Alegre, 695, Porto, Portugal, 4150-179.