Automating the Adaptive Reuse of Cultural Heritage

A computational framework for reconfiguring historic masonry buildings

Daniele M. S. Paulino[1], Kali L. Lewis[2], Rebecca Napolitano[3], Heather Ligler[4]
[1,2,3]The Pennsylvania State University [4]Florida Atlantic University
[1,2,3]{dpaulino|kll5590|nap }@psu.edu [4]hligler@fau.edu

Adaptive reuse is a challenging problem that combines spatial, structural, and architectural requirements, along with current restrictions in design, enhanced in buildings with heritage value. In this scenario, generative design techniques allow for exploring different arrangements for a pre-defined problem, assisting professionals during complex design tasks. Therefore, this work explores the potential of Shape Grammars in generating solutions for the adaptive reuse of Sobrado buildings, the prevalent typology in the historic center of São Luís. This paper focuses on the computational implementation of the transformation rules using rhino script syntax and python. The methodology proposes a framework for adapting buildings into multi-family apartments, considering existing spatial and structural requirements. It investigates the allocation of three apartment types in the floor plan: studios, one-bedroom, and two-bedroom apartments. The adopted strategy for the distribution of internal spaces considers existing elements, such as windows and balconies, allowing to benefit from the natural daylighting characteristics of the buildings, following the original projects during the XVIII and XIX centuries. An interactive approach is implemented to explore the potential of generating diverse spatial arrangements during the adaptive reuse process.

Keywords: *Adaptive Reuse, Shape Grammars, Generative Design, Mass Customization, Space Planning*

A GRAMMAR-BASED STRATEGY FOR ADAPTING HISTORIC BUILDINGS

In the last decade, a rising effort to improve sustainable solutions due to climate change has impacted how we design cities and buildings. The construction & demolition (C&D) building sector represented around 37% of CO2 emissions and 34% of energy consumption in 2022 (United Nations, 2022). Thus, more than re-imagining how we sustainably design new buildings, we should develop solutions that preserve the existing building stock, adapting it to modern user needs.

In this context, adaptive reuse (AR) of cultural heritage buildings can reduce material waste and maximize embodied energy use (Foster and Kreinin, 2020). It also can contribute to preserving cultural heritage elements and sites and ensuring a sense of community through how people interact with existing buildings.

Defining frameworks that support adaptive reuse strategies can facilitate the development and execution of AR projects. In this context, Shape Grammars (SG) is a powerful tool for generating design solutions and defining rules to describe transformations in design processes (Duarte, 2005; Eloy and Duarte, 2015; Ligler, 2021). Defining grammar-based computational tools can advance the generation of design options for a given

problem, allowing one to quickly explore a wide range of design variations.

The Reviver grammar (RG) is introduced in (Paulino et al., 2023, 2022) as strategy for analyzing and facilitating the transformation of buildings in the Historic Centre of São Luís – Brazil (HCSL), a UNESCO Cultural Heritage Site. Essentially, the grammar defines a strategy for subdividing a given floor plan, maximizing the reuse of existing elements, such as structural walls and openings, and allocating apartment units based on existing daylight conditions.

Implementing SG systems is a challenging task, especially in terms of programming and computation rule abstraction. Eloy et al. (2018) highlights the need for frameworks that fully demonstrate the SG's generative power. This study defines an interactive tool for automating the exploration of solutions for the adaptive reuse process of historic masonry buildings. The prototype tool is designed to implement a grammar-based strategy for converting an existing plan into multi-family units, considering the allocation of studios, one-bedroom (1-BR) and two-bedrooms (2-BR) apartments.

A TOOL FOR ASSISTING THE ADAPTIVE REUSE PROCESS OF HISTORIC BUILDINGS

The HCSL is a 220-acre territory, with around 5,600 buildings, presenting a rich typology diversity. Since the early 80's, there has been a concern with promoting social habitation in the HCSL. The Social and Habitation Promotion Subprogram (SPSH from the Portuguese *Subprograma de Promoção Social e Habitação*) was created in 1981 to stimulate affordable housing implantation. However, less than 15 buildings were renovated for residential purposes in more than four decades (Silva, 2019).

Thus, we propose a computational tool for automating the adaptive reuse of cultural heritage. The methodology allows exploring multi-family apartment possibilities for an existing building, based on its floor plan configuration. The tool is envisioned to be used by preservation organizations that desire quickly identifying design solutions during pre-architectural detailing phases. Usually, these agencies must approve architectural projects for buildings with historical values, ensuring compliance with preservation guidelines.

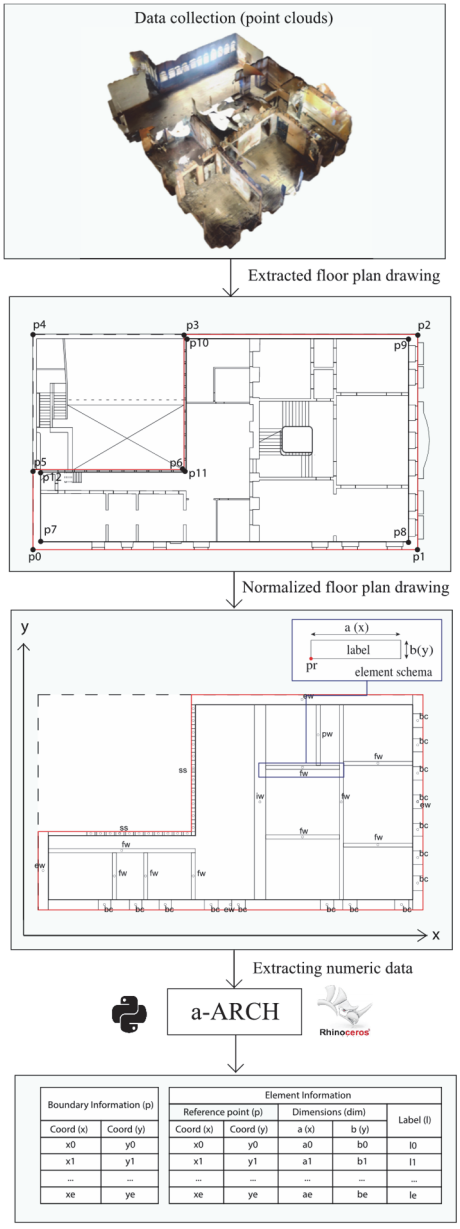
This can be particularly beneficial to:

1) Define a sequential and automated strategy for allocating apartments in an existing floor plan.
2) Identify acceptable layout solutions considering the target number and unit type (studios, one-bedroom and two-bedroom apartments).
3) Accelerate the adaptive reuse process, especially considering large-scale solutions.

Buildings in the HCSL follow a modular construction, profiting from masonry and wood elements. The construction typology known as Luso Brazilian architecture facilitated a large-scale implementation in a time of economic growth during the Portuguese colonization. Thus, developing a framework for accelerating the generation of viable design solutions can facilitate the adoption of a large-scale urban plan for the adaptive reuse of buildings in the HCSL, especially considering the high demand for residential properties in the region.

For implementation purposes, grammar rules are abstracted into numeric data and operations and are incorporated into a python library introduced as a-ARCH (automating the Adaptive Reuse of Cultural Heritage). The python library incorporates functions written using RhinoScriptSyntax (RSS), which converts numeric data into visual components within Rhino 3D.

Floor plan drawings can be automatically obtained while conducting a building survey using documentation techniques, such as laser scanning and photogrammetry (Liu et al., 2018). Currently, applications can facilitate the documentation process of buildings. As an example, the app Polycam (Polycam Website, n.d.) allows the generation of 3D models and facilitates the



Boundary Information (p)		Element Information				
		Reference point (p)		Dimensions (dim)		Label (l)
Coord (x)	Coord (y)	Coord (x)	Coord (y)	a (x)	b (y)	
x0	y0	x0	y0	a0	b0	l0
x1	y1	x1	y1	a1	b1	l1
...	...	...	...	...	...	...
xe	ye	xe	ye	ae	be	le

Figure 1 – Methodology for initializing the grammar-based approach.

extraction of a blueprint with measurements. This is a quick and less expense solution compared to more sophisticated laser scanning techniques.

Thus, any floor plan can be abstracted considering its geometric information, such as the dimension and position of elements in the floor plan. Essentially, a building plan contains two sets of information: a) *boundary*, which consists of a list of points defining appropriate boundaries of the building; b) *elements*, which consists of geometric information on existing elements, such as walls, windows, stairs, patios, etc.

When defining the geometric information input, elements are rationalized to orthogonal geometries aiming to facilitate the implementation process. Since the library aims to automate the adaptive reuse process, the geometric information for a given building can be automatically extracted from a drawing. Once the user to selects the plan drawing, a function is defined to identify objects, extracting, and storying their information as arrays.

The numeric data will be used to initialize the derivation process. The process for initializing the grammar-based approach is summarized in Figure 1. In sequence, functions equivalent to operations for each grammar stage can be executed to obtain design derivations.

ANALYZING AND TRANSFORMING AN EXISTING FLOOR PLAN

The grammar defines five stages, which can be summarized into three tasks: 1) Converting numeric data input into a labeled floor plan; 2) Defining a grid-based strategy for analyzing an existing plan and 3) Generating variations of design solutions for the adaptive reuse process.

Tasks one and two provide a strategic approach for generating and analyzing floor plan drawings. First, it automatically color differentiate elements in the given floor plan (stage 1 in the RG) and subdivides and classifies the design space according to daylight conditions (stages 2 and 3 in the RG).

Then, during the apartment allocation process, an interactive approach is defined to allow users to

select areas to combine into individual units (stage 4 in the RG). This can provide flexible apartment combinations for a given floor plan. Stage 5 is not yet implemented, which concerns the room allocation within each apartment unit. The next sections in this paper will detail and illustrate the implementation process.

Initially, the numeric data input is used to generate and visually differentiate elements in the floor plan, according to their type/labels. As defined, the data input consists of information regarding a building's boundary and its individual elements (walls, windows, etc). Based on the given data input, the code reads and stores the information as two main types of variables: 1) a list of point coordinates coords (x_i, y_i) defining a given boundary, and 2) a list of reference point coordinates *coords* (x_j, y_j), element dimensions *dim* (a_j, b_j), and *label* (l_j) for each element in the given story floor plan.

These two variables are fundamental for starting the ruleset implementation process within the numeric component. The indices *i* and *j* represent the number of boundary points and number of elements, respectively. For each boundary in the building, a different *boundary* type variable is defined (*lot, ext, int*).

Based on these variables, the code can be initialized to generate and color label elements in the plan, as defined in stage 1 for the RG. Essentially, it starts with a function to generate and label boundary lines considering the list of points defining each boundary (rules in step 1.1). In sequence, a function is defined to apply color hatches according to the labels, aiming to visually differentiate elements in the floor plan (rules in step 1.2). Once these rules are applied, the labeled floor plan is generated.

Stage 2 defines a grid system to subdivide the floor plan. Aiming to simplify the implementation of this step, constant values defining grid lines parallel to the x-axis and y-axis are identified as the grid reference coordinates (as shown in Figure 2). Step 2.1 defines rules for generating grid lines for a given floor plan, two components are used: 1) the midpoint between two consecutive openings, defined as rule 2.1.1; 2) points defining the internal boundary (int), defined as rule 2.1.2.

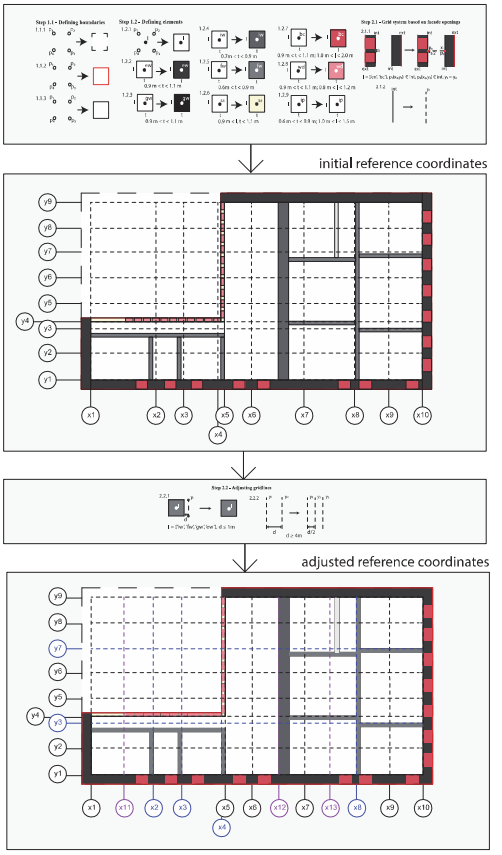


Figure 2 – Defining the grid system.

Then, a function to adjust reference coordinates is defined. First, it adjusts coordinates considering the distance to a structural element. If the distance is less or equal than 1 m, the coordinate is adjusted to match the coordinate defining the structural element. This step can facilitate the preservation of existing structural elements in the floor plan (rule 2.2.1).

In addition, if the distance between two consecutive coordinates, in either x or y direction, is greater than 4 m, a new reference coordinate is added in the

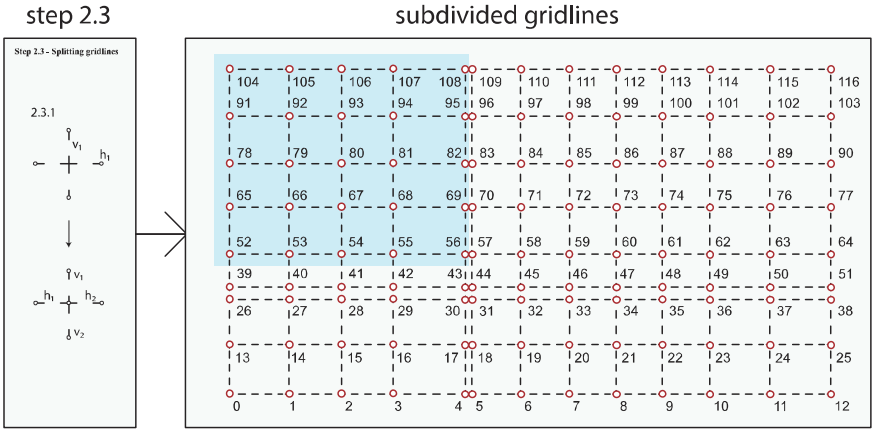


Figure 3 – Numbering system for grid points.

middle distance between them (rule 2.2.2). This is particularly important during the apartment's allocation and rooms subdivision since it can generate a more uniform space distribution.

It is important to emphasize that these adjustments cannot be performed to coordinates corresponding to lines from the internal boundary, only from midpoints between façade elements. The process for applying rules from stage 1 and steps 2.1 and 2.2 for stage 2 is presented in Figure 2.

In sequence, step 2.3 defines a rule for splitting grid reference coordinate lines considering the intersecting points between vertical and horizontal lines. For determining grid points, a function is defined by combining x and y components of reference coordinates. Thus, a vector containing each grid point's coordinates (x, y) is stored (*grid_points*). The index for each point is defined according to the number of columns and rows in the grid system, numbering each row in sequence (Figure 3).

Then, a vector containing grid line information is defined as *grid_lines*. Since two points can determine any line, the vector essentially stores the index of connectivity points for any vertical and horizontal grid line. However, some points and grid lines in the domain are located outside the internal boundary line (region highlighted by the blue box in Figure 3).

A function is defined for checking which points are outside the boundary and a binary variable (*isInsideBoundary*) defines if the point is inside or outside the internal boundary. This function is particularly important for the studied typology since building's shape can vary between L, U, and C shapes (Figure 4). This defines the internal patios with the shuttle system, which consists of wooden closures and glass windows.

It may be necessary to reapply rule 2.2.1 to adjust grid lines near structural elements, as previously detailed. This can occur because reference

Figure 4 – Different layout configurations for buildings in the HCSL.

coordinates based on the internal boundary were not adjusted. Also, one reference coordinate can be near multiple internal structural walls, which during the previous adjustment process can only consider one element. Now, since grid lines were split into smaller elements, it is possible to adjust for each internal structural element.

Areas from grid lines (rule 3.1.1) can be easily implemented considering the same logic used to create elements in the floor plan. Thus, for each grid area, a reference point (p_1) and its dimensions in x and y are determined. The area dimensions can be achieved considering the difference between two consecutive elements of *grid_coords*, for x and y direction. This information is stored as grid_areas and used to create areas in the rhino-python script by using the AddRectangle command, considering the point object id corresponding to the reference point and the area dimensions.

In the strategy, areas within 1.5 m to the shuttle system will be allocated as corridors. If an existing area is wider than this distance, a rule is applied to split it into two new areas. Since this rule can be generalized to be used in other steps, as well as a rule for merging areas, they are defined in step 3.0 as auxiliary rules.

Then, each area can be classified according to its proximity to openings, following the rules from stage 3.2. The implementation uses the CurveCurveIntersection command, which identifies two intersecting curves. First, it identifies regions intersection the shutter system (walls containing patio windows), which are classified as 'fa' (fixed areas), and color labeled in grey (rule 3.2.1).

Next, it identifies 'ds' (direct sunlight) areas (rule 3.2.2) by verifying grid areas that intersect an existing opening (both façade openings and secondary windows facing external patios). These areas are color labeled in orange. In sequence, 'cs' (close to sunlight) areas are identified by verifying areas that intersect 'ds' areas (rule 3.2.3). These regions are color labeled in yellow. Then, areas intersecting 'cs' regions are classified as 'fs' (far from sunlight) areas and color labeled in light yellow (rule 3.2.4). Lastly, if a 'fs' intersects a 'fa' area, they are automatically changed to 'fa' (rule 3.2.5).

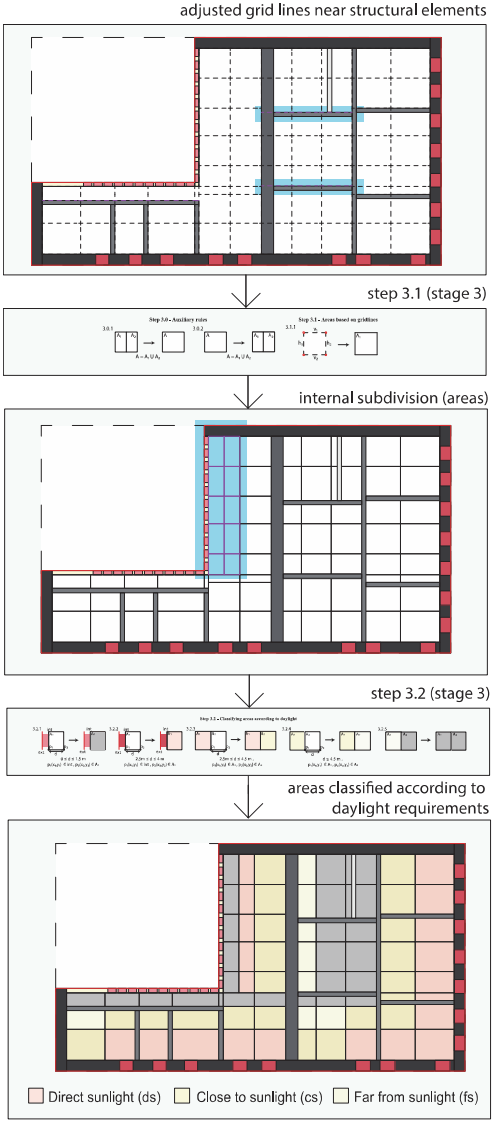


Figure 5 – Design space subdivision and classification according to daylight requirements.

Adjustments to 'fa' regions are also performed, following the rules for merging (rule 3.0.1) and splitting (rule 3.0.2) areas. Essentially, if an 'fa' has one of its dimensions smaller than 1.5 m, they can be expanded considering adjacent areas. Figure 5 exemplifies the process defined by stage 3.

ACHIEVING DESIGNS VARIATIONS USING THE A-ARCH TOOL

Once the design space is subdivided into grid areas and classified according to their daylight conditions, step 4 in the RG concerns the multi-family apartments allocation process. A prompt-based strategy is implemented to facilitate the interaction of users with the proposed grammar-based approach. It allows users to select grid areas to define apartment units, interactively generating space layout variations.

The grammar was designed as a generative solution for the adaptive reuse process. Aiming to generate viable solutions, the design space is constrained to allow users selecting only 'ds', 'cs' and 'fs' regions.

Figure 6 highlights the allowable regions for the selection process in green. Depending on the type and number of desired apartments, the combination of grid areas can have multiple solutions. Aiming to facilitate the apartment allocation process, a prompt command-based strategy is implemented to receive input from users.

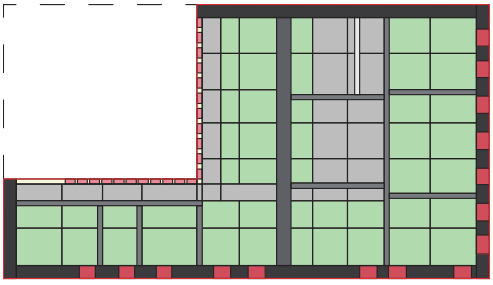


Figure 6 – Allowable areas for the user selection process.

It essentially asks users how many units for each apartment type they aim to allocate. It calculates if the given number of studios, 1-BR and 2-BR is possible based on the total available area and number of windows, following the minimum requirements for each unit. Studios must have a minimum square footage equal to 30 m^2 and at least one window. 1-BR apartments must have a minimum square footage equal to 39 m^2 and at least two windows. While 2-BR apartments must have a minimum square footage equal to 47 m^2 and at least three windows.

If it is a viable solution, the area selection process starts. Otherwise, it gives the user some combination options to select. Essentially, it asks the user to click into allowable regions for transforming them into an apartment unit.

Once the selection for a unit is concluded, it checks the total square footage and the number of windows for the given selection, and it classifies according to the minimum requirements for each apartment type (studio, 1-BR and 2-BR apartments). After each unit selection, the code allows to delete existing units or to add more units, always informing the user how many remaining units for each apartment type, considering its desired combination. This process is repeated until the desired number of apartments is achieved. Since the design space is constrained based on the area types and their daylight conditions, variations maintain a certain consistency, which guarantee the achievement of viable apartment configurations.

The user interactive solution also contributes to inform the user of square footage for each unit allocated in the plan, as well as the number of windows. This is useful to guide the selection process, allowing the user to delete any units that may not complain with the desired metrics. The authors tested the tool and generated 34 possibilities for the same case study building. Figure 7 shows a representative down sample with 20 variation possibilities, considering the apartment allocation design space (region highlighted in green in Figure 6).

One possible quantifying measure is the mean square footage for similar unit types. As for example,

Figure 7 – A down sample of variations achieved using the a-ARCH tool.

variations 3 and 5 contains three studios, two 1-BR and two 2-BR apartments. However, the mean square footage of studios for variation 3 is 32.5 m^2 in contrast of 32.7 m^2 for variation 5. Similarly, for 1-BR apartments, variation 3 has a mean square footage of 45.5 m^2 in contrast to 40.75 m^2 for variation 5. And for 2-BR apartments, variation 3 has a mean square footage of 51.85 m^2 in contrast to 54.2 m^2 for variation 5.

CONCLUSIONS AND FUTURE WORK

This work explores a generative design tool for the adaptive reuse of historic masonry buildings. It introduces an interactive tool for facilitating the allocation of apartments in an existing floor plan, targeting the reuse of existing elements, such as structural walls and openings. The computation tool is based on a grammar approach and considers a prompt-based strategy for receiving user inputs, guiding the apartment allocation process.

An existing floor plan is first analyzed and subdivided considering existing elements, such as façade openings and structural walls. Based on the position of elements, it constrains the design space to identify suitable regions for apartment allocation. Then, users can select allowable areas to generate different combinations of apartment types (varying between studios, 1-BR and 2-BR apartments).

The main goal is to introduce a tool to facilitate planning stages for the adaptive reuse of historic buildings, especially by preservation organizations. Thus, it can automate the adaptive reuse process, targeting a large-scale implementation in historic districts with high vacancy rates and low residential concentration.

Although this work explores the context of the HCSL, it defines a generalized computational framework that can assist analyzing other urban contexts with similar layout typology. Many historic downtowns, especially in Europe and their colonized countries, present similarities in how buildings interact among blocks, despite differences in architectural styles and functional typology. Often historic downtowns have low-rise masonry buildings distributed in the same block, prevalently, with dimensions and façade openings similar to the buildings in the HCSL.

Next steps in this research includes automating the generation of apartments combination, by implementing a clustering algorithm, such as the depth-first backtracking (Zawidzki and Szklarski, 2020). In addition, evaluation metrics can be included to analyze design variations, such as Construction & Demolition (C&D) costs.

ACKNOWLEDGEMENTS & FUNDING

This material is based upon work supported by the National Science Foundation under grant IIS-2123343. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

We thank José Pinto Duarte and Thomas E. Boothby for fruitful discussions during committee meetings. We also would like to thank IPHAN-MA and Margareth Figueiredo for their assistance during data collection. We acknowledge FAPEMA, SECTI and the Maranhão State Government for providing partial funds for this work. We also acknowledge the Architectural Engineering department and the Stuckeman Center for Design Computing at The Pennsylvania State University for all the support during the execution of this work.

REFERENCES

Duarte, J.P., 2005. Towards the Mass Customization of Housing: The Grammar of Siza's Houses at Malagueira. Environ. Plan. B Plan. Des. 32, 347–380. https://doi.org/10.1068/b31124

Eloy, S., Duarte, J.P., 2015. A transformation-grammar-based methodology for the adaptation of existing housetypes: the case of the 'rabo-de-bacalhau.' Environ. Plan. B Plan. Des. 42, 775–800. https://doi.org/10.1068/b120018p

Eloy, S., Pauwels, P., Economou, A., 2018. AI EDAM special issue: advances in implemented shape grammars: solutions and applications. AI EDAM

32, 131–137. https://doi.org/10.1017/S0890060417000634
Foster, G., Kreinin, H., 2020. A review of environmental impact indicators of cultural heritage buildings: a circular economy perspective. Environ. Res. Lett. 15, 043003. https://doi.org/10.1088/1748-9326/ab751e
Ligler, H., 2021. Reconfiguring atrium hotels: Generating hybrid designs with visual computations in Shape Machine. Autom. Constr. 132, 103923. https://doi.org/10.1016/j.autcon.2021.103923
Liu, C., Wu, J., Furukawa, Y. (2018). FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans. Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 201-217
Paulino, D.M.S., Knapp, C., Ligler, H., Napolitano, R., 2022. The Reviver Grammar: transforming the historic center of São Luís through social housing, in: Proceedings of the XXVI Conference of the Iberoamerican Society of Digital Graphics, Lima, Peru (online). https://doi.org/10.19083/978-612-318- 444-5
Paulino, D.M.S., Ligler, H., Napolitano, R., 2023. A Grammar-Based Approach for Generating Spatial Layout Solutions for the Adaptive Reuse of Sobrado Buildings. Buildings 13, 722. https://doi.org/10.3390/buildings13030722
Polycam Website (n.d.). LiDAR scanning and photogrammetry made easy. Available at: https://poly.cam (Accessed 19 March 2023)
Silva, I. da C. (2019). Políticas Habitacionais No Centro Histórico Ludovicense. In Proceedings of the IX International Seminar of Public Policies, São Luís, Brazil, 20–23 August 2019, pp. 1–12.
United Nations (2022). CO2 emissions from buildings and construction. Available at: https://www.unep.org/news-and-stories/press/co2-emissions-buildings (Accessed 14 March 2023)
Zawidzki, M., Szklarski, J., 2020. Multi-objective optimization of the floor plan of a single story family house considering position and orientation. Adv. Eng. Softw. 141, 102766. https://doi.org/10.1016/j.advengsoft.2019.102766