Generative Diagrams: Embedding Spatial Data to Generate Informed Architectural Designs

Paula Gómez
Georgia Institute of Technology, USA
✉ paulagomez@gatech.edu

ABSTRACT

In early stages of architectural design, designers manage complex but not accurate information in terms of spatial dimensions and technical specifications. To include quantitative information at early stages, parametric diagrams (PD) are designed to manipulate accurate information in order to generate architectural design alternatives. PD controls the diagrams that manage all of the information that relates to the design, including the geometry of an architectural design proposal. PD models work by designing spatial relationships through diagrams, and not by designing physical architectural elements.

KEYWORDS: architectural diagrams, generative, parameters, software, design.

Diagrams are commonly used in many disciplines with different meanings. In architecture, the range is wide; diagrams can be considered from sketches to notations (Goodman, 1976, 218) depending on the accuracy of the information they contain. In this case, the term "parametric diagram" considers the discrete data of the geometric descriptions. Geometrical descriptions can be abstract as well as complex ideas of 3D space, from concrete geometrical parameters containing physical information such as lighting, or abstract information such as "connectivity", a term borrowed from the field of space syntax, where those concepts are calculated by the geometric properties of a space.

The age of creative design by imposing shapes stays behind in this approach. The generated model is not only a representation of the architectural information; it is also an informed and parameterized model generated by this bottom-up approach. The first case study was done using connectivity information, which is a measurable property of a space (space syntax) that sums the amount of surface that is seen from one position. To calculate connectivity, a top drawing of the space is divided into a grid and the sum of "pixels" seen from each pixel indicates the connectivity level. The connectivity of a space is represented graphically by a range of colors filling the pixels, from low (dark gray) to high (light gray). A tool created by UCL, DepthMap, is used to evaluate many space qualities such as connectivity, integration, and control, among others.

From Drawings to Informed Diagrams

The word "diagram" has many different definitions and interpretations depending on the field of study. While for design disciplines a diagram is an "essential representation for thinking, problem solving, and communication" (Do, 2001, 1), in cognitive science it could be anything that visually represents another thing or process. During the twentieth century, the fundamental diagrammatic approaches in architecture have shifted their focus from the drawing to the diagram. Christopher Alexander, in his book *Notes on the Synthesis of Form*, states "the ultimate object of design is form", presenting his focus on form as a problem of theory (Alexander, 1964, 15-28). In his book, *Diagram Diaries*, Peter Eisenman focused instead on the diagrammatic techniques that provoke the emergence of forms "through an exhaustive sequence of operations: transformations, decomposition, grafting, scaling, rotations, inversion, superposition, shifting, folding, etc." (Eisenman, 1999, 7).

During the 90s diagrammatic techniques in architecture were based on new software capabilities and their commands. Some of the topics of the decade were the manipulation of digitalized inputs (Kalay, 1989, 5), the exploration of commands to construct 2D, 3D and 4D diagrams, and animation and simulation. For example, Greg Lynn, in his book *Animated*

Form, represents traffic fluxes as input information for 4D designs (Lynn, 1998, 103). The shapes designed were a result of the modeled and animated variables, and had a pictorial look, mixing verbal concepts and mathematical operations. The inquiry was: how to interpret diagrams to extract the information that had unexpectedly emerged.

Generation, Evaluation, and Data Visualization

Today, parametric modeling and evaluation software applications have produced two different technical approaches in architecture: generative geometries and performance analysis respectively. These approaches are considered diagrammatic paradigms, or "the paradigm of digital architecture" (Leach, 2004, 10), since both use quantitative information and produce visual representations as results. In the generative geometries approach, information defines the shape. Mathematical equations are used to generate patterns within hierarchical relationships between geometrical properties of objects driven by parameters, constraints, and rules.

In the performance analysis approach, diagrams are constructed with external and accurate inputs. Two examples are energy (Becker, 2008, 2) and space syntax analysis (Bafna, 2003). Both evaluations result in data sets that are visually represented in relation to building geometry. However, the major difference between them is when each process is applied within the design timeline. The exploration of new shapes in the geometrical approach takes place in an early stage of the design process; the performance-based evaluation is closer to the final stages of a design process, space syntax lying between the two.

The query that emerges is how to integrate the geometrical generative process of early stages of design and the information included in the analysis done at the later stages of design. A concrete approach would be to evaluate the design as close as possible to perceive it as a real time evaluation, but a gap will still exist. What tacit information is it that designers consider in their decision-making to fill this gap? If all the information were explicit information for the computer, the designer could consider quantitative information from the beginning of the design process to make informed decisions.

Generative Informed Diagrams

The approach proposed for producing generative informed diagrams (GID) is the transformation of information into computable data to create visual and interactive diagrams. GID is expected to fill the gap between the generation of geometries and the analysis of design outputs. Graphically, information is mapped into the space representation, and it can be adjusted to design through these diagrams. The three main characteristics of GID are visualization, interaction, and pattern recognition (Yi, 2007, 1). Visualization graphically maps the data into the spatial geometry; interaction allows designers "tuning" the parameters to obtain "optimized" geometrical outputs, and pattern recognition allows users to interpret data in order to make design decisions.

Today, interaction is an important characteristic to consider in architectural diagrams because a static 2D image is a difficult way to include the necessary amount of data. Selecting specific data, exploring it, filtering it by concepts, correlating the results, and reconfiguring it to find emergent patterns help designers to better understand correlations. The recognition of patterns, or "data missing," is another important characteristic that allows informed design decision-making that allows the modification of diagram components to define new patterns.

GID Tool

The GID tool approach is to design by painting the desired connectivity level in a top view of a building. The design intentions are considered inputs to create divisions or walls that support that specific connectivity level. As connectivity counts the number of pixels seen by each pixel in the array, the solutions to generate walls are expressed in a 1:many relationship—one connectivity level has many solutions. To reduce the number of solutions, the variables should have different weights and some variables depend on other variables. For example, instead of being $A + B + C = D$, it will be: $4A + 3B + 0C = D$ (tuning) & $A= B/2$ (parametric).

The complete proposal has three stages; this paper only develops the first stage. The first stage is to create a GID tool where a "seed" of the connectivity is sown, generating the size of the space required to accomplish the connectivity level required (Fig. 1). The second stage is to create a simplified version of DepthMap in Java to incorporate modifications in real time and return a new feedback. The third stage will incorporate more variables to make the design more informed. A requirement for the second and third stages is to find the precise algorithms in order to get concrete results (and not infinite solutions) for a connectivity state. The advantage of this exploratory process is that it provides some unexpected alternatives that match the best design outputs.

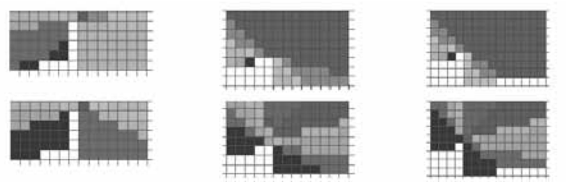

Figure 1: Space syntax analysis, top: connectivity; bottom: integration

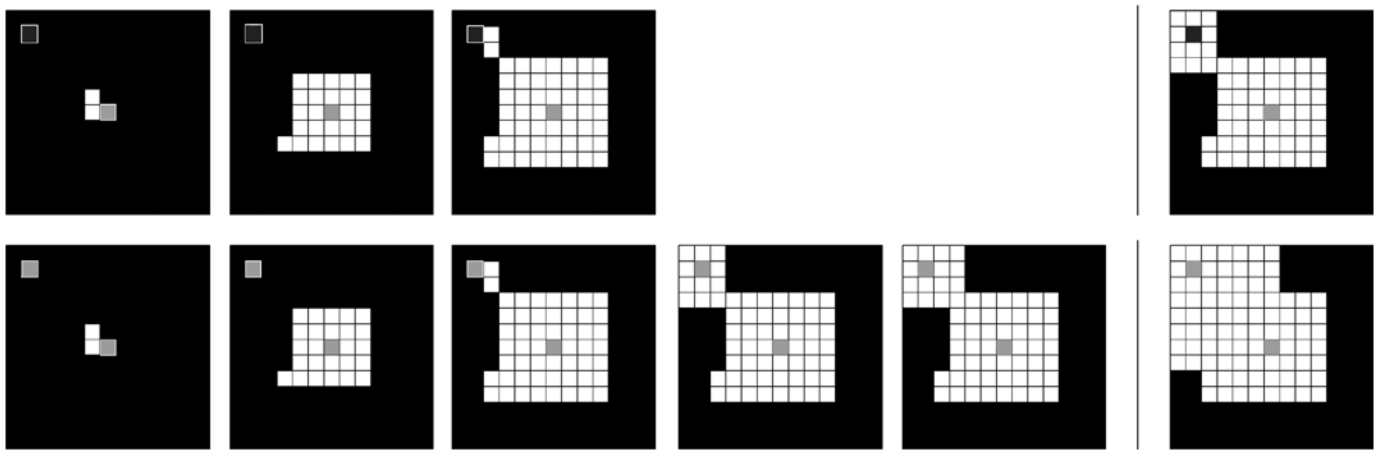

Figure 2. GID process: dark-gray + light-gray seeds

In the first stage of the GID tool the operation implants the color-seeds that initiate the generation of connectivity (Fig. 2). The components for this tool are the field of operation and the seeds. In this first test a grid of twelve by twelve is given as an operational field. The grid is considered a non-space, or a space full of walls represented by black. The seeds have four important properties: a color that represents the level of connectivity desired, a number of pixels associated with that color (relative to the total of pixels involved), a radius of action that depends on the number of pixels and on the perimeter of the field, and the order to start running the algorithm. The first three characteristics are inherent to the seed; however, the last one is related to the other seeds located in the field. The order of operation and spread direction of the seeds will depend on the distance of the other seeds implanted in the field.

The rules are simple: the first seed implanted is the first starting the algorithm. The color of the seed indicates the number of pixels to empty in order to create the space we are looking for. The algorithm is always the same: delete black pixels (turning them white) in a spiral direction. If the pixel to be deleted is within the field, it is counted. If the pixel is not in the field or it already white, it is not counted. The spiral continues jumping these pixels. The direction of the spiral (clockwise or counterclockwise) depends on the position of the second seed. The staring direction of the spiral (north, south, east or west) is defined by the closest next seed in X and Y terms, meaning if a cell is closer in the X coordinate than in the Y the spiral will start deleting the X pixel next to the seed to the next-seed direction (Fig. 2).

The radius of action of the seeds will determine if other seeds are inside or outside of it. If a seed is located inside of another radius of action, the less integrated seed (dark-gray) would start running the algorithm, and the rule of creation order will play second (Fig. 3). When a dark-gray seed is inside the radius of action of a light-gray seed, the dark-gray seed will start running the algorithm. After that, when the light-gray seed runs the algorithm, it will consider the perimeter as the dark-gray seed radius of action plus one pixel. A wall will appear (or will be never deleted) between those two radii of action, with the exception of one pixel to connect the two spaces created.

When the results of the tests made in a twelve by twelve grid are compared with the analysis made in the DepthMap tool, the connectivity indicators are very similar in both cases if the radii of influence are not intersecting other radii seeds. The ranges of color are similar but not exact in all cases. These evaluations validate somehow the set of rules that define the GID first test (Fig. 4), however, GID will need further development of the algorithm to be more precise.

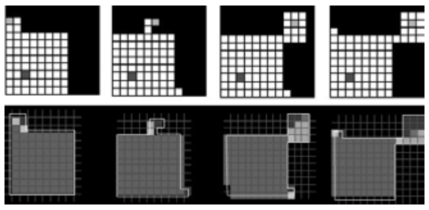

Figure 3. Seeds' radii of action and connectivity evaluations

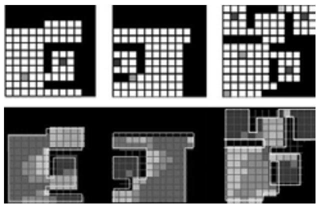

Figure 4. Dark-gray seed inside light-gray seed action zone

Conclusions

As the GID algorithm used to generate basic shapes for testing is in the initial state of development, it considers only the connectivity variable. Due to the simplicity of the algorithm, it works better for two seeds and basic shapes than for more complex cases. The algorithm must be further developed to support the inclusion of additional spatial variables, such as the distance between architectural elements and human behavior. For example, the architectural program influences the size of an atrium. Other variables could be orientation and connections based on human behavior that occur during the day, week or month.

In the most complex test done, it was observed that the contour of the connectivity areas was not regular as they were in the previous tests. The accuracy of the results is related to the contour and proportions of the operation field defined by the grid as well as the proportion and distances to the walls that appear during the process. New variables are the implantation new elements during the process and the rearrangement of existing elements or spatial conditions. The improved GID algorithm should support the re-arrangement of the cells when changes occur, such as implanting an architectural element or modifying the proportions of the operation field or the final contours of the shapes.

As the proposed methodology is a first step of further stages it lacks the necessary complexity to construct rich spaces. It generates spaces typologies following certain algorithms, which should become more complex when increasing the number of variables involved. However, in all the cases presented, the methodology provides a graphic visualization to interpret the results, which allows unexpected design patterns to emerge. GID will be a powerful tool that generates unexpected design alternatives and real-time feedback when it supports manipulation of the diagram and interaction with the results.

Acknowledgments

Thanks to my advisor Dr. Ellen Do and Dr. Sonit Bafna from the College of Architecture of Georgia Tech for their valuable feedback in this research, and special thanks to Marcelo Bernal, Ph.D. student of the College of Architecture of Georgia Tech for his constructive feedback and daily support.

References

Alexander, C. (1964). Goodness of fit. In *Notes on the Synthesis* (pp. 15-28). Cambridge: Harvard University Press.

Bafna, S. (2003). Space syntax: A brief introduction to its logic and analytical techniques. *Environment and Behavior*, 35 (1), 17-29.

Becker, R. (2008). Fundamentals of Performance-Based Building Design. *Building Simulation*, 1 (4), 356-371.

Do Yi Luen, E. & Gross, M. (2001). Thinking with Diagrams in Architectural Design. *Artificial Intelligence Review*, 15, 135-149.

Eisenman, P. (1999). *Diagram Diaries*. Universe Publishing.

Goodman, N. (1976). The Theory of Notation; Score, Sketch, and Script. In *Languages of Ar*t. Indianapolis: Hackett.

Kalay, Y. (1989). Introduction. In *Modeling Objects and Environments* (1). New York: John Wiley and Sons.

Leach, N., Turnbull, D. & Williams, C. (2004). *Digital Tectonics*. London: Artimedia Press.

Lynn, G. (1998). *Animate Form*. New York: Princeton Architectural Press.

UCL Bartlett School of Graduate Studies (n. d.). *UCL Depthmap: Spatial Network Analysis Software*. Retrieved from: http://www.vr.ucl.ac.uk/depthmap/.

Yi, J., Kang, Y., Stasko, J. & Jacko, J. (2007). Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13 (6), 1224-1231.