Computer-Based Form Generation in Architectural Design – a Critical Review

Yasha Jacob Grobman, Abraham Yezioro and Isaac Guedi Capeluto

Computer-Based Form Generation in Architectural Design – a Critical Review

Yasha Jacob Grobman, Abraham Yezioro and Isaac Guedi Capeluto

ABSTRACT

The idea of using computers for form generation and evaluation in the architectural design process has been put forward already in the early days of computers. However, as opposed to computer aided drafting, the generation of form, its optimization and manufacturing has not been widely accepted and implemented by practitioners.

The paper critically reviews the research and state of the practice experiments that has been done in this field and develops an argument regarding the possibilities and limitations of computer-based form generation in the architectural design process.

1. Introduction

The introduction of computers to architectural design ushered in the possibility of using computers to generate architectural form. While form evaluation acts on an initial form in an "after-the-fact" manner, in form generation the computer creates form directly from raw information [1] [2]. Based on this characterization, it can be assumed that form generation in architecture takes place only at the beginning of the design process while the rest of the process could be regarded as an optimization process which involves multiple evaluations. This would probably be the case for a design process that deals with the design of one element as in some cases of industrial design, or the design of fairly simple structures that, for example, have no internal divisions to secondary spaces. Nevertheless, when it comes to the design of buildings, it is clear that several levels of form generation might take place within the design process, as follows:

1. Generation of initial form/envelope
2. Generation of secondary spaces/division to floors and sub-spaces
3. Generation of building elements (façade, windows, doors, etc)
4. Generation of building details

The above four levels can be compressed into two main types: The first and more obvious has to do with the introduction of new forms, while the second has to do with changing the topology of the existing forms. Changing topology could be done by formal modifications such as adding walls or openings.

Kalay [3] questions the direct connection between form and function as suggested in Sullivan's famous slogan "form follows function." He claims that the solution space for the "appropriate design solution" lies in the intersection between form, function and context. Using this division as a source of information to generate design alternatives, one can suggest that these three "building blocks" represent programmatic information: Form/space defines the specification of spaces and the relationship between these spaces. Function defines the expected performance from these spaces and context defines the relationship with the urban fabric and environment. However, the above division does not cover the entire scope of information that could be used to generate form. Another important layer of information derives from the designer. This layer consists of perceptual and cognitive premises that are drawn from the designer's knowledge/experience and intuition.

In the early '60s, the computer was considered to be an intelligent problem-solving machine that would eventually match, and perhaps even supersede, human intelligence. Researchers have developed a plethora of models and theories to automate the design process and optimize its product [4]. Some believed that an entire "optimal" project could be computer-generated by translating the designer's creative work and

functional/performance-based criteria into quantitative information. This idea rested on the assumption that because the computer had superior "intelligence", it would be able to generate the "optimal" design. However, it is now clear that generating a complete "optimal" building is not possible:

- One cannot define finite performance criteria for an architectural project's final form because it is too complex and involves too many criteria [5], [6], [7], [8].
- Performance criteria can contradict one another. This forces the designer to rank (using formal or calculated criteria) each performance within a hierarchy of influence. By not fully adhering to any criteria, the design solution stops being optimal and becomes subjective, which aligns it with traditional design.
- Final building design consists of performance criteria that derive from perception/feeling and cognitive grounds. As of now, there is no known empirical way to translate these criteria into computer algorithms[1].

Nevertheless, it can be argued that computer form generation has significant advantages over traditional design when it comes to a small number of criteria and smaller well-defined problems such as the generation of a building's envelope/initial form or the generation of elements in the building.

2. Computer-based generative design methods/approaches developed in architectural research and practice

Computer-based form generation developed in a nonlinear manner along several parallel lines. In its early days, computer-aided design was expected by some scholars to replace conventional design by generating buildings from data. Thus, some generative systems aimed at complete spatial design that used the computer's processing power to overcome the designer's information-processing limitations and produce a final design, mainly plans. Less comprehensive approaches exploited the advantages of computer-aided design in helping the designer to lay out building plans, mainly for the type of building in which empirical criteria, such as the distances between programmatic functions and frequency of trips could be defined [9]. Other approaches argue that generation of form by computers is limited to narrow parts of the design process, mainly initial or conceptual design, geometry problems or other performance-related problems [10]. The following sections critically present and categorize the main existing approaches to computer-based form generation in academic research and practice:

[1]See also the discussion by Woodbury & Burrow [64] and Terzidis [65] on the advantages and limitations of computers in design.

2.1. Complete enumeration

This approach is based on the premise that computers can generate all the possible alternatives to the modification of an examined dependent variable. Despite the existence of several academic software programs (see examples presented by Kalay [3]), this research located no recent research in this domain or commercial software that uses enumeration.

2.2. Space allocation problems

Space allocation[2] aims to lay out spaces and activities in a building according to rational principles [3]. While space allocation approaches for design were developed mainly in 1960–1980[3], some research is still being done in this field (an example is the work by Michalek et al. [11]). Many of the earlier approaches concentrated on generating building plans from programmatic information. Kalay argues that space allocation is suitable for projects where circulation plays an important role and divides space allocation approaches into additive, permutational and constraint satisfaction [3]. The additive approach places the highly connected spaces first and then situates the other spaces around them. The permutational approach introduces the possibilities of swapping between spaces if the result is more satisfactory, as well as starting the allocation process randomly and then evaluating the results [12]. Constraint satisfaction refers to attempts to include additional design criteria (besides circulation) in the decision-making process by adding constraints to the placement algorithms. According to Kalay, this is done "for buildings where distances are of lesser importance than other criteria (e.g., privacy in a house), and where additional criteria (e.g., lighting and site condition) influence as much as or more than circulation." Various approaches to space allocation borrowed metaphors from other fields such as electricity [13] and mechanics [14] to define the algorithms.

Many of the earlier approaches concentrated on generating building plans from programmatic information. Most of these approaches concentrated on the generation of 2-D plans. Although some of them suggested 2.5-D (extruded planes), this research found no 3-D based approaches[4]. While the approaches mentioned above and others produced prototype tools that demonstrated the ability to generate floor plans, this research found they had no significant influence on architectural practice, nor did it find any commercial application for the generation of building plans currently used by designers.

---

[2]Space allocation is also known as "automated floor plan generation," "automated spatial synthesis" or "quadratic assignment formulation" [3].

[3]A survey on space allocation algorithms was done by Frew [9].

[4]3-D based approaches should include the possibility of defining different heights for different programmatic demands, which by definition will generate complex sections of buildings, as opposed to the 2.5-D approach in which the generated building consists of flat floors.

2.3. Cellular automata

One of the earliest types of computer-form generation tools can be traced to Cellular Automata (CA). Based on von Neumann's theories from the 1940s about computed-based, self-replicating forms, cellular automata began mainly as 2-D growth-simulating algorithms/software. One of the best-known examples of CA is Conway's Game of Life, developed in the 1970s, which allows the growth of 2-D patterns with three basic rules. Another well-known illustration is L-systems, an algorithm developed by the biologist and botanist Aristid Lindenmayer in the late '60s as a way to model the growth of plants by using a set of rules, constants and modifying parameters and by using different starting points [15].

Over the last two decades, CA has been adapted to architecture in various ways; one of the more promising avenues is the use of CA to simulate and predict the growth of cities, as seen in the work being done at University College London Centre for Advanced Spatial Analysis (CASA)[5]. Another avenue is based on the use of CA to develop complex formal expressions as suggested by architects such as Karl Chu [16] and Studio Rocker [15].

This work is interesting in terms of its potential to create highly complex conceptual forms that could be used for inspiration in the architectural form-finding process. Silver [17] used CA-based algorithms to generate building façade patterns. This approach was based on the understanding that computer power and simple rules can be used to generate a large number of complex formal patterns for the architect to evaluate.

2.4. Case-based reasoning/expert systems

A popular approach in CAD in the 1990s, case-based reasoning refers to an optimization method in which design decisions are guided by a single distinctive prior case (precedent, prototype, exemplar, or episode). Case-based reasoning seeks to determine a "source case" relevant to a given design problem or source case rules in expert systems. The process is thus separated into two parts: first, finding the appropriate source case and second, determining the appropriate parameters that need modification for the given design problem and developing an algorithm to perform these changes. Oxman [18], Heylighen and Neuckermans [19] and Kalay [3] present many approaches that were developed over the years in academic research[6]. Nevertheless, Heylighen and Neuckermans [19] argued that no convincing breakthroughs have yet been made. Moreover, none of the developed approaches seems to have deeply influenced architectural practice.

[5]See the working papers published by Michael Batty, Paul Torrens and other members of CASA [66].

[6]Both Oxman [20] and Heylighen [67] submitted doctoral dissertations on expert systems/case-based reasoning. Both are sources for a deeper understanding of this domain.

In terms of form generation, in addition to the approaches identified by Oxman [20], Heylighen and Neuckermans [19] and Kalay [3], this research identified another type of approach based on the idea that it is possible to define a parametric building template for certain building types. The template then can be used to generate different buildings from the similar type. One example of such a tool is Variomatics, developed by Oosterhuis [21].

2.5. Shape grammar and formal rule-based form generation

Shape grammar theory was first presented by George Stiny and James Grip in the early 1970s [22] [7]. They compared it to phase structure grammars, which were introduced by Chomsky in linguistics. While phase structure grammars use an alphabet of symbols to generate a one-dimensional string of symbols, shape grammars use singular shapes and produce multidimensional complex shapes. A shape grammar is defined in terms of shapes and rules. The rules in this case are strictly compositional. Shape grammar is oriented toward the initial stages in the design process where initial alternatives are generated [23] [24]. Several shape grammar-based form generation approaches were developed in the last two decades:

Straightforward shape grammar approaches allow the generation of complex forms by defining basic formal rules. An example of this type of application is the Shaper2D tool, developed by Miranda C. McGill [25]. Similar approaches are found in 3DShaper by Wang and Duarte [26], GEdit by Tapia [27] and Xp-GEN by Pak et al. [28].

Analysis-based approaches try to define the formal rules of certain building types or building designs by an architect and then to generate a new building in a similar style. Koning et al developed an application that can generate Frank Lloyd Wright prairie style houses [29], while Duarte developed a way to generate buildings based on a grammar derived from the analysis of Alvaro Siza's buildings [30].

Formal rules were utilized as a form generation mechanism in numerous approaches other than shape grammar. Shaviv et al. developed a rule-based application that uses solid modeler as a "high hierarchy architectural language" to generate models of church basilicas [31]. Oxman developed an application to generate kitchen layouts based on predefined elements [18]. Sei Watanabe [32] developed an application used to design a spatial framework structure based on formal computer manipulations.

Only a few of the rule-based applications have been distributed commercially; they include Genesis, an intuitive interactive and automated generation of complex 3-D designs in context, which has been applied to the design of Queen Anne-style houses and aircraft systems [33], Kaos by Tov Strikovsky [34] and Automason [35].

[7] Further information about shape grammar (history, people, projects, etc.) can be found at www.shapegrammar.org

Another formal rule-driven approach derives its inspiration from nature. In Digital Botanic Architecture (D-B-A), Dennis Dollens presents a series of designs that were generated in Xfrog using rules that derive from nature (mainly botany) [36]. Two design labs are at the forefront of a wider-ranging approach that seeks genetic rules or code for the design of architectural forms: The first is the International University of Catalunya's ESARQ (Escuela Técnica Superior de Arquitectura) program, which promotes the Genetic Architectures line of research [37] and the second is the Politecnico di Milano's Generative Design Lab headed by Celestino Soddu [38]. Both labs concentrate on defining formal codes/genes for the generation of "genetic" architectural forms.

In a different type of rule-based approach promoted in the late 1990s by Greg Lynn [39], architectural form was generated by modifying an initial form with "forces," form modification rules coded to objects that were placed around the initial form.

Shape grammar and the other rule-based "mechanical" approaches described in this section use computer processing power to generate intricate forms that could not be achieved within a normal design process. The fact that these forms were generated by computer rules makes their geometric logic easy to understand, thus facilitating better control, modification and construction. Nevertheless, since these forms are based only on formal rules, their impact on architectural design is arguably limited to realms such as creativity and inspiration. In terms of fitness criteria or the ability to differentiate between the generated forms, this approach alone does not offer any real solution.

To step beyond strict formalism, this approach must be associated with evaluation algorithms, as in the design of a coffeemaker by Agarwal et al. in which shape grammar was associated with cost evaluation [40]. A different approach could be to embed grammar modules in performance-based or another type of generative approach, as suggested by Shea and Cagan [41].

2.6. Evolutionary methods

Computer-based evolutionary methods are based on the idea of seeking the best solutions from a population of solutions (phenotype) based on different genetic code (genotype) [3].

Bentley [10] defines four characteristics of the evolutionary process: reproduction, inheritance, variation and selection. According to Bentley computer-based evolutionary programs that are based on algorithms also require initialization, evaluation and termination.

Evolutionary search algorithms are inspired by and based upon evolution in nature – evolving solutions to problems; instead of one solution at a time, these algorithms consider a large collection or population of solutions. Frazer [42] posits that to achieve the evolutionary model, it is necessary to define a genetic code-script, set rules for the development of the code, map the code to a virtual model, determine the nature of the environment for

the model's development and, most importantly, designate the criteria for selection. Bentley [10] argued that before applying an evolutionary algorithm, we must define the boundaries of the solution space (specify the phenotype), define the search space (genotype), find the algorithm[8] most suitable to the problem and define fitness function. He also suggests that the computer does not evolve anything; it is currently impossible to program in evolution, since we do not fully understand how evolution works. Instead, computers are instructed to maintain population of solutions, allow better solutions to "propagate" and allow worse solutions to "die."

This approach has garnered increasing interest in the last decade. As opposed to earlier approaches that were developed and tested almost solely by scholars, this time new approaches that use evolutionary methods were developed and tested by designers as well. This change can be attributed to the ubiquity of computers, the increasing familiarity of designers with computers and the enhanced connectivity between software and computers. Although it seems that the focus of research into evolutionary for generation methods in architecture has shifted in recent years toward improving its applicability in practical design problems, so far this study has not found evolutionary methods embedded in any commercial design application. Nevertheless, some tools have been developed within this realm in academic research, an example is a tool developed by Malkawi et al. [43] that combines morphing-based GA, CFD evaluation with user inputs for the optimization of the relationships between the measurement of a single room space, proportions and position of a window in that space and the specifications of the air conditioning supply in that space. The aim was to find the optimal window proportions based on the proportions and position of the air-conditioning supply. What is interesting about this approach is that the designer is given a chance to add his preferences, which are based on qualitative criteria, to the quantitative performance optimization.

The following paragraphs present several applications that represent various directions in evolutionary methods with a clear applicative approach:

GADES (Genetic Algorithm Designer) is a software program developed by Bentley [10] that offers differentiation into genotype and phenotype. Each block of genes is a coded primitive shape and each gene a coded parameter. "A mutation operation is used within the genetic algorithm to vary the number of primitives in a design by adding or removing new blocks of nine genes from chromosomes" [10]. Following this random increase in the form's complexity, the genetic evaluation algorithm examines the results

[8]Bentley [10] defines four main types of evolutionary algorithms: a) genetic algorithms – created by John Holland in the 1970s and made famous by David Goldberg in the late 1980s; b) evolutionary programming – created by Lawrence Fogel in the 1960s and developed by his son David Fogel; c) evolution strategies – created by Ingo Rechenberg in the 1970s and promoted by Thomas Back; and d) genetic programming – created by John Koza in the beginning of the 1990s.

and chooses the best solutions. Based on testing the design of a table, a boat, a hospital layout and a car, Bentley found that about 500 generations are needed to develop a solution [10]. GADES is the only tool located during this research that is capable of handling an entire design process, from zero to design prototype. However, the design scenarios in which it was tested were rather simple in terms of the fitness criteria used to evaluate the forms. It is reasonable to assume that in more complex problems (as in designing a complex building), it would be more difficult to define the fitness criteria[9]. An important feature of GADES is the possibility of adding user preferences to the evaluation, which could be helpful in managing complex fitness criteria scenarios.

Genr8 is a plug-in for Alias/Wavefront's Maya developed in 2001 at MIT. Based on an L-System growth algorithm, it generates surfaces, rule-based reflectors and retractors that are used to modify the initial surface and a genetic algorithm that "improves" the surface[10]. Genr8 was promoted and tested by the AA Emergent Technologies and Design program and the MIT master's program. Genr8 was used to produce some prototypes, but because it does not offer any way to introduce fitness criteria other than the one that controls the surface properties in the genetic algorithm, its use is limited to the form-based design domain and is not really different from manual manipulation of form.

Karl Chu argues for evolutionary design method in his X Phylum project. These forms are generated by writing an initial algebraic formula that develops continuously through further generations [44] [45]. Although highly intricate, it is unclear from Chu's writing what fitness criteria are used to choose the appropriate solution in a certain generation. The criteria appear to be strictly formal, making the major advantages of these evolutionary methods their ability to generate many solutions.

Although evolutionary methods are widely used in dealing with optimization problems in other disciplines, this research found no sign of evolutionary-based applications that are embedded in architectural software or used on a regular basis by architects in practice.

2.7. Geometric constraints-based form generation

Geometric constraints-based form generation refers to design methods that generate form via writing coded parametric constraints that control the formal relationship between several geometric forms. This tool permits the designer to set down basic elements of the architectural project's geometry, to define key control points and other parameters for these elements, and then to establish relationships among them such that alterations made to one or more of them will cause corresponding changes in the others. The

[9]An important feature that is offered by GADES is the possibility to add user preferences to the evaluation process. This feature has a potential to be helpful in complex fitness criteria scenarios.

[10]Information on Genr8 can be found in [68].

development of this approach can be traced to the UK office of Foster + Partners [46]. Employing Bentley's Microstation software as an interface for the new code, the office used this method to generate the initial form of several projects including the Swiss Re building, the London City Hall, the Chesa Futura and the Gateshead Sage Music Center. In the British Museum's Great Court design, instead of working with geometrical constraints, an algorithm was developed to negotiate the complex geometry of the basic design idea [47]. This method is less interactive than the ones previously described in this section and demands greater knowledge of mathematics and programming.

This idea is being developed further in two main directions: the first is similar to the approach suggested by Nir in his doctoral thesis and later developed into a commercial tool called Paracloud. It addressed the "need for schematic representations which allow handling real-world problems with simplified interfaces and the ability to drive multiple representations from a single logical model ..." [48]. Instead of geometric constraints or direct algorithms as in the case of Foster + Partners, Paracloud is based on "smart cloud of points" in which an "i" parameter that can be coded with geometrical and performance information was added to the x,y and z Cartesian parameters [49].

The second approach is based on the work of the SmartGeometry research group, which argues that "Architecture is fundamentally about relationships. Many of those relationships are geometric in nature or find a geometric expression" [50]. Generative Components, an application currently being developed by Bentley Systems, aims to generate form by defining geometric form with parametric coded constraints.

2.8. Performance-driven form generation

Performance-driven form generation refers to the idea that performance data can be used to generate architectural form. As opposed to the previous geometric constraint-based form generation, which is oriented toward geometry in terms of its target function (surface division, finding the most suitable curve in terms of geometrical relations to other curves), in performance-driven form generation, performance simulation is used directly to generate the form; in many cases, it is also the target function/criterion. To date, this study has identified only single-performance criteria solutions in performance-based form generation.

Of the two main approaches for this type of form generation, the first uses performance simulation as an inspiration for formal expression. This approach concentrates on the formal aspects of the generated form and does not argue for performance optimization. Examples are Greg Lynn's Port Authority Bridge project in which the shape of the bridge traced a simulation of trajectories of movements in the site and the H2 house, where the formal expression was generated by data on light and traffic [51].

The second approach tries to generate an optimal formal solution for predefined performance-oriented target functions, as seen in the design of a competition entry for the Florence train station by Isozaki and Sasaki, a structural engineer. The structure in this project was generated using Sasaki's Extended ESO Method, an evolutionary process that both multiplies and deletes elements during the generation process, as opposed to the currently most common practice of deletions only. Sasaki used the method to generate the final shape of the project's support columns, given the loads and the columns' desired location [52].

Capeluto [53], [54] developed a similar approach that used Solar Rights Envelope (SRE) and Solar Collect Envelope (SCE) data as target functions. The solar envelopes define the space of all possible design solutions that either considers solar insolation or solar shading.

Another performance-based generation tool, EifForm, is based on a method called structural shape. Developed by Kristina Shea, EifForm aims to develop an "overall form of a structure, together with its triangulated breakdown into structural elements and joints it works by repeatedly modifying an initial design with the aim of improving a predefined measure of performance, which can take into account many different factors, such as structural efficiency, economy of materials, member uniformity and even aesthetics" [55]. The aesthetic criteria were based on the relations of the generated form to certain proportion systems such as the golden section. EifForm was developed as conceptual and experimental tool. In practice it was used to generate some prototype installations.

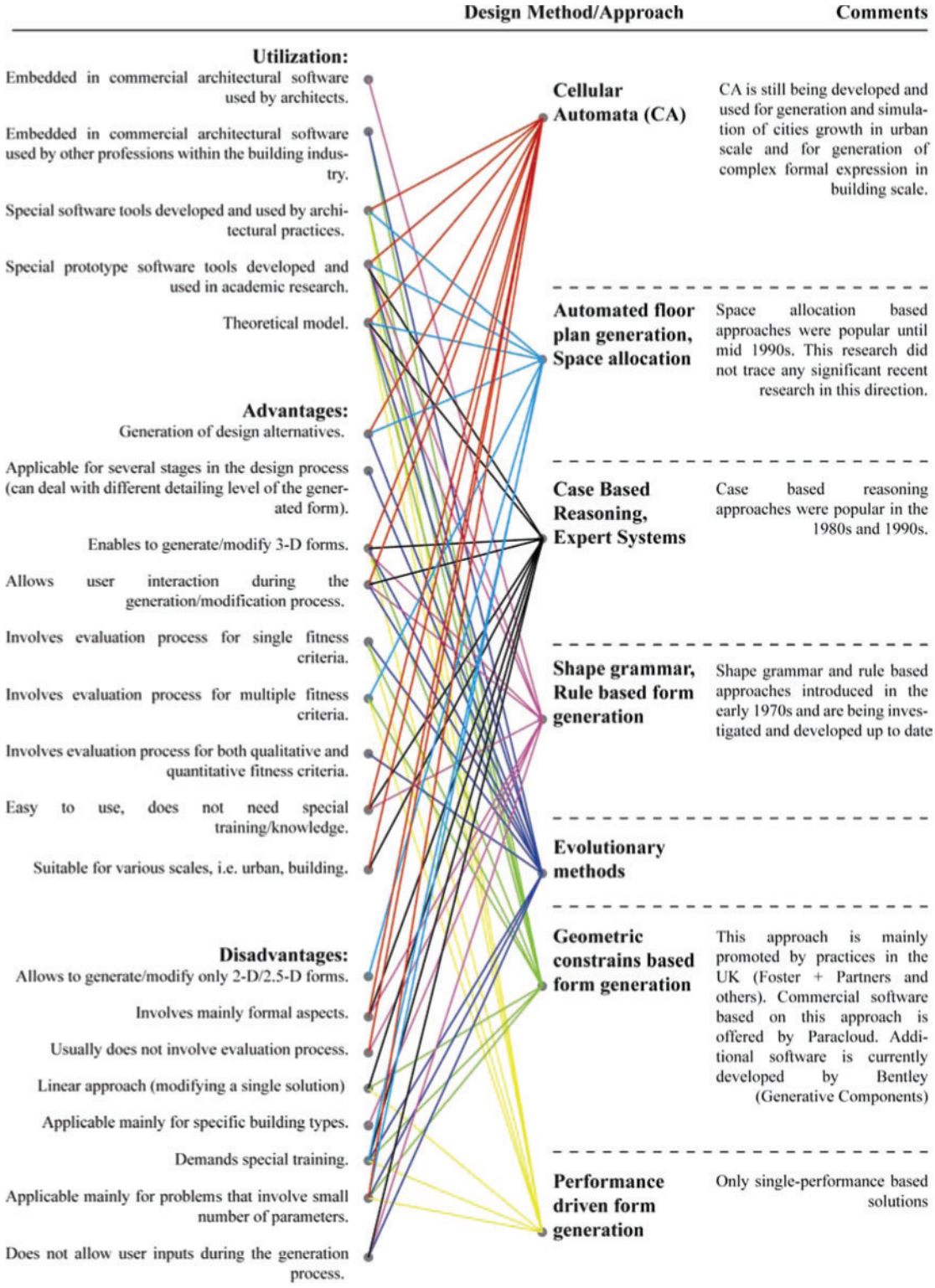
3. Summary of advantages and disadvantages of computer based form generation methods

The main advantages and disadvantages of the various computer-based form generation methods that were examined in this research are summarized in Figure 1.

4. Conclusions – gaps and direction worth pursuing in future research and practice in computer-based form generation

4.1. Use of computer code by architects

Using customized computer code in the design process can liberates the designer from the limitations imposed by the different design tools. In that respect, the commercial design tools can be perceived as a collection of preset codes and algorithms, designed to fulfill the need of the common design processes. The possibility of extending the capabilities of existing tools by means of custom applications and plug-ins allows the evolution of the simple user who uses only pre-defined applications, into a sophisticated user who knows how to adapt and transform the



▲ Figure 1. Evaluation of computer based form generation main approaches.

software according to his new needs. Writing custom code in that context can be regarded as part of the nonstandard approach that characterizes the digital revolution in architectural design in which there is no common design process and therefore, no possible common tool can be used. Scripting[11], in that sense, is a compromise between independent code writing and the employment of commercial computer software. In scripting, the designer is not limited by the present commands of the specific software, but by its mathematic and graphic engines. One of the possible implications of the transition to computer-based design is that the architects of the future will require a greater mastery of mathematics, geometry and computer code languages in order to be able to at least to adapt existing tools to their own needs and provide connections between them, if not to actually write new code. On the whole, these skills are not part of current architectural education [56]. Although this implication is not directly supported by the text it is clear that at least this will be a field in which many professionals will be working in the near future.

This situation raises some questions regarding the necessity of architects mastering code at all, as opposed to using professional programmers (similar to the way consultants are employed during the different design stages): should architectural education include mastering computer code at all, and if it does, what should be the level of code mastery architects have to achieve? Are we to expect a new expertise within the architecture profession - "codesman" as the new draftsman?

4.2. Design alternatives

One of the main advantages of computer–based form generation is the ability to use the computer processing power to allow ever-growing numbers of design alternatives to be created in a short time. Since it looks likely that complete (entire building) multi-criteria generation/optimization will not be possible in the near future, generation as design tools should be redirected toward design exploration and the development of alternatives and variations considering only selected aspects of design. This will boost the designers' creativity, rather than provide highly subjective answers to an oversimplified problem. Increasing creativity can also be attributed to the fact that computers can generate complex forms that could not be envisioned by human design[12]. Several alternative oriented design approaches were already suggested by Grobman et al. [57], Kim et al. [58] (for optimization in structural engineering), and by Frazer [42],

[11]Scripting in this context refers to both writing code and using visual code applications such as Grasshopper [69].

[12]See also Eisenman's argument on the possible benefits of using computer as a creative design tool [70].

Bentley [10] and Koutamanis [8] (within the frame of their research on evolutionary design and computer form generation).

4.3. Design evaluation

The idea of creating alternatives in design is directly connected to the notion of evaluation. Having generated design alternatives the designer has to choose the most appropriate alternative according to certain fitness criteria. Mitchell has already argued in 1992 that since design is "exploration of the interrelationships between beliefs and possibilities", form generation and optimization should be regarded as a boundary case [59]. Therefore, computer based architectural design has to involve computer-based evaluation modules. Since the evaluation of multiple criteria necessarily involves subjective decisions regarding the priorities and weighting of the various criteria, it would be logical to allow designers to change these priorities and compare the various solutions from the different evaluation runs. Moreover, it can be also argued that allowing for designers' inputs, which are based on experience and intuition, will increase creativity and narrow the design space to more accepted (for the designer) directions[13]. User inputs are starting to be embedded in software tools. A basic option for user inputs was embedded in Bentley's GADES optimization software [10] and suggested by Balcomb and Curtner in the MCDM software [60].

4.4. Differential accuracy

Since the level of information and details on the project's formal expression increases during the design process, generation/optimization tools must be able to adapt to the various levels of inputs and demands. This could be accomplished in two ways: The first is to use different algorithms for different stages of the design. These algorithms should be highly specific and presented to the designer as a menu of algorithms divided as to type of parameters, stages in the design and desired solution. Robinson et al [61] argued that this kind of approach would also save computer processing power since accurate optimization usually involves highly complex calculations and thus demands extensive processing power resources. This conclusion is supported by another argument suggesting that the processing power demands of intricate optimization/generation tools will always be close to technological limits, since it will develop concurrently with the increase in computer processing power. The second way calls for using the same tool in various stages of the design process by introducing "smart" preset or default values for data that has not been defined by the designer. This approach was demonstrated by Capeluto [62] and Papamichael [63].

[13]See Eckert et al [71] for a discussion on the possibility of human-computer synergy in generative design.

References

1. Kolarevic, B. Computing the Performative in Architecture. In: eCAADe 21. Graz, Austria: 2003.
2. Kolarevic, B. Digital Morphogenesis. In: Architecture in the Digital Age – Design and Manufacturing. London, New York: Spon Press; 2003. p. 26.
3. Kalay, Y.E. Architecture's New Media: Principles, Theories, and Methods of Computer-Aided Design. Cambridge, Mass: MIT Press; 2004.
4. Andia, A.S. Managing Technological Change in Architectural Practice: The Role of Computers in the Culture of Design. 1999
5. Venturi, R. Complexity and Contradiction in Architecture. 2nd ed. New York: Museum of Modern Art in association with the Graham Foundation for Advanced Studies in the Fine Arts, Chicago; 1977.
6. Negroponte, N. The Architecture Machine: Toward a More Human Environment. The MIT Press; 1973.
7. Schmitt, G. Design for Performance. In: Kalay YE, editor(s). Principles of Computer-Aided Design: Evaluating and Predicting Design Performance, New York: John Wiley & Sons; 2001. p. 87.
8. Koutamanis, A. Redirecting Design Generation in Architecture. In: Suddu C, editor(s). 3rd International Conference on Generative Art 2000. Roma: AleaDesign Publisher; 2000. p. 225-269.
9. Frew, S.R. A survey of space allocation algorithms in use in architectural design in the past twenty years. In: Proceedings of the 17th Conference on Design Automation. Minneapolis, Minnesota: 1980. p. 165-174.
10. Bentley, P.J. Evolutionary Design by Computers. 1st ed. Morgan Kaufmann; 1999.
11. Michalek, J, Choudhary, R. Papalambros, P. Architectural layout design optimization. Engineering Optimization. 2002;34(5):461-484.
12. Shaviv, E. Gali, D.A model for space allocation in complex buildings: A computer graphics approach. Build International. 1974;7493-517.
13. Steadman, M. The Geometry of environment: an introduction to spatial organization in design. Methuen London; 1971.
14. Arvin, S.A, House, D.H. Modeling architectural design objectives in physically based space planning. Automation in Construction. 2002 Feb ;11(2):213-225.
15. Rocker, I.M. When code matters. Architectural Design. 2006;76(4):16-25.
16. Chu, K. Planetary Automata. In: Open SA, Noever P, editor(s). The Gen[h]ome Project. Los Angeles, CA: MAK Center for Art and Architecture, Los Angeles; 2006. p. 32-37.
17. Silver, M. Building without drawings: Automason Ver 1.0. Architectural Design. 2006;76(4):46-51.
18. Oxman, R. Multiple operative and interactive modes in knowledge-based design systems. In: Kalay YE, editor(s). Principles of Computer-Aided Design: Evaluating and Predicting Design Performance. John Wiley & Sons; 1991.
19. Heylighen, A. Neuckermans, H. A case base of Case-Based Design tools for architecture. Computer-Aided Design. 2001 Dec;33(14):1111-1122.
20. Oxman, R. Expert Systems for Generation and Evaluation in Architectural Design. 1988;15-38.
21. Oosterhuis, K. Kas Oosterhuis: Programmable Architecture. L'Arcaedizioni; 2002.

22. Stiny, G. Gips, J. Shape grammars and the generative specification of painting and sculpture. In: Information Processing. Holand: Amsterdam: North-Holland; 1971. p. 1460-1465.

23. Stiny, G. Introduction to shapes and shape grammars. Environment and Planning B. 1980;7343-351.

24. Knight, T. Shape grammars in education and practice: History and prospects. International Journal of Design Computation. International Journal of Design Computation. 1999;2

25. McGill, M. A Visual Approach for Exploring Computational Design. 2001 ;

26. Wang, Y. Duarte, J.P. Automatic generation and fabrication of designs. Automation in Construction. 2002 Apr;11(3):291-302.

27. Tapia, M. A visual implementation of a shape grammar system. Environment and Planning B: Planning and Design. 1999;26(1):59-73.

28. Pak, B. Ozner, O. Erdem, A. Xp-GEN: A randomized design tool for non-deterministic digital design methods in architecture and visual design. In: eCAADe 21. Graz, Austria: 2003. p. 485-488.

29. Koning, H. Eizenberg, J. The language of the prairie: Frank Lloyd Wright's prairie houses. Environment and Planning B: Planning and Design. 1981;8(3):295-323.

30. Duarte, J.P. Customizing mass housing: A discursive grammar for Siza's Malagueira houses. 2001 ;

31. Shaviv, E. Gavish, O. Amir, U. Implementations of Solid Modeling in High Hierarchy Architectural Language. Planning and Design. 1990;17205-220.

32. Watanabe, M.S. Induction Design. 1st ed. Birkhäuser Basel; 2002.

33. Heisserman, J. Callahan, S. Mattikalli, R. A design representation to support automated design generation. In: Gero J, editor(s). Artificial Intelligence in Design. Kluwer Academic; 2002.

34. Kaos [Internet]. 2009 Jun 30;Available from: http://kaos-i.com/

35. Automason [Internet]. 2009 Jun 30;Available from: http://www.automason.com

36. Dollens, D. Digital-Botanic Architecture: D-B-A. Lumen Books; 2005.

37. Estevez, A.T. Truco, J. Felipe. S. Genetic Architectures II: Digital Tools & Organic Forms. Bilingual. Lumen Books/Sites Books; 2006.

38. Celestino, S. Generative Design Lab [Internet]. Available from: http://www.generativedesign.com/

39. Lynn, G. Animate Form. 1st ed. New York: Princeton Architectural Press; 1999.

40. Agarwal, M. Cagan, J. Constantine, K.G. Artificial Intelligence for Engineering. In: Analysis and Manufacturing 13. Cambridge, UK: Cambridge University Press; 1999. p. 241-251.[cited 2009 Feb 19]

41. Shea, K. Cagan, J. Languages and semantics of grammatical discrete structures. In: Analysis and Manufacturing 13. Cambridge, UK: Cambridge University Press; 1999. p. 241-251.[cited 2009 Feb 19]

42. Frazer, J. An Evolutionary Architecture. Architectural Association Publications; 1995.

43. Malkawi, A.M. Srinivasan, R.S. Kyu, Yi .Y. Choudhary, R. Performance-based design evolution: The use of genetic algorithms and CFD. In: Eighth International IBPSA. Eindhoven, Netherlands: 2003. p. 793-798.

44. Chu, K. The Turing Dimension [Internet]. 2000 ;Available from: http://www.archilab.org/public/2000/catalog/xkavya/xkavyaen.htm

45. Imperiale, A. New Flatness: Surface Tension in Architecture. Birkhauser Verlag AG; 2000.

46. Whitehead, H. Laws of Form. In: Kolarevic B, editor(s). Architecture in the Digital Age: Design and Manufacturing. Taylor & Francis; 2005.

47. Williams, C. Design by Algorithm. In: Leach PN, Turnbull D, Williams C, editor(s). Digital Tectonics. John Wiley & Sons; 2004. p. 79-87.

48. Nir, E. 2009 Jun 30; Available from: http://www.paracloud.com

49. Nir, E. Capeluto, I.G. Smart cloud-of-points model: Point-based digital media and tools for architectural design. In: education and research in COMPUTER AIDED ARCHITECTURAL DESIGN in Europe. Lisbon, Portugal: 2005. p. 687-694.

50. Smart Geometry group [Internet]. Available from: http://www.smartgeometry.com/

51. Rahim, A. Catalytic Formations. 1st ed. Taylor & Francis; 2005.

52. Sasaki, M. Ito, T. Isozaki, A. Morphogenesis of Flux Structure. London: AA Publications; 2007.

53. Capeluto, I.G. The influence of the urban environment on the availability of daylighting in office buildings in Israel. Building and Environment. 2003 May;38(5):745-752.

54. Capeluto, I.G. Shaviv, E. On the Use of Solar Volume for Determining the Urban Fabric. Solar Energy. 2001 ;70(3):275-280.

55. Shea, K. Directed Randomness. In: Leach PN, Turnbull D, Williams C, editor(s). Digital Tectonics. John Wiley & Sons; 2004. p. 96.

56. Gauchet, U. The $300,000/Year Architect. Architectural Design. 2009;79(2):32-37.

57. Grobman, J.Y. Yezioro, A. Capeluto, I.G. Building Form Generation Based On Multiple Performance Envelopes. In: 25th Passive and Low Energy Architecture Conference. Dublin: 2008.

58. Kim, H. Querin, O.M, Steven, G.P. On the Development of Structural Optimisation and its Relevance in Engineering Design. Design Studies. 2002 Jan;23(1):85-102.

59. Mitchell, W. The uses of inconsistency in design. In: Kalay YE, editor(s). Evaluating and Predicting Design Performance. Wiley-Interscience; 1992. p. 12.

60. Balcomb, J. Curtner, A. Multi-criteria decision-making process for buildings. Las Vegas, USA: 2000.

61. Robinson, G. El-Beltagy, M. Keane, A. Optimization in mechanical design. In: Bentley PJ, editor(s). Evolutionary Design by Computers. Morgan Kaufmann; 1999. p. 147-167.

62. Capeluto, I.G. A CAD System for Design and Evaluation of Solar Apartment Buildings. 1991;

63. Papamichael, K. Designers and information overload: a new approach. Advanced Buildings Newsletter. 1997;1(18):

64. Woodbury, R.F, Burrow, A.L. Whither design space? Artif. Intell. Eng. Des. Anal. Manuf. 2006;20(2):63-82.

65. Terzidis, K. Expressive Form: A Conceptual Approach to Computational Design. London: Spon Press; 2003.

66. Center for Advanced Spatial Analysis [Internet]. 2009 Jun 29; Available from: http://www.casa.ucl.ac.uk

67. Heylighen, A. In Case of Architectural Design: Critique and Praise of Case-Based Design in Architecture. 2000;

68. GENR8 [Internet]. 2009 Jun 30; Available from: http://projects.csail.mit.edu/emergentDesign/genr8/index.html

69. Grasshopper [Internet]. 2009 Jun 30; Available from: http://www.grasshopper3d.com

70. Eisenman, P. Visions Unfolding: Architecture in the Age of Electronic Media. Domus. 1992;(734):17-21.

71. Eckert, C. Kelly, I. Stacey, M. Artificial intelligence for engineering design. Analysis and Manufacturing. 1999;13303-320.

Yasha Jacob Grobman
Harvard University
Graduate School of Design
Gund Hall, 48 Quincy St, Cambridge
MAss 02138, USA

ygrobman@gsd.harvard.edu

Abraham Yezioro
Technion, Israel Institute of Technology
Faculty of Architecture and Town Planning
Haifa 32000, Israel

ayez@techunix.technion.ac.il

Isaac Guedi Capeluto
Technion, Israel Institute of Technology
Faculty of Architecture and Town Planning
Haifa 32000, Israel

arrguedi@technion.ac.il