Collection of Research Papers and Articles

Ganapathy Mahalingam, Ph.D.

THE ALGORITHMIC AUDITORIUM

A computational model for auditorium design

GANAPATHY MAHALINGAM
Department of Architecture and Landscape Architecture
North Dakota State University
Fargo, North Dakota
USA

Abstract. Auditorium design is a complex task. Various programmatic, functional and acoustical parameters have to be resolved in the spatial design of an auditorium. This ongoing research project deals with the development of a computer-aided design system for the preliminary spatial design of proscenium type auditoriums. The concept of "acoustic sculpting" is used to generate the spatial form of the auditorium from programmatic, functional and acoustical parameters. These parameters are incorporated using a combination of mathematical, empirical and statistical methods. The generation of the spatial form of the auditorium is implemented as an algorithm that is executed on the computer. The spatial form of the auditorium generated by the system is exported as a computer model for design development and acoustical analysis.

1. Introduction

Auditorium design is a complex task. Various programmatic, functional and acoustical parameters have to be resolved in the spatial design of the auditorium. The emergence of sophisticated computational modeling tools has now enabled the creation of design systems that treat the design of auditoriums as an algorithmic process. In this paper, the design of proscenium-type auditoriums is presented as an algorithmic process. This process is implemented in a design system where the generator of the spatial form of the auditorium is modeled as a "virtual computer."

2. Auditorium Design Parameters

The complexity of auditorium design arises from the need to resolve many interacting parameters. Some of the programmatic design parameters of the auditorium include the type of performance that is to be presented in the auditorium and the capacity of the auditorium. Programmatic parameters help decide the dimensions of stage enclosures and seating areas. Functional parameters include anthropometric constraints such as the area per seat, visual constraints such as sight lines, and conditions for visual clarity. However, the key parameters that influence the generation of the spatial form of the auditorium are the acoustical parameters. Acoustical parameters are integrated in the auditorium design system using the concept of acoustical sculpting.

3. Acoustic Sculpting

Acoustic sculpting is the creation of architectural shapes and forms based primarily on acoustical parameters. It can be likened to sculpting, not with a chisel, but with abstract entities such as acoustical parameters. Acoustical parameters become special abstract tools that shape the environment in their own characteristic way, hence the term acoustic sculpting.

In this context, it will be interesting to introduce the concept of a *locus*. In planar geometry, loci are lines traced by points according to certain rules or conditions. A circle is the locus of a point that is always equidistant from a given point. An ellipse is the locus of a point whose sum of distances from two given points is always equal. From these examples, it can be seen that a particular rule or condition can trace a particular locus. The scope of application of the concept of a locus can be dramatically widened by realizing that the word *locus* in Latin means *place*. Architecture involves the creation of places and spaces. A question can be posed - What is the locus of an acoustical parameter? In answering that question, architecture based on acoustical parameters can be created. Acoustics can become a form-giver for architecture. Figure 1 shows how the time delay gap, an acoustical parameter, is used to generate a semi-elliptical spatial field using the concept of the locus.

Acoustical parameters are often measured to assess the acoustical quality of a space or a scaled architectural model. They are indicators of the acoustical quality of the space in which they are measured. However, it is important to realize certain facts about acoustical parameters. Acoustical parameters are location specific. For a given sound source in a room, acoustical parameters vary systematically at different locations in the room. Acoustical parameters also vary when the sound source is varied. Hence, a set of acoustical parameters at a given location, for a specific sound source, can be used only to generate the general features of the architectural space around that location. Figure 2 shows the source-receiver locations used in the design system. This, to stay within the metaphor of sculpting, will result only in a first cut. Different sets of acoustical parameters from different locations can further refine the definition of the architectural space encompassing those locations. It has been found by researchers that at least 10 to 12 sets of acoustical parameters are required to derive the mean values of acoustical parameters in an auditorium (Bradley & Halliwell, 1989). If architectural shapes and forms can be created from acoustical parameters, then a rational basis can be established for the creation of acoustical environments.

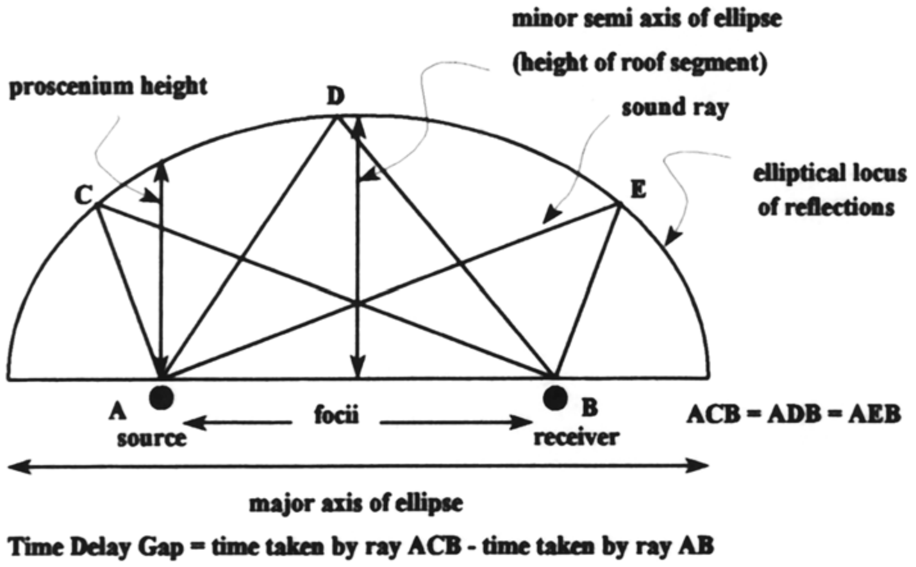

minor semi axis of ellipse
(height of roof segment)
proscenium height
D
sound ray
C
E
elliptical locus of reflections
A
source
focii
B
receiver
ACB = ADB = AEB
major axis of ellipse
Time Delay Gap = time taken by ray ACB - time taken by ray AB

Figure 1. Concept of locus used to derive a spatial field from an acoustical parameter.

Currently, the creation of acoustical environments is a trial-and-error process that tries to match the acoustical parameters of the space being created, probably in the form of a physical model, with acoustical parameters that have been observed in other well-liked spaces. The manipulations of the space's shape and form to achieve the match, are done in an arbitrary fashion, with no explicit understanding of the relationships between the shape and form of the space and the corresponding acoustical parameters. There has been extensive research conducted in the 1960s, 1970s and 1980s by Ando (1985), Barron (1988), Barron & Lee (1988), Beranek (1962), Bradley (1986, 1990), Cremer (1978), Hawkes (1971) and Sabine (1964) to establish those aspects of the auditory experience that are important in the perception of the acoustical quality of a space, and how they relate to objectively measured acoustical parameters in that space. There has not been much research conducted except by Gade (1986, 1989) and Chiang (1994) regarding the relationships between acoustical parameters and the shapes and forms of the spaces in which they are generated.

Acoustic sculpting attempts to define the latter relationships and uses them to create a system that generates spatial forms of auditoriums based on acoustical parameters. This generative system is used as a tool for creating preliminary designs of proscenium-type auditoriums.

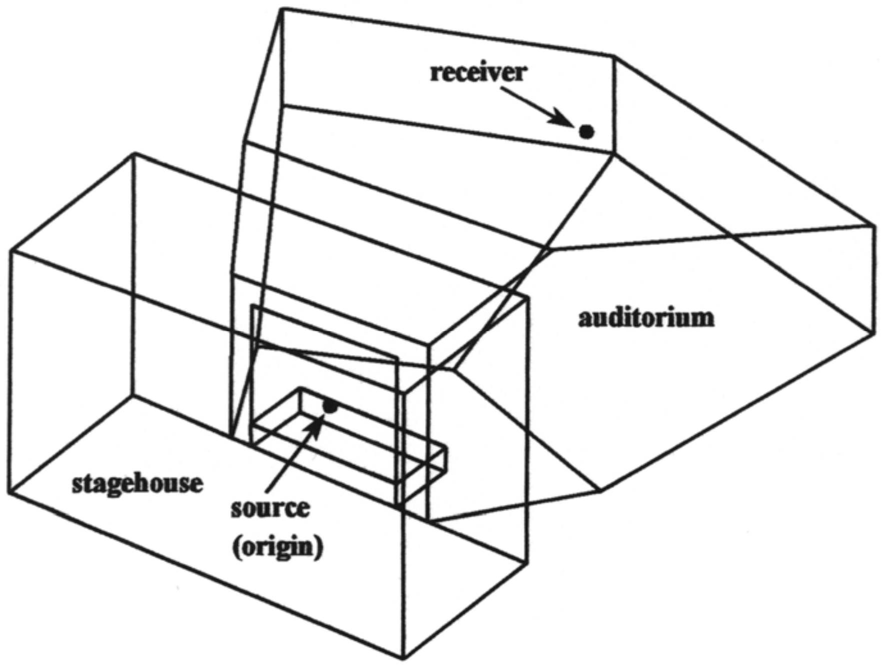


Figure 2. Spatial form of the auditorium showing the source-receiver pair for acoustical parameters.

3.1. METHODS OF ACOUSTIC SCULPTING

The process of generation of the spatial form of the auditoriums is related to the set of acoustical parameters both statistically and theoretically. The acoustical parameters for the generative system are drawn from, but are not limited to, the set presented in the following section. This set of parameters is used by acousticians to study concert hall and lecture room acoustics. These parameters are derived from response graphs of sound intensity variations at the receiving location. Figure 3 shows a response graph. Though the set is extensive, not all of the parameters are used in the spatial form generation stage.

3.1.1. Acoustical Parameters

The acoustical parameters include Reverberation Time, Early Decay Time, Room Constant, Overall Loudness or Strength of Sound Source, Initial Time Delay Gap, Temporal Energy Ratios: Early/Total Energy Ratio (Deutlichkeit), Early/Late Energy Ratio (Clarity), Center Time, Lateral Energy Fraction, Spatial Impression, Bass Ratio, Bass Level Balance, Early Decay Time Ratio and Center Time Ratio, Useful/Detrimental Ratio, Speech Transmission Index and the Rapid Speech Transmission Index.

The different acoustical parameters cited above resolve into related groups that have corresponding subjective perception characteristics. These subjective perception characteristics are classified as Reverberance, Loudness, Clarity, Balance and Envelopment.

A limited set of acoustical parameters related to these subjective perceptions are incorporated in the system (using both statistical and theoretical methods) that derives architectural parameters from the acoustical parameters. It must be remembered that, in the spatial form generation stage, acoustical parameters are not the only factors determining the shapes and forms of the auditoriums. Other factors like seating requirements, visual constraints and other programmatic requirements, along with the acoustical parameters, determine the spatial forms of the auditoriums. The values of the acoustical parameters for use in the generative system are drawn from a database of objectively measured readings in different architectural settings that have been subjectively evaluated as desirable. Based on studies done so far, a generative system based on macrostatic statistical relationships and some analytical theory has been developed by the author. Details of this system are to be found in another paper by the author (Mahalingam, 1992).

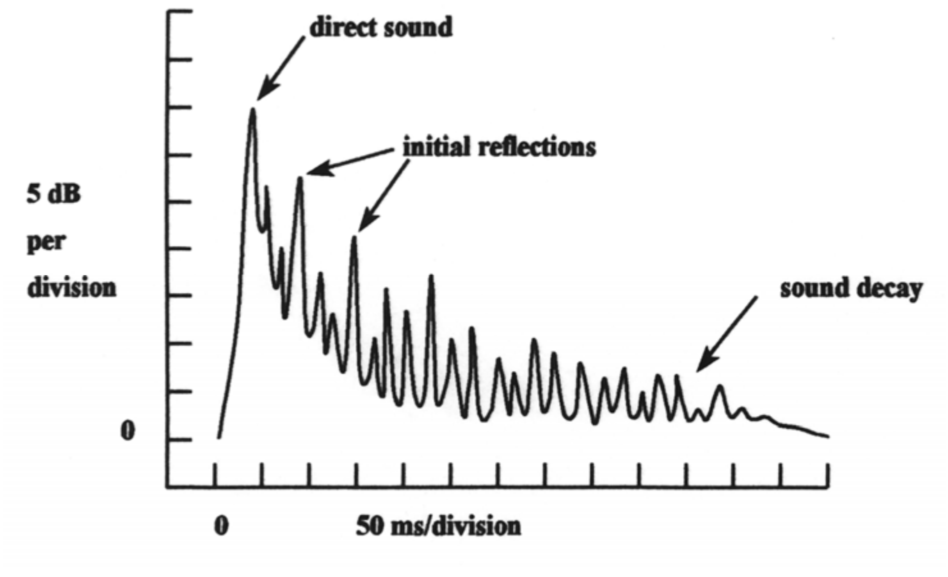


Figure 3. Response graph showing sound intensity variation over time at receiver location.

4. Spatial Model Of The Auditorium

The spatial form of the auditorium is modeled as a parametric object. Each vertex that makes up the topology of the auditorium is spatially located by a function of multiple parameters. These parameters may be directly input by the user of the design system or derived from the user input using calculations. The various parameters are linked in a spatial form generating algorithm using a structure that resembles an ASIC (application specific integrated circuit). Figure 4 shows this relationship of the various parameters. This structure can also be reconfigured as a network or semi-lattice. The connectivity of the vertices that establishes the topology of the auditorium is derived from the spatial type of the proscenium auditorium. The whole design system is a “virtual computer” that outputs spatial designs of proscenium-type auditoriums.

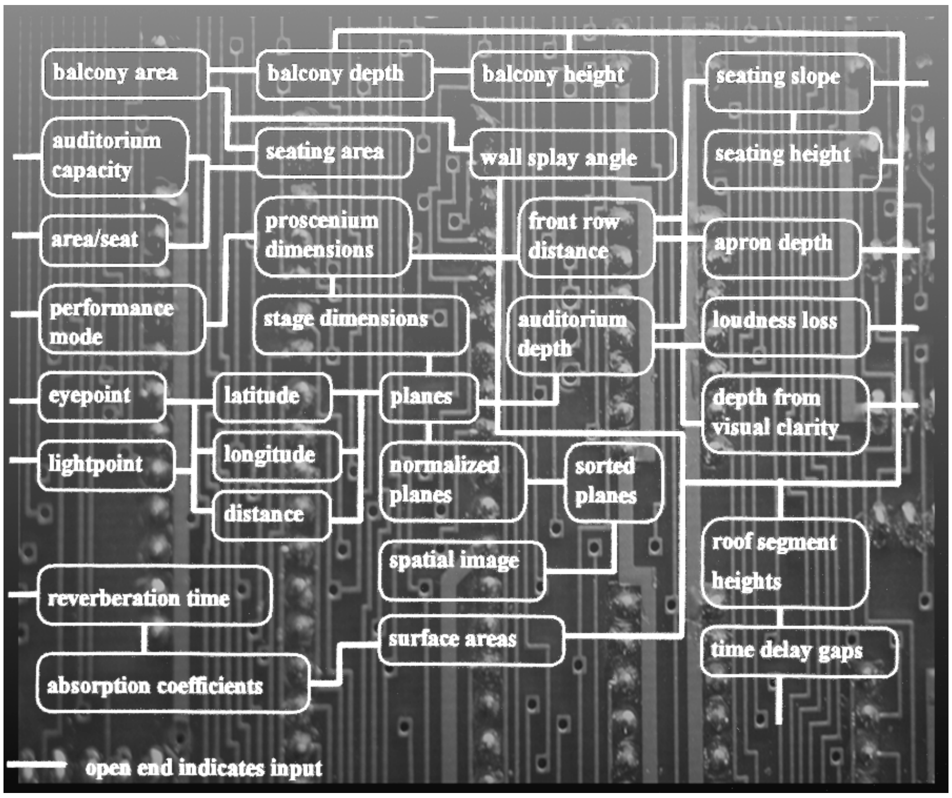


Figure 4. The relationship of the various parameters of the spatial form generator as an ASIC (application specific integrated circuit)

5. What Is A Virtual Computer?

In object-oriented computing, entities are modeled as encapsulations of data, and operations that can be performed on that data. Encapsulation is a computer abstraction. A collection of data and operations normally performed on the data are closely related, so, they are treated as a single entity (rather than separate) for purposes of abstraction. Each encapsulation can be thought of as a virtual computer that is mapped onto a physical computer (see Figure 5) with its own private memory (its data) and instruction set (its operations). The reference to objects as computers was made by Alan Kay (1977). He envisaged a host computer being broken down into thousands of computers (virtual?), each having the capabilities of the whole, and exhibiting a certain behavior when sent a message which is a part of its instruction set. He called these (virtual?) computers "activities." According to him, object-oriented systems should be nothing but dynamically communicating "activities." As such they form an interesting model with which to simulate architectural design. Mitchell's recent call (1994) for a "society of design" with a "collection of agents of different kinds interacting over a network" echoes the ideas of Alan Kay. In another interesting perspective, encapsulations have been likened to integrated circuits rather than virtual computers by Ledbetter & Cox (1985) (see Figure 4).

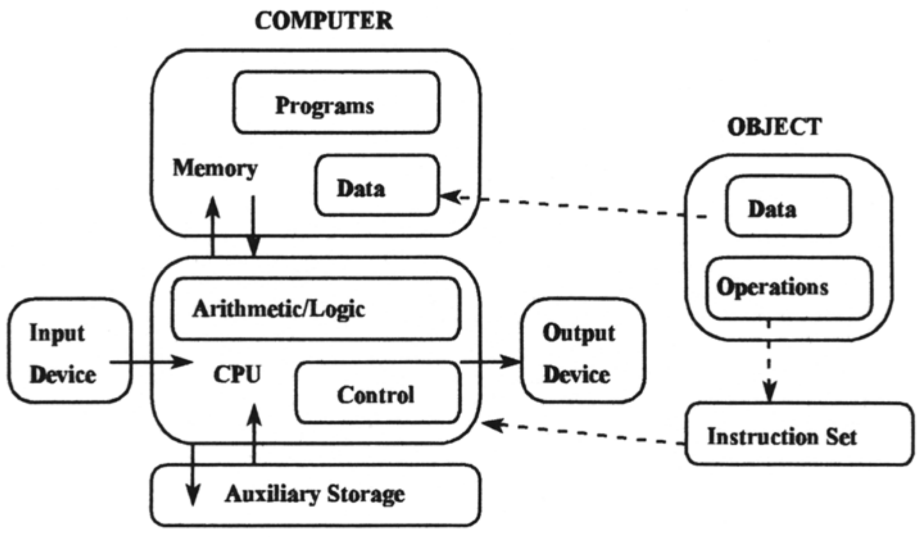


Figure 5. The concept of a virtual computer (or computational object) being mapped onto a physical computer.

6. The Auditorium Design System

The design system used to generate the preliminary spatial designs of proscenium type auditoriums is based on acoustical, functional and programmatic parameters. The computational model of the auditorium is parametric. The various acoustical, functional and programmatic parameters are its data. Procedures that compute the spatial parameters of the auditorium and create its graphic representation are its operations. These data and operations, when encapsulated, act as a virtual computer that is mapped onto the physical computer (see Figure 5). The function of this virtual computer is to output auditorium designs.

The generative system involves an algorithmic procedure for the design of the auditoriums based on constants, user input of independent variables and derived variables. These constants and variables are used to calculate the spatial location of sets of vertices in 3D space that are linked to form wire-frame and shaded plane images of the auditoriums. The topology of the auditorium is based on the proscenium-type auditorium typology. It is a variant topology with the introduction of balconies only when necessary (see Figure 6). The vertices are parametrically controlled and change with changing parametric inputs.

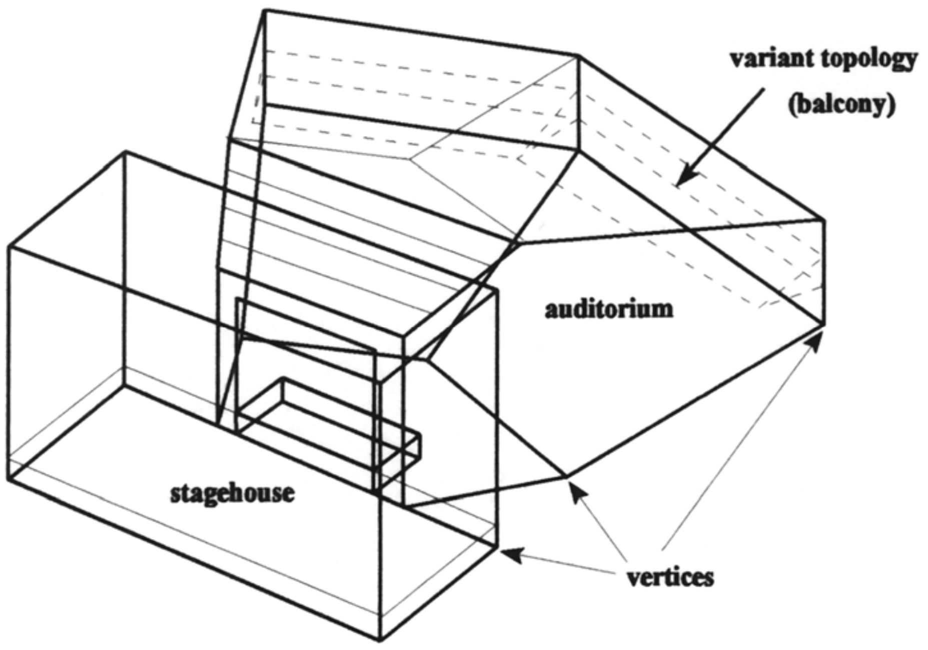
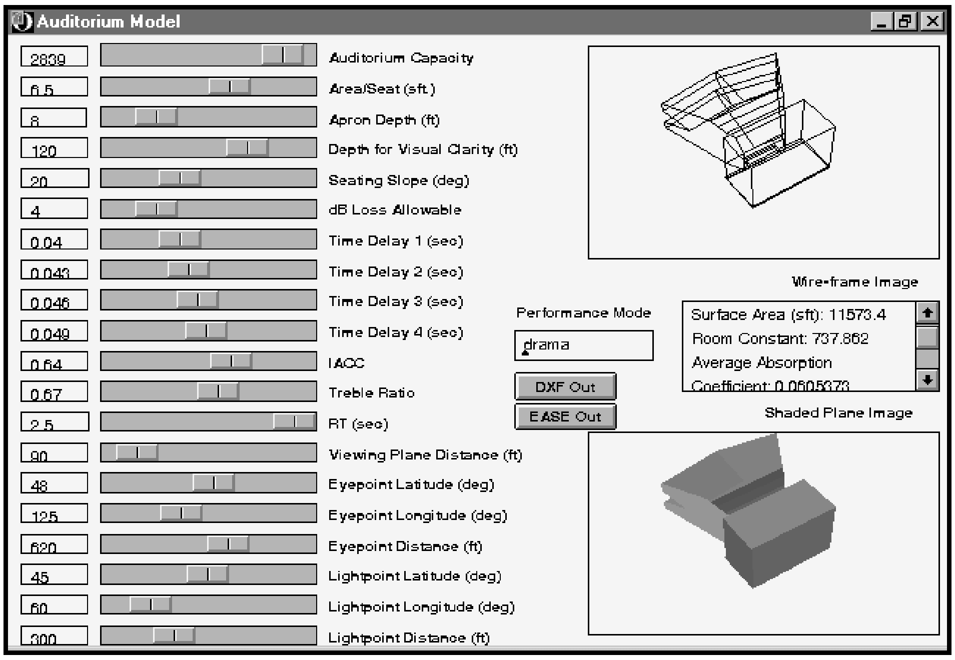


Figure 6. Topological model of the auditorium showing the variant topology for the balconies.

The algorithmic procedure is implemented in the Smalltalk^{TM} object-oriented programming language. The software has a user-friendly menu and graphic interface with which to input acoustical, functional and programmatic parameters. When any aspect of the model is changed, the spatial form is updated. The system provides a dynamic design environment. In the system, the spatial form changes in real time with changing input of the parameters. The auditorium is depicted in true perspective. Once the spatial form is generated, it can be viewed from any angle and from any distance. The systems can be used to rapidly generate alternate designs based on the various parameters.

To limit the scope of the software design to manageable limits, the initial version of the generative system has a limited set of 21 independent variables. However, the total number of variables (both independent and derived) in the system is large, indicating a complex system. An interface has been developed that can transfer the computer model generated by this system in a format readily acceptable by commercial CAD packages (the DXF format) for design development. An interface has also been developed to link this system to acoustical simulation software (EASE and EARS) to predict what the auditorium will sound like if it is built. The view of the computer screen when running the software is shown in Figure 7.



2839	Auditorium Capacity
6.5	Area/Seat (sft)
8	Apron Depth (ft)
120	Depth for Visual Clarity (ft)
20	Seating Slope (deg)
4	dB Loss Allowable
0.04	Time Delay 1 (sec)
0.043	Time Delay 2 (sec)
0.046	Time Delay 3 (sec)
0.049	Time Delay 4 (sec)
0.64	IACC
0.67	Treble Ratio
2.5	RT (sec)
90	Viewing Plane Distance (ft)
48	Eyepoint Latitude (deg)
125	Eyepoint Longitude (deg)
620	Eyepoint Distance (ft)
45	Lightpoint Latitude (deg)
60	Lightpoint Longitude (deg)
300	Lightpoint Distance (ft)

Performance Mode: drama

Surface Area (sft): 11573.4
Room Constant: 737.862
Average Absorption Coefficient: 0.0605373

Figure 7. View of the screen of the design system software for auditoriums.

Acknowledgements

Much of this research work was completed as part of the author's doctoral dissertation under the supervision of Professors John Alexander and Gary Siebein at the University of Florida, Gainesville, USA.

References

Ando, Y.: 1985, *Concert Hall Acoustics*, Springer-Verlag, Berlin.

Barron, M.: 1988, Subjective Study of British Symphony Concert Halls, Acustica, Vol.66, No.1, pp. 1-14.

Barron, M. and Lee J.: 1988, Energy Relations in Concert Auditoriums, *Journal of the Acoustical Society of America*, Vol. 84, No. 2, pp. 618-628.

Beranek, L. L.: 1962, *Music, Acoustics and Architecture*, John Wiley & Sons, New York.

Bradley, J. S.: 1990, The Evolution Of Newer Auditorium Acoustics Measures, *Canadian Acoustics*, Vol.18, No.4, pp. 13-23.

Bradley, J. S.: 1986, Auditorium acoustics measures from pistol shots, *Journal of the Acoustical Society of America*, Vol.80, No.1, pp. 199-205.

Bradley, J. S. and Halliwell R. E.: 1989, Making Auditorium Acoustics More Quantitative, *Sound and Vibration*, February, pp. 16-23.

Chiang, Wei-Hwa: 1994, Effects of Various Architectural Parameters on Six Room Acoustical Measures in Auditoria, Ph.D. Dissertation, University of Florida, Gainesville.

Cremer, L.: 1978, *Principles and Applications of Room Acoustics*, Vol. 1, (Translated by T. Schultz), Applied Science Publishers, London, England.

Gade, A. C.: 1989, Acoustical Survey of Eleven European Concert Halls, Report No. 44, The Acoustics Laboratory, Technical University of Denmark, Lyngby.

Gade, A. C.: 1986, Relationships Between Objective Room Acoustic Parameters And Concert Hall Design, Proceedings of the 12th International Congress on Acoustics, Vol./Band D-G, E4-8, Toronto.

Hawkes, R. J.: 1971, Experience in Concert Auditoria, *Acustica*, Vol. 24, pp. 236-250.

Kay, A. C.: 1977, Microelectronics and the Personal Computer, *Scientific American*, September, pp. 230-244.

Ledbetter, L. and Cox B.: 1985, Software-ICs, in *Byte*, June, pp. 307-316.

Mahalingam, G.: 1992, Designing the Sound Environment: Acoustic Sculpting, Proceedings of the ACSA Technology Conference, San Diego, February.

Mitchell, W. J.: 1994, "Three paradigms for computer-aided design," in *Automation in Construction*, Vol. 3, Numbers 2-3.

Sabine, W. C.: 1964, *Collected Papers on Acoustics*, Harvard University Press, Cambridge, Massachusetts, Reprinted by Dover Publications.

The Algorithmic Auditorium: Automating Auditorium Design

Ganapathy Mahalingam, Ph.D.
North Dakota State University

Abstract

The goal of this ongoing research project is the development of an algorithm for the process of auditorium design. A version of this algorithm has been implemented in a computer-based design system for the preliminary spatial design of proscenium-type auditoriums. The spatial form of the auditorium is derived from acoustical, functional and programmatic parameters. Each of these parameters implies an appropriate spatial form for the auditorium. The algorithm resolves the spatial implications of these multiple parameters to arrive at an "optimized" form of the auditorium. The design system is to be used by architects as a performance-based preliminary design tool. The process of generating the spatial form of the auditorium from architectural acoustical parameters is called acoustic sculpting. Acoustic sculpting is a type of performance-based design. Performance loci of the various parameters drive the form generation. This concept can be extended to include lighting and HVAC parameters as well in spatial form generation. This approach will eventually lead to the generation of various architectural spaces based on environmental performance criteria. The design of architectural spaces will then be based on tracing the loci of programmatic, functional and environmental performance parameters in addition to visual considerations. Current research work is focused on developing algorithms to derive acoustical parameters from the spatial forms generated by the design system. This will complete the cycle of inquiry to see if the forms that are generated yield the acoustical parameters on which they were based. Since similar measurements of acoustical parameters can be made in different spaces, this research will most likely not yield a deterministic design system but an effective design system. The successful implementation of an algorithmic process to design auditoriums also addresses the issue of the computability of architectural design. If auditorium design can be automated, then there is hope yet for the computability of architectural design.

Introduction

The algorithmic auditorium project (Mahalingam, 1998) started with two main premises. One was that you could automate auditorium design by developing an algorithmic process to generate the spatial form of the auditorium. The other was that you could generate spatial forms based on acoustical, functional and programmatic parameters. Both these premises were realized in the development of a design system for the preliminary spatial design of proscenium-type auditoriums (Mahalingam, 1995).

The characterization of the system as a design system can be called into question. To resolve the issue of whether computers, or more appropriately, computer-based systems, can design, an Architectural Turing Test should be used. Take this scenario in an architect's office. The principal walks into the studio and gives a staff architect the task of generating the initial spatial form of a proscenium-type auditorium. The architect is given general requirements such as the seating capacity, area per seat, the performance type, and acoustical parameters such as reverberation time. The principal walks away. The architect sets to work. After 12 hours, she has drawn a perspective drawing of the initial spatial form of the auditorium, after resolving issues such as volume, seating areas, sight lines, seating slopes and sound reflection panels. Has she designed the auditorium? If she has, then a computer-based system that does what she has done is also designing. This is the basis of an Architectural Turing Test. Given the same input, if a computer-based system produces the same output as a human designer, and the human is considered to be designing, then the computer-based system can be said to be designing as well.

In the design system, non-spatial information is converted to spatial form by an algorithm. This can be said to be the essence of computer-based architectural design, or for that matter, any process of designing material artifacts. This system proves that certain non-trivial design tasks can be automated, therefore, other non-trivial design tasks can also be automated if they are computationally articulated. The auditorium design system is based on performance criteria and their translation into spatial form. It was possible to create this system, because it was possible to computationally articulate the decision making in auditorium design. Rather than focus on the design behavior of the architect in the auditorium design process, the focus was placed on design decision making. This approach acknowledges that designs are created by a sequence of explicit decisions.   Design behavior is a larger rubric that surrounds design decision-making.   Protocol analyses of design behavior often lead to unresolvable complexity in the articulation of design because they do not focus on the decision-making process. Protocol analyses can lead to computational articulation if they focus on the design decisions that are actually made.

Main Concepts

The single main concept used in the design system is acoustic sculpting (Mahalingam, 1992). Acoustic sculpting is the process by which spatial form is generated from architectural acoustical parameters. The sculpting analogy was used because the use of a particular tool for sculpting yields characteristic results. Sculptors use stone chisels, wood carving implements, blowtorches and lasers to sculpt material. What if sculptors working in digital media use abstract mathematical tools to sculpt forms? What if the sculptor's material was architectural space and the abstract mathematical tools were environmental performance criteria?

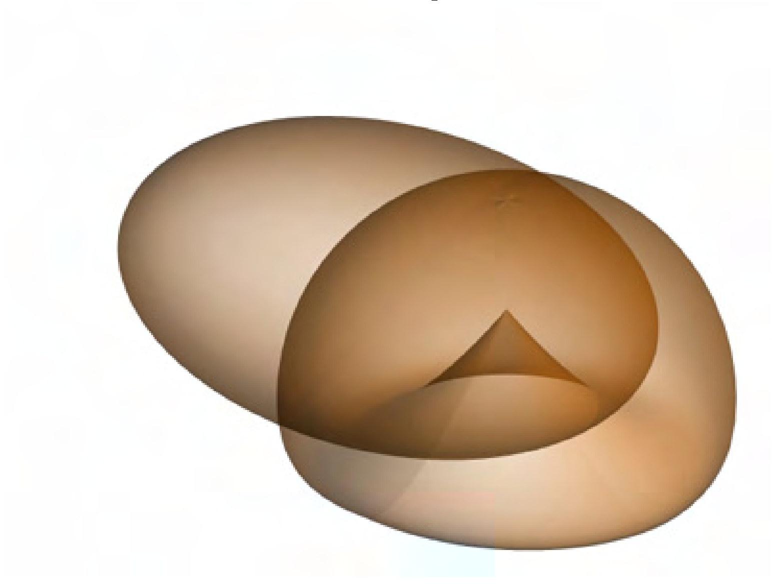


Figure 1. Two performance loci, an elliptical field for the initial time delay gap and an isocandle envelope.

Each acoustical parameter has its performance locus, a spatial form in which the acoustical parameter is generated. For example, the performance locus of the initial time delay gap (an acoustical parameter) is an elliptical spatial field. The initial time delay gap is the difference in time between the time taken by the first reflected ray to arrive at a receiver location and the time taken by the direct ray from the sound source. The initial time delay gap's performance locus is an elliptical field because an ellipse is the locus of a point that moves such that the sum of its distances from two fixed points is constant. The two fixed points are the source and the receiver locations. Similarly, the performance locus of an area of seating that minimizes distances from a point source is a segment of a circle. Deriving the performance locus of a parameter can be based on geometrical, mathematical or statistical analysis. For example, in the design system, some of the spatial implications of acoustical parameters were established by performing regression analysis between architectural dimensions and acoustical parameters recorded in various spaces. The use of performance loci to generate spatial form is a powerful concept. Performance loci of many environmental performance criteria including lighting and HVAC parameters can become form-givers for architectural spaces. Performance loci are a means to derive spatial form from non-spatial information. Multiple performance loci can be resolved into an optimal spatial form using Boolean operations such as intersection (see Figs. 1 & 2). The common space occupied by various performance loci becomes the optimal architectural space. This is similar to the constraint envelopes used in three-dimensional constraint-based reasoning.

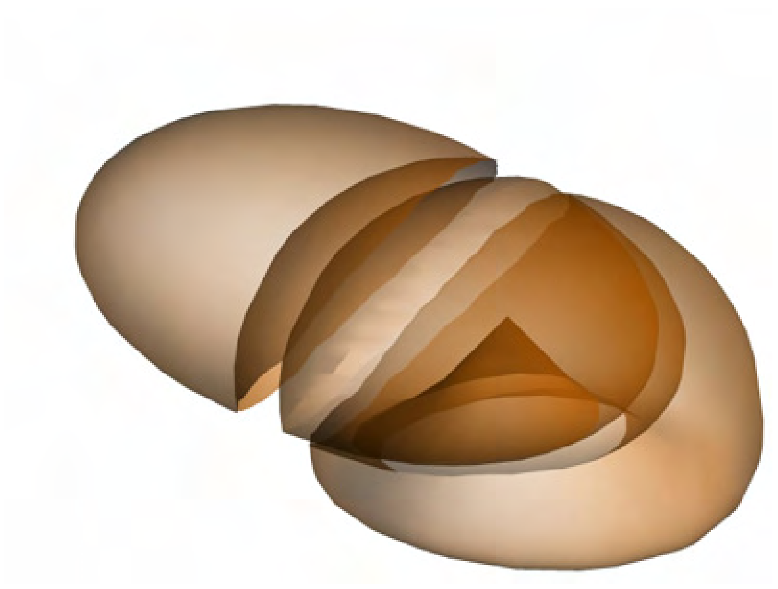


Figure 2. The common optimal space between the two performance loci obtained through the Boolean operation of intersection

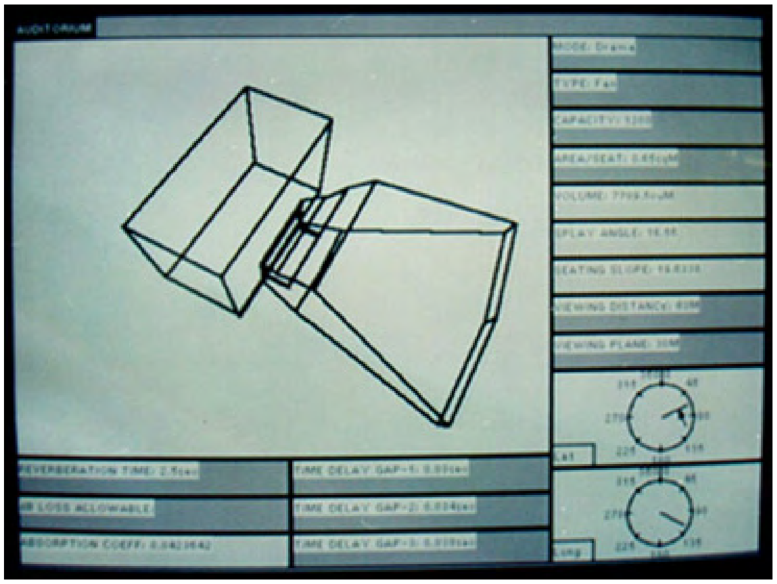
Gyorgy Doczi (1981) wrote an important book on the power of limits in which he showed how many forms found in nature and in cultural artifacts were the results of "dinergy" patterns. Doczi showed that the intersection of "dinergy" patterns and their limiting conditions generated various forms of flowers, leaves, shells and fish. Doczi described these patterns as the creative energy of organic growth. Performance loci are like "dinergy" patterns that can be used to generate a new breed of organic architectural spaces. Greg Lynn (1999) uses the concept of animate form, but primarily deals with the transformation (or more accurately, deformation) of architectural forms based on external contextual forces using the "affector" technology available in special effects systems. Lynn's forms accommodate functions in unique and innovative ways, but the form generation process itself is not governed by environmental performance criteria. Animate forms will acquire more power if the transformation or deformation of the form being designed is being done by programmatic, functional and environmental performance criteria. Rather than being a pliant form acted on by external forces, architectural space should unfold as a response to performance criteria.

Implementation

The initial version of the auditorium design system (see Fig. 3) was implemented in 1991 using the object-oriented software development environment ObjectWorks^{TM}. The programming language used was Smalltalk. The current version was developed using the VisualWorks^{TM} software development environment, which is also based on Smalltalk (see Fig. 4). The system runs on the VisualWorks^{TM} virtual machine and is platform independent. It can run on PCs, Macintoshes and Unix machines. The design system is modeled as a virtual computer. The various parameters are its input, the spatial form of the auditorium is its output, and the algorithmic process is like an integrated circuit in the processing unit.

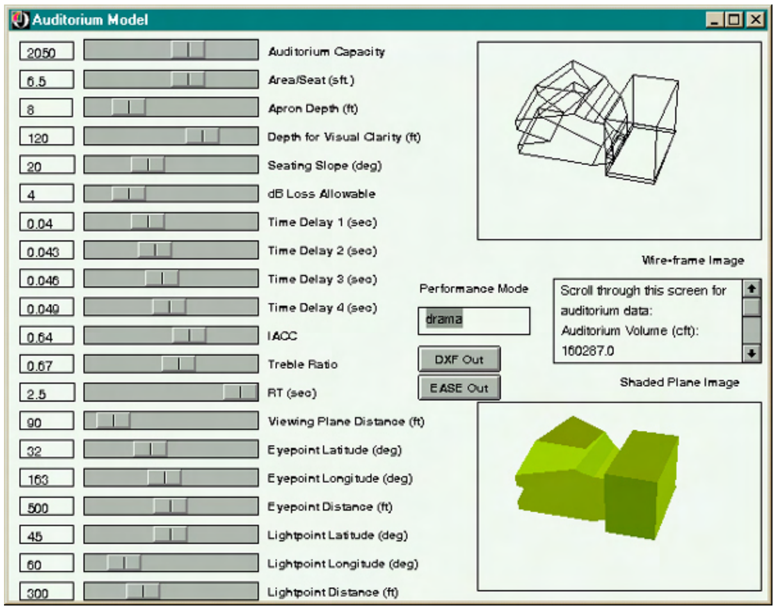
The spatial form of the auditorium is a collection of vertices. Each vertex is spatially located by a function of acoustical, functional and programmatic parameters. The vertices are connected based on the typology of the proscenium-type auditorium. The topology of the spatial form is provided by the typology. This topology generates the coupled space configuration of the stage house and the auditorium. The topology is not a fixed topology. For instance, balconies are embedded in the initial topology only to be activated when the parameters warrant it. The whole spatial form of the auditorium collapses to a point, a singularity, when the various parameters are set to zero. The embedded topology of the balcony collapses into the initial topology when not needed. Connecting appropriate vertices creates the different surfaces of the auditorium. Currently an algorithm is being developed (Mahalingam, 1999) that models sound propagation in the auditorium as radiation from surface to surface. This model of diffuse sound propagation allows for a quick assessment of the acoustical parameters of the computer model of the auditorium.

The spatial form of the auditorium generated by the design system can be exported in file formats accepted by commercial CAD (AutoCAD^{TM}) and acoustical analysis (EASE/EARS^{TM}) software. This allows an architect using the system to further develop and articulate the spatial form of the auditorium generated by the design system. In the design development stage, the spatial form generated by the design system will be the constraining envelope because it is based on the various performance criteria.



MODE: Drama
TYPE: Fan
CAPACITY: [illegible]
AREA/SEAT: 0.65sqM
VOLUME: 7709.5cuM
SPLAY ANGLE: 16.55
SEATING SLOPE: 16.8338
VIEWING DISTANCE: 83M
VIEWING PLANE: 30M
REVERBERATION TIME: 2.5sec
dB LOSS ALLOWABLE
ABSORPTION COEFF: 0.0423642
TIME DELAY GAP-1: 0.03sec
TIME DELAY GAP-2: 0.034sec
TIME DELAY GAP-3: 0.035sec

Figure 3. Initial version of the auditorium design system from 1991.



2050	Auditorium Capacity
6.5	Area/Seat (sft)
8	Apron Depth (ft)
120	Depth for Visual Clarity (ft)
20	Seating Slope (deg)
4	dB Loss Allowable
0.04	Time Delay 1 (sec)
0.043	Time Delay 2 (sec)
0.046	Time Delay 3 (sec)
0.049	Time Delay 4 (sec)
0.64	IACC
0.67	Treble Ratio
2.5	RT (sec)
90	Viewing Plane Distance (ft)
32	Eyepoint Latitude (deg)
163	Eyepoint Longitude (deg)
500	Eyepoint Distance (ft)
45	Lightpoint Latitude (deg)
60	Lightpoint Longitude (deg)
300	Lightpoint Distance (ft)

Performance Mode: drama

Scroll through this screen for auditorium data:
Auditorium Volume (cft):
160287.0

Figure 4. Current version of the auditorium design system.

Concluding Thoughts

Acoustic sculpting addresses the generation of spatial forms from acoustical parameters. Similar spatial form generation techniques can be applied to lighting and HVAC parameters. Just as we can ask what the performance locus or spatial form of an acoustical parameter is, so we can ask what the performance locus or spatial form of a lighting or HVAC parameter is! Architectural space generation will become a supple manipulation and visual (?) resolution of the elliptical spatial fields of time delay gaps, isocandle envelopes and isothermal bubbles! Generating optimal forms from performance loci will be the challenge. One can begin the design process with performance loci, yet the final optimization of the spatial form can still be visual, which however, may now be difficult to justify with the spatial articulation of other performance criteria.

References

Doczi, G. (1981), *The Power of Limits*, Shambala Publications Inc., Boulder, Colorado, and London, England.

Lynn, G., (1999), *Animate Form*, Princeton Press, New York.

Mahalingam, G. (1999), “A New Algorithm for the Simulation of Sound Propagation in Spatial Enclosures,” Building Simulation '99 Conference, Kyoto, Japan.

Mahalingam, G., (1998), *The Algorithmic Auditorium*, Initiative for Architectural Research, Research Poster Competition Winner.

Mahalingam, G. (1995), *The Application of Object-oriented Computing in the Development of Design Systems for Auditoria*, Ph.D. Dissertation, University of Florida, Gainesville, Florida.

Mahalingam, G., (1992), "Designing the Sound Environment: Acoustic Sculpting," ACSA Technology Conference, San Diego.

A COMPUTATIONAL MODEL OF A SENSOR NETWORK FOR THE OPTIMIZATION AND CONTROL OF ACOUSTICAL PERFORMANCE CRITERIA IN SPATIAL ENCLOSURES

GANAPATHY MAHALINGAM
North Dakota State University, Fargo, North Dakota, U.S.A.
Email address:Ganapathy.Mahalingam@ndsu.edu

Abstract. The technology of nanoblock-based circuits is enabling the creation of ultra-small sensors that can transmit their location and readings using radio frequencies. This technology has the potential to enhance the optimization of environmental performance criteria in spatial enclosures, using a wide range of actuators that function at different scales. The combination of such ultra-small sensors with actuators will enable the creation of spatial enclosures that are complex adaptive systems, which dynamically optimize the various environmental performance criteria for the enclosures. Defining a model for the optimization process in these systems presents significant challenges. This paper will set out a model for the use of ultra-small sensor systems in optimizing environmental performance criteria in spatial enclosures, especially acoustical performance criteria.

1. Introduction

The technology of nanoblock-based circuits is enabling the creation of ultra-small sensors that can transmit their location and readings using radio frequencies. This technology has the potential to enhance the optimization of environmental performance criteria in spatial enclosures, using a wide range of actuators that function at different scales. These actuators range from material modifiers that use piezoelectric effects to micro-electrical mechanical systems to motors that move large panels. The combination of such ultra-small sensors with actuators will enable the creation of spatial enclosures that are complex adaptive systems, which dynamically optimize the various environmental performance criteria for the enclosures. Defining a model for the optimization process in these systems presents significant challenges. This paper will set out a model and relevant challenges for the use of ultra-small sensor systems that are combined with actuators to optimize environmental performance criteria in spatial enclosures. This model will enable the consideration of architectural space as a dynamic, adaptive entity rather than as a static entity, which has been the traditional approach. This model also has the potential to serve a broader range of optimization problems in other contexts as well.

2. Sensors and Effectors

2.1 NANOBLOCKS AND ULTRA-SMALL SENSORS

Nanoblocks are substrates for circuit components at the scale of nanometers. These nanoblocks are the size of pepper flakes. These blocks are combined into ultra-small circuits that can function as a sensor or be part of an actuator. A sensor measures some environmental criterion. An actuator takes the measurement or reading of a sensor and performs an action through electrical and mechanically driven devices. An actuator may have a digital signal processor as part of its configuration. These nanoblock-based ultra-small sensors can be so ubiquitous as to form a coat of 'sensor paint' on surfaces of spatial enclosures.

2.2 SENSOR ARRAYS AND NETWORKS

Since the ultra-small sensors are so small that thousands could be placed in one square inch, their individual readings may be so close to each other that they are not significantly different from each other. This allows the combination of many sensors into sensor arrays using statistical techniques such as cluster analysis. Cluster analysis is a statistical technique that groups entities together such that the within-group variation is minimized and between-group variation is maximized. These sensor arrays can be dynamically defined based on a cyclic sampling of the sensor readings, performing the cluster analysis, and grouping sensors into sensor arrays. In the case of steady-state criteria, these sensor arrays are fairly stable in terms of their boundaries, but in dynamic criteria, the boundaries of the sensor arrays may vary in time. The same process can also be used for the actuators if the actuators are at the same scale as the sensors.

These sensor arrays or 'zones' can communicate with each other using radio frequency waves, exchange information, and create ad-hoc networks of measurements. Optimization of environmental performance criteria in the spatial enclosures is modeled as an optimization based on this network of measurements.

2.3 EFFECTORS

An effector is a complex entity that produces an effect on another entity or on the environment. An effector is a sensor-actuator pairing. It combines, at a minimum, one sensor with one actuator. Other configurations that are possible are the coupling of one sensor with many actuators, many sensors with one actuator, and many sensors with many actuators. The actuator performs its action based on the readings from one or more sensors. This actuation process can utilize computation or other forms of processing of the sensor data. The optimization, both spatial and temporal, of readings from multiple sensors requires sophisticated techniques. The sensor can simply be a measuring instrument that record measurements of different kinds. Sensors are available that can measure temperature, humidity, mass airflow, position, etc. An actuator can range in scale from one that does molecular manipulation to micro electro-mechanical systems to large scale motor-based kinetic systems.

3. Environmental Performance

3.1 ENVIRONMENTAL PERFORMANCE CRITERIA

The environmental performance criteria of interest to designers of spatial enclosures for various purposes are illumination levels, sound frequencies, sound amplitude, sound intensities, temperature and humidity. These performance criteria correspond to the human senses of sight, hearing and touch respectively. Being able to optimize these criteria in complex spatial enclosures will enhance human activities in those enclosures in transparent and subtle ways.

For example, patrons sitting in a library carrel and reading books will not notice that these ultra-small sensors have recorded the illumination levels on their desktops and signaled the lights overhead to increase their brightness. Concert goers in a concert hall will not notice the detection of an echo condition and its cancellation by a network of actuators controlled by sensors on the surfaces of the spatial enclosure of the auditorium. A homeless man will not realize that the park bench he is approaching is being warmed up for him to curl on as he approaches the bench.

3.2 STEADY-STATE AND DYNAMIC ENVIRONMENTAL PERFORMANCE CRITERIA

A steady-state environmental performance criterion is one that does not vary significantly over time, unless a change is made to one of its causal agents. A dynamic environmental performance criterion is one which varies in time. The time cycles can range from mere milliseconds or seconds in the case of sound, to a year in the case of temperature. Illumination levels in a spatial enclosure are steady-state phenomena. Unless the light sources are changed, the illumination levels stabilize in a very short duration after the light sources are turned on. The only exception to this is daylight which causes illumination levels to vary over a time cycle. Temperature variation is similarly a steady-state phenomenon, especially in interior spatial enclosures. Temperature variation can be a dynamic phenomenon if the spatial enclosure has a membrane exposed to the exterior, which has a dynamic temperature range over a time cycle.

4. Optimization

4.1 OPTIMIZATION OF ENVIRONMENTAL PERFORMANCE CRITERIA

Optimization can be simply defined as the maximizing or minimizing a particular quantity, in this case, the measurements of various environmental performance criteria recorded by the various sensors.

The spatial distribution of sensors makes it necessary to consider the spatial effect of the optimization. A local optimization has global effects, and global optimization has local effects. This spatial effect can be resolved using relatively simple techniques in the case of steady-state criteria. The optimization becomes very complex when the criteria are dynamic and change over different time cycles. For example, sound changes over time cycles measured in millisecond intervals and temperature varies over time cycles measured in hours.

Optimization of environmental performance criteria is complicated by the fact that human preferences are often based on aggregate measurements of the environmental performance criteria rather than instantaneous measurements. For example, human preferences for acoustical conditions are based on the ratio of sound energy summations over time intervals, rather than the instantaneous sound energy variation in time, as would be indicated by an energy response graph.

4.2 RESOLVING THE SPATIAL EFFECTS OF SOURCES AND EFFECTORS

The optimization of the environmental performance criteria depends to a large extent on the resolution of the spatial effects caused by the sources (light fixtures, sound sources, heat sources, etc.) and the spatial effects caused by the surfaces that make up the enclosure. The measurement at a particular sensor at a particular time can be modeled as a vector of the spatial effects of the various sources and surfaces that are part of the spatial enclosure. All the vectors are convolved in time to produce the dynamic variation of criteria at a particular sensor location.

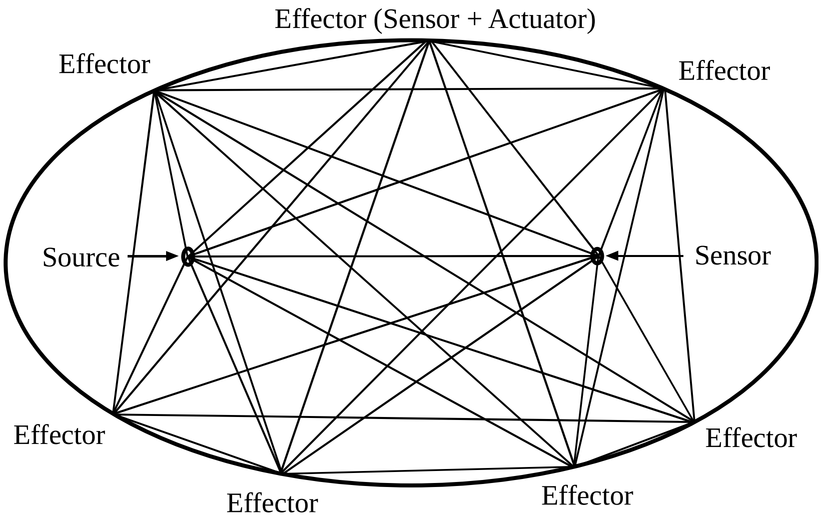


Figure 1. Diagram of a system showing a single source, a set of effectors (sensor-actuator pairs) and a performance sensor. This system is called an optimaton.

An optimization network that links a single source, effectors and a single performance sensor is called an *optimaton*. An *optimaton* behaves like a neural network in that the source acts as an input, the network of effectors optimizes the effect of the source or "transforms" the source and the performance sensor receives the output of the transformation. Multiple *optimatons* can be linked to form meta-networks by networking the sources or the performance sensors.

The mathematical model for the spatial effect of multiple sources in a spatial enclosure made up of a finite number of surfaces can be modeled as follows:

1) Let us say there are m sources S_1……S_m.
2) Let us also say that there are n surfaces E_1……E_n such that each functions as an effector (sensor-actuator pair), and
3) Let us measure performance characteristics at one performance sensor R.

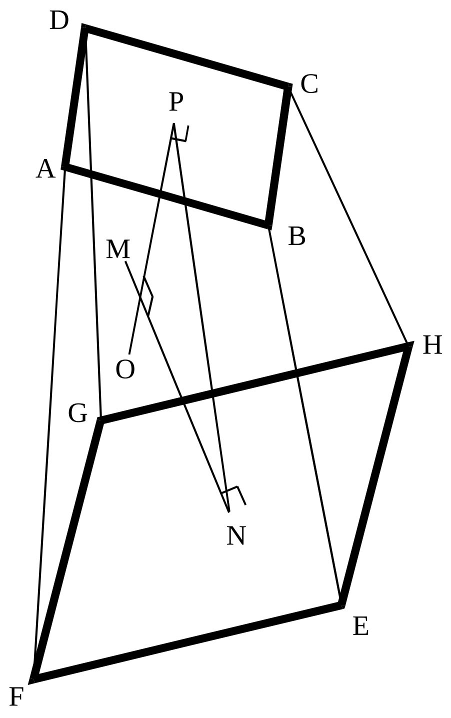


Figure 2. Diagram showing the form factor between two components

The spatial relationship between any two components of an optimization network is called its form factor. Form factors establish the spatial effect between a source and an effector, or between an effector and another effector, between a source and a performance sensor, and an effector and a performance sensor. Figure 2 shows the form factor relationships between any two components in the model.

In Figure 2, ABCD and EFGH are two components. MN and OP are the normals to the two components at their centroids. NP is the distance between the two components. The angle between the two components is the angle between the normals. To derive the form factor between the two components, the following relations are taken into account:

1. The radiation between the components is directly proportional to the ratio of their areas. This can also be modeled as being proportional to the solid angle subtended by the two components.
2. The radiation between the two components is inversely proportional to the square of the distance between the components.
3. The radiation between the two components is proportional to the cosine of the angle between the two components.

The form factor for the two components will then be: (A_{abcd}/A_{efgh}) * $(1/D_{np}{}^{2)}*(\cos\theta)$

For a situation where there is a single source, n effectors and one performance sensor, there are the following form factors:

1) Source to performance sensor (SR)
2) Source to all effectors (SE_1……SE_n)
3) Each effector to all other effectors (E_1E_2…..E_1E_n)

4) Each effector to the performance sensor ($E_1R\ldots\ldots E_nR$)

These form factors can be written as a sum of vectors: $V(SR) + V(\sum_{1 \text{ to } n} SE_n) + V(\sum_{1 \text{ to } n} E_n\sum_{1 \text{ to } n-1} (E_{n+1})) + V(\sum_{1 \text{ to } n} E_nR)$. These form factors add up to a total of $(1) + (n) + n(n-1) + (n) = n^2 + n + 1$ form factors.

The radiation sequence from the sources to the effectors to the performance sensor can be modeled. In this model,
1) Energy is radiated from the source to the performance sensor
2) Energy is radiated from the source to all the effectors in the spatial enclosure
3) Energy is radiated from all the effectors to the performance sensor
4) Energy is radiated from each effector to all other effectors
5) Energy is radiated from all effectors to the performance sensor
6) Steps 4 and 5 are repeated till the energy is dissipated

Steps 1-5 are defined as constituting the primary propagation. This propagation engages all form factors once. The energy is therefore multiplied by $(n^2 + n + 1)$ form factors for this primary propagation. After this propagation, the secondary propagation is a repetition of the radiation from each effector to all other effectors, and from all effectors to the performance sensor (steps 4 and 5). This represents $n(n - 1) + n = n^2$ form factor calculations per cycle. This propagation cycle is repeated at the required frequency f. Therefore the total number of form factor computations for energy propagation is $((n^2 + n + 1)+(n^2)f)$ if there are no form factor updates required by changes in effectors. For m sources there are $m((n^2 + n + 1)+(n^2)f)$ form factor computations for the propagation. This is a polynomial and can be computed in polynomial time.

This process is complicated by the fact that any change at an effector will affect the subsequent propagation. If an effector changes in a single propagation cycle, then the following form factors change:
1) That effector to all other effectors
2) That effector to the performance sensor
3) That effector to the source

This represents an update of $(n-1) + (1) + (1) = (n + 1)$ form factors for the next propagation cycle. If all effectors change in one cycle, then the following form factors will change:
1) All effectors to each other
2) All effectors to performance sensor
3) Source to all effectors

This represents an update of $n(n - 1) + (n) + (n) = (n^2 + n)$ or $n(n + 1)$form factors for the next propagation cycle. The algorithm for energy propagation will be as follows:
1) Start with energy Q
2) Multiply Q by vectors of a total of $(n^2 + n + 1)$ form factors

3) Record energy and time of arrival at the performance sensor, after multiplication by vectors that end in performance sensor
4) Update n(n + 1) form factors, where n is the number of effectors that have changed
5) Multiply Q by vectors of a total of (n^2) form factors
6) Update/record energy and time of arrival at the performance sensor, after multiplication by vectors that end in performance sensor
7) Repeat updates of form factors
8) Repeat multiplication by vectors of a total of (n^2) form factors
9) Update/record energy and time of arrival at the performance sensor, after multiplication by vectors that end in performance sensor
10) Plot energy response graph
11) Sum energy for time intervals of interest
12) Compute parameters based on relations between energy at different time intervals

4.3 OPTIMIZATION FUNCTIONS

Optimization in this system will be the optimization of form factors based on changes made by effectors. Each effector can be changed based on an objective function, the form factors can be updated, new vectors of form factors can be computed, and the propagation cycle can be repeated.

The objective function of a form factor between two components can be based on three terms, the area, the distance between components and the angle between the components. It will take the form: $f(a, d, \theta)$

Each of the variables in the objective function has a range of values. The ranges are as follows:
1) The area (a) can vary from 0 to ∞
2) The distance (d) can vary from 0 to ∞, but a practical upper limit is $\sqrt{(\text{initial energy} / \text{perception threshold})}$
3) The angle between components (θ) can vary from 0 to π

The objective function for a spatial enclosure is a function of a maximum of ($n^2 + n + 1$) form factors. It will take on the form: $f\ (ff_1 \ldots\ldots ff_{n^2+n+1})$. An objective function used in the optimization of environmental performance criteria can have up to ($n^2 + n + 1$) terms. This is a large number of terms in an objective function. However, these terms can be grouped into four sets that behave in a concerted way. These four sets are:
1) Term associated with direct transfer of energy from source to receiver.
2) Terms associated with transfer of energy from a source to all other effectors.
3) Terms associated with transfer of energy from an effector to all other effectors.
4) Terms associated with the transfer of energy from all effectors to the receiver.

4.4 COMMON FRAMEWORK

The complex process of optimizing the various environmental performance criteria can be resolved by adopting a common framework for the spatial propagation of the various types of energy. One such framework is the general model of radiation (Mahalingam, 2000). Radiation is simply considered as the transfer of energy between spatially separated surfaces. For the modeling of temperature variation in a spatial enclosure, this technique can be applied directly. The radiosity based modeling of light propagation in spatial enclosures provides a theoretical framework for modeling illumination levels in the spatial enclosure based on radiation. The radiation-based modeling of sound propagation provides a theoretical framework for the modeling of sound intensity levels in the spatial enclosure (Mahalingam, 1999).

The triple integral form of the radiation equation, that measures radiation from a source to a surface, integrates intensities based the variation of the surface orientation (the cosine or Lambertian component), the area of the surface, the solid angle subtended by the surface at the source, and time. This may provide the common framework that may simplify the optimization process.

This relationship is given by:

$Q_e = \iiint L_e \cos\theta \, dA \, d\Omega \, dt$ (Woan, 2000)

Q_e = energy at a surface
L_e = rate of transfer of energy per unit area per steradian
$\cos\theta$ = angle between surface where energy is being measured and the source
A = area of surface where energy is being measured
Ω = solid angle in steradians subtended by surface where energy is being measured
t = time in seconds

5. Conclusion

This paper has introduced a model for the optimization of environmental performance criteria in spatial enclosures using a system of effectors (sensor-actuator pairs). This model uses a common radiation-based propagation framework for different kinds of energy, namely thermal, luminous and acoustical energy. The effectors that regulate the various environmental performance criteria are shown to form optimization networks or *optimatons*. These *optimatons* behave in a manner that is similar to other well-known computational networks such as neural networks.

References

Farlow, S.J.: 1993, *Partial Differential Equations for Scientists and Engineers*, Dover Publications Inc., New York.

Mahalingam, G.: 2000, Enhanced Boundary Representation: A *Lingua Franca* For Computer-based Building Performance Simulation?, in the *ACADIA Quarterly*, Association for Computer-aided Design In Architecture.

Mahalingam, G.: 1999, A New Algorithm for the Simulation of Sound Propagation in Spatial Enclosures, in the Proceedings of the Building Simulation ’99 Conference, Kyoto, Japan, September.

Woan, G.: 2000, *The Cambridge Handbook of Physics Formulas*, Cambridge University Press, Cambridge, United Kingdom.

Enhanced Boundary Representation: A *lingua franca* for computer-based building performance simulation?

Ganapathy Mahalingam, Ph.D.

Abstract

With the realistic visual representation of buildings on the computer having reached maturity, the emphasis has now shifted to the performance simulation of buildings on the computer. The challenge of performance simulation in computer-based models of buildings lies in the integration of various simulation techniques that require different kinds of building representations. Traditional simulation techniques for luminous, acoustic and thermal environments require different building representations. The paper proposes that an enhanced boundary representation is a viable, common building representation format for performance simulation of illumination levels, acoustical parameters and thermal comfort, thereby providing a building representation format for multi-domain performance simulation on the computer. Simulation techniques that have been developed for radiosity-based modeling of illumination in buildings, radiation-based modeling of sound propagation in spatial enclosures, and the modeling of thermal comfort based on mean radiant temperatures, point to a convergence of techniques. These techniques can all work based on an enhanced boundary or surface representation of buildings. The paper suggests that an enhanced boundary representation format, and integrated performance simulation techniques based on radiation, can together serve as a core model for developers of computer-aided design analysis systems.

1 Introduction

In the four decades of its rapid development, the field of computer-aided architectural design (CAAD) has successfully focused on the visual synthesis and representation of architectural designs. The visualization of built things has been achieved at unprecedented levels of realism in both static and animated forms. As architects, we cannot be satisfied, dealing with built things on a purely visual level. Architectural creations engage us and affect us in many other ways that have to be understood and computationally modeled to augment our design capabilities. The key benefit of the computational modeling of architecture, which is yet to be fully realized, is that it makes evaluation of designs possible before they are built.

Extensive research has been conducted in the computational modeling of the natural environment. The visual modeling of the natural and the built environment has been pursued rigorously with excellent results. However, the computational modeling of the built environment in its multiple aspects is still in its infancy. The research challenge of the next decade is the computational modeling of the built environment at all scales, from individual buildings to large cities, that focuses on aspects other than visualization. This computational modeling of architectural entities and architectural design processes should emphasize performance simulation over the visualization of the product. This will enable architects to computationally evaluate their designs based on various aspects of performance. The performance of designs for buildings should now become the center of attention for researchers engaged in the computational modeling of the built environment. Though this task presents many challenges, the development of computer-aided design analysis systems (CADAS) is the logical next step in the evolution of computer-aided architectural design. The various issues in the multi-domain simulation of building performance in computer-based models of buildings have to be systematically addressed if progress is to be made.

2 Performance Simulation

The challenge of performance simulation in computer-based models of buildings lies in the integration of various simulation techniques that require different kinds of building representations. This has been called the "integration problem." Traditional simulation techniques for luminous, acoustic and thermal performance require different building representations or data models that contain both geometric and attribute data. Initial data models for the representation of buildings on computers were directed towards visual representation. The boundary representation of building forms enabled surface characteristics of buildings to be successfully modeled and rendered. The use of color and texture for the representation of materials by applying texture, transparency and bump maps onto surfaces enabled the realistic visual representation of buildings. When the material properties of forms, such as mass, needed to be addressed in structural analysis, constructive solid geometry (CSG) was used as the representational model. Mass properties could be easily calculated from the CSG representational model. The CSG representational model, however, did not serve the needs of performance simulation in other domains well, and was useful only in a narrow domain. This has restricted the use of CSG mainly in computer-aided design systems used by mechanical and structural engineers. This may not be the case with boundary representation.

2.1 Domains of Performance Simulation

While there is a wide range of performance measures used in the design of buildings, the performance of buildings needs to be evaluated in the domains of the five senses of human beings to test the quality of a building's habitability. The five human senses are sight, hearing, touch, smell and taste. Of these, sight, hearing and touch (felt as kinesthetics or tectonics, though these may be argued to be muscular senses) are the critical senses that architects have traditionally addressed while designing buildings. Architecture has long reigned primarily as a visual art, with the visual manipulation of virtual material being the predominant mode for designing buildings. Architects using traditional techniques have evaluated the performance of buildings only in the domains of the three critical senses mentioned earlier. To evaluate computer-based architectural designs in the domains of the human senses, computer-based models of designs for buildings should have a representational structure that allows the simulation of the buildings in various sensory modes, or alternatively, provide indices for the sensory modes.

2.2 Integration of Performance Domains

Since human sensory domains primarily involve the processing of radiation information (with the exception of taste and smell), simulation techniques based on radiation models become a natural choice when trying to integrate performance simulation techniques. Simulation techniques that have been developed for the radiosity-based modeling of illumination in buildings, radiation-based modeling of sound propagation in spatial enclosures, and the modeling of thermal comfort based on mean radiant temperatures, point to a convergence of simulation techniques. These techniques can all be used with an enhanced boundary or surface representation of buildings. The enhanced boundary representation format, and integrated performance simulation techniques based on radiation, can now serve as the core model for developers of computer-aided design analysis systems.

3 Integrated Performance Simulation

A significant research effort in integrated performance simulation in computer-based models of buildings is the SEMPER system being developed at Carnegie Mellon University. The SEMPER system uses a variety of building representations for various performance simulations. In the SEMPER system, energy flows are simulated using cell nodes for walls and spaces. Airflow is simulated using multiple zones of differing pressures. HVAC systems are modeled as components linked in a distribution network. Lighting is simulated using a radiosity-based method. Sound propagation is modeled as the generation and emission of "virtual phonons" generating an excitance pattern for the room enclosures (Mahdavi 1999). In the SEMPER system, the building representations needed for the various performance simulations are derived from a shared object model and a topology kernel. To solve the "integration problem," the developers of SEMPER have taken the approach of creating a lowest common denominator building representation for the shared object model and the topology kernel, from which other building representations are derived. These complex derivations create domain models and domain kernels for performance simulation in the various domains.

Enhanced boundary representation is an alternate, common building representation format that can enable performance simulation of multiple aspects of a building's performance. Performance simulation of illumination levels, acoustics and thermal comfort can be achieved using an enhanced boundary representation format without a complex process to derive different building representations for different performance simulations.

The enhanced boundary representation method for representing architectural forms promises to be a strong candidate for a common representational format for the multi-domain evaluation of building performance for many reasons. The radiosity-based simulation of light propagation in spaces enables visual simulation, and the evaluation of computer-based building models for illumination levels (Greenberg 1995). The success of modeling and rendering software such as form.Z^{TM}, which uses boundary representation predominantly, is a testimony to this fact. The virtual light meter provided in the Lightscape^{TM} software, is an example of how illumination levels can be measured in computer models of buildings created with a boundary representation format. The radiation-based model for the propagation of sound in computer-based building models enables acoustical parameters related to the perception of speech and music to be computed and evaluated (Mahalingam 1999). The radiation-based indicator, mean radiant temperature (MRT) allows thermal comfort to be evaluated in computer-based models of buildings. Though the mean radiant temperature is not a complete measure of thermal comfort (researchers suggest that air temperature be averaged with it to derive what is called the operative temperature), it is a good general indicator (Stein and Reynolds 2000). In these three examples, enhanced boundary representation is shown to be sufficient for the performance simulations in three human sensory domains, sight, hearing and touch (touch can be represented by thermal comfort, since the skin is the sensory organ for the perception of thermal comfort). The various performance indicators and the simulation techniques used for these three performance simulations are also related because they are all based on radiation models.

3.1 Enhanced Boundary Representation Format

To serve as a common representational format for various types of performance simulations, enhanced boundary representation of buildings should have the following specifications:

- All the forms that make up the building should be constructed out of well-formed surfaces.
- It should be possible to compute the center of each surface.
- It should be possible to compute the area of each surface.
- It should be possible to compute the surface normal at the center and vertices of each surface.
- It should be possible to compute the form-factor for a pair of surfaces (form-factor is a relationship between two surfaces, including orientation, to simulate a radiation exchange between them)
- It should be possible to define the material properties related to the various performance domains for each surface.

These specifications will enable the various performance simulation techniques to be used effectively based on what is essentially a common representational format. Because the surface components are the same for the performance simulations, and form-factors between pairs of surface components are the same, it is also possible to represent the surfaces computationally using object-oriented modeling techniques and modeling the radiation interaction between the surfaces as computations in a network.

3.2 Limitations

The use of an enhanced boundary representation format for computer-based building models has some limitations. All surfaces should preferably be planar to reduce computational complexity. Curved surfaces must be faceted into planar surfaces. This needs to be done so that the surface normals for the surfaces (especially at the center of the surfaces) can be effectively computed. The occlusion of surfaces by other surfaces also needs to be computed efficiently. The size of the surfaces should also be at the appropriate level of resolution for efficiency in computations. Having many tiny surfaces in the building model will greatly increase the computational resources required for the simulations. Because of the nature of the model, boundary representation and radiation models will be more successful in the performance simulation of global, diffuse effects rather than directional and specular effects.

4 Conclusions

The performance simulation of computer-based designs of buildings in various performance domains has been difficult because of what has been called the 'integration problem." Researchers trying to use performance simulation techniques with building representations that are not compatible with the simulation techniques have augmented this problem. In this paper, an approach has been suggested that takes a building representation format that is currently being used in computer-aided architectural design systems, and using an enhanced version of that representation format as a basis for an integrated set of performance simulation techniques based on a common radiation model.

References

Mahalingam, G. (1999). A New Algorithm for the Simulation of Sound Propagation in Spatial Enclosures. Building Simulation '99 Conference Proceedings. Kyoto, Japan.

Greenberg, D. P. (1995). Computers and Architecture. In *Scientific American*, vol. 6, no. 1, eds. J. Rennie, M. Press and J. T. Rogers, 120-125. New York: Scientific American Inc.

Mahdavi, A. (1999). A comprehensive computational environment for performance based reasoning in building design and evaluation. In *Automation in Construction*, vol. 8, no. 4, eds. Y. E. Kalay and R. Becker, 427 - 435. New York: Elsevier.

Stein, B. and J. S. Reynolds. (2000). *Mechanical and Electrical Equipment for Buildings,* 9th ed., 42-43. New York, New York: John Wiley & Sons, Inc.

POCHÉ
__P__olyhedral __O__bjects __C__ontrolled by __H__eteromorphic __E__ffectors

Ganapathy Mahalingam
North Dakota State University

Key words: Effectors, abstract machines, design as interface

Abstract: This paper takes the architectural concept of poché and uses it to explore new possibilities in transforming polyhedra with effectors. In many computer-aided design systems, architectural entities are represented as well-formed polyhedra. Parameters and functions can be used to modify the forms of these polyhedra. For example, a cuboid can be transformed by changing its length, breadth and height, which are its parameters. In a more complex example, a polyhedron can be transformed by a set of user-defined functions, which control its vertices, edges and faces. These parameters and functions can further be embodied as effectors that control and transform the polyhedra in extremely complex ways. An effector is an entity, which has a transforming effect on another entity or system. An effector is more complex than a parameter or function. An effector can be a modelled as a virtual computer. Effectors can take on many roles that range from geometric transformation agents and constraints to performance criteria. The concept of the poché, made famous by Venturi is familiar to architects. The poché is a device to mediate the differences between an interior and an exterior condition or between two interior conditions. In a poché, the role of the effector changes from being an agent that acts on a polyhedron from the outside, to an agent that acts as a mediator between an interior polyhedron and an exterior polyhedron, which represent interior and exterior environments respectively. This bi-directionality in the role of the effector allows a wide range of architectural responses to be modelled. The effector then becomes an interface in the true sense of the word. This concept will work best in a three-dimensional or four-dimensional representational world but can be used effectively in a two-dimensional representational world as well. The application of this concept in design systems is explored with examples drawn from the work of the author, and practitioners who are using the concept of effectors in their work. A brief discussion of how this technique can evolve in the future is presented.

1. ARCHITECTURAL ENTITIES AS POLYHEDRAL OBJECTS

Architectural entities can be classified as physical or conceptual. Physical architectural entities are made of building materials and have geometric form. Physical architectural entities comprise building materials, components and assemblies. Conceptual architectural entities may have geometric form but are not made up of any material. Conceptual architectural entities comprise entities such as ordering systems and circulation systems. Both physical and conceptual architectural entities have spatial location. Conceptual architectural entities can influence the geometric form and spatial location of physical architectural entities. Conceptual architectural entities, in turn, can be defined by physical architectural entities. Architectural design can be considered as the definition and integration of physical and conceptual entities and fixing their location in space.

In computer-aided design systems, physical architectural entities are represented as well-formed polyhedra. Polyhedra, as their name implies, are volumes defined by a closed boundary of faces. The representation of architectural entities as well-formed polyhedra is called boundary representation. By their definition polyhedra are finite and can be fabricated with material. In their computer-based representation, polyhedra are defined as hierarchical collections of vertices, edges and faces. Of these, the actual variables are the values for the tuples that define each of the vertices of a polyhedron. These variables, in turn, determine the variations in the dimensions and spatial locations of the edges and faces of the polyhedron. Controlling the variables allows a designer to control the various forms that the polyhedra can take. The manipulation of form, which is one of the principal activities of the designer, can be enhanced by creating armatures for the manipulation and transformation of architectural entities represented as polyhedra. The concept of effectors provides one such armature.

2. EFFECTORS

An effector is an entity, which has a transforming effect on another entity or system. An effector is more complex than a constraint, parameter or function. An effector can be a computational entity in its own right. It can accept different kinds of input, perform computations and cause an effect in another entity as part of its output. An effector can be modelled as a virtual computer that can embody both state and behavior (Mahalingam, 1998). Effectors can take on many roles that range from geometric transformation agents and constraints to performance criteria. Multiple effectors can be integrated as a network of effectors that act in a concerted manner to function as a meta-effector. A hierarchical system of effectors and meta-effectors can also be developed that may or may not be concerted in their transforming effects. The overall effect of a hierarchy of effectors, that do not act in a concerted manner, needs to be controlled by a meta-effector.

Effectors change other entities based on computations. Effectors are different from constraints or parameters. Constraints specify an acceptable range of values for a variable, or relationships between variables. When a constraint specifies a maximal or minimal value for a variable, then it is considered a "limit." If the constraint specifies a range of values for a variable that straddles a certain value, then it is considered a "tolerance." The acceptable values for a variable specified by a constraint can be simply declared, or computed based on a function of that variable or other associated variables.

A parameter specifies values for a variable indirectly. The actual values of the variables are computed using the parameter. For example, the parameters for a cuboid are length, breadth and height. Changing these parameters changes the values for the vertices of the cuboid. The values of the vertices are computed using the parameters. A parameter needs an origin in order to compute its effect unambiguously. For example, changing the length of a cuboid can change only four or all eight of its vertices, depending on whether a vertex or the centroid of the cuboid is used as the origin.

A function can also be used to determine the value of a variable, which is usually called the dependent variable. A function can be any mathematical relation and may not require an origin to compute its effect. A function is defined in terms of one or more independent variables. The function may sometimes be recursive and use the variable it is determining in its computation. <u>Effectors subsume constraints, parameters and functions when they are modelled as virtual computers.</u>

When an effector is modeled as a virtual computer, a network of effectors becomes a network of computers. Polyhedra controlled by bi-directional effectors become a multi-layered network. Mathematical models used to model neural networks, parallel systems and 3D graphs are all viable tools to model networks of effectors. In its most abstract and general form, an effector is a relation between two virtual computers. Each virtual computer can change the state and behaviour of the other virtual computer. This relationship can be realised through one or several computational processes.

2.1 Homomorphic and heteromorphic effectors

If the component of a polyhedron being controlled by an effector is the same kind as the effector, then the effector is homomorphic. If the effector is not the same kind as the component it is transforming, then the effector is heteromorphic. To determine if an effector is the same kind as the component it is transforming, it should correspond in form or type. Geometric transformation effectors transforming the geometry of a polyhedron are homomorphic effectors. This is type correspondence. Since effectors are not geometric entities, the issue of form correspondence arises only if there are connected effectors. For example, two effectors joined to form a straight line that transform the edges of a polyhedron are considered homomorphic effectors. Also, a polyhedral network of effectors transforming a polyhedron is considered a homomorphic effector.

3. POCHÉ

The concept of the poché, made famous by the architect Robert Venturi, is familiar to architects. The poché is a device to mediate the differences between an interior and an exterior condition or between two interior conditions. It is usually used to resolve two conflicting requirements or conditions.

In his influential book, *Complexity and Contradiction in Architecture*, Venturi explains the wide ranging implication of the concept of poché by quoting Gyorgy Kepes, "Every phenomenon - a physical object, an organic form, a feeling, a thought, our group life - owes its shape and character to the duel between opposing tendencies; a physical configuration is a product of the duel between native constitution and outside environment." (Venturi, 1966). Venturi also states that designing from the outside in, as well as the inside out, creates necessary tensions, which help make architecture. He goes on to make a significant and influential statement, "Since the inside is different from the outside, the wall - the point of change - becomes an architectural event." The concept of bi-directional effectors takes the condition of poché described by Venturi and provides a computational framework to implement the design processes that he describes. For example, the "native constitution" of an entity can be governed by uni-directional effectors, and "duel" with the "outside environment" can be mediated by bi-directional effectors.

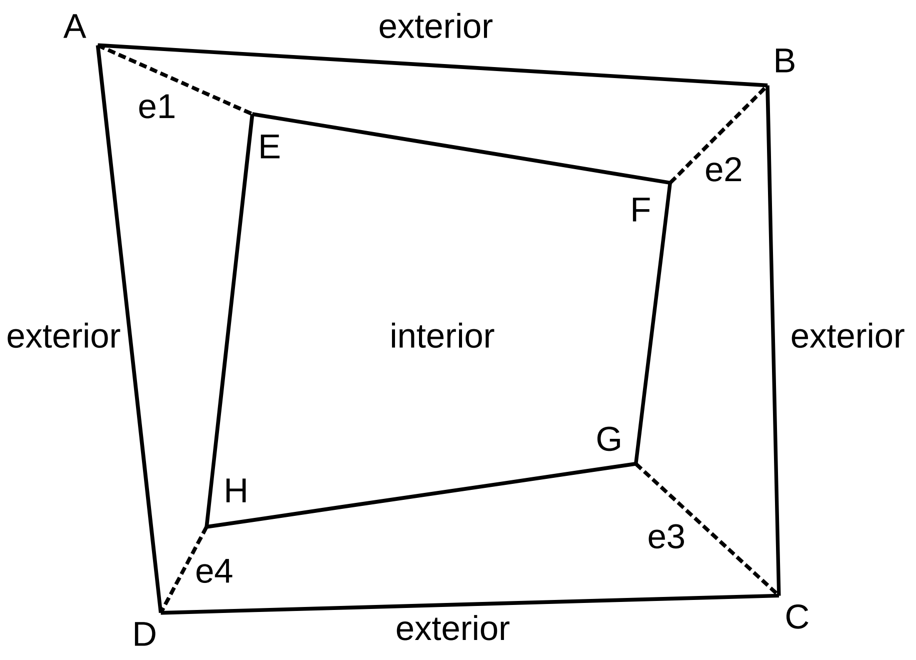


Figure 1. A simple poché condition with four bi-directional effectors in a two-dimensional representational world

In Figure 1, an example of a poché condition in a two-dimensional representational world is shown. This figure is not a literal diagram but represents an abstract machine in the Deleuzian sense (Deleuze and Guattari, 1987). ABCD is an exterior polygon and EFGH is an interior polygon. The states of A, B, C and D are determined by exterior contextual criteria. The states of E, F, G and H are determined by interior performance criteria. The four bi-directional effectors are e1 (AE), e2 (BF), e3 (CG) and e4 (DH). The bi-directional effectors e1, e2, e3 and e4 can be parameters, constraints, functions or virtual computers (Mahalingam, 1998). The role of the bi-directional effectors is to mediate and determine the states of the interior/exterior variable pairs that they link. The four bi-directional effectors can be linked to form a polygon of effectors thereby defining a meta-effector that is homomorphic. A meta-effector is a network of effectors. The role of the meta-effector in this case is to co-ordinate the effects of the four bi-directional effectors. Meta-effectors can be constraining agents. If, for example, the above condition represents a single-room structure, then the meta-effector can ensure that a minimum wall thickness is maintained.

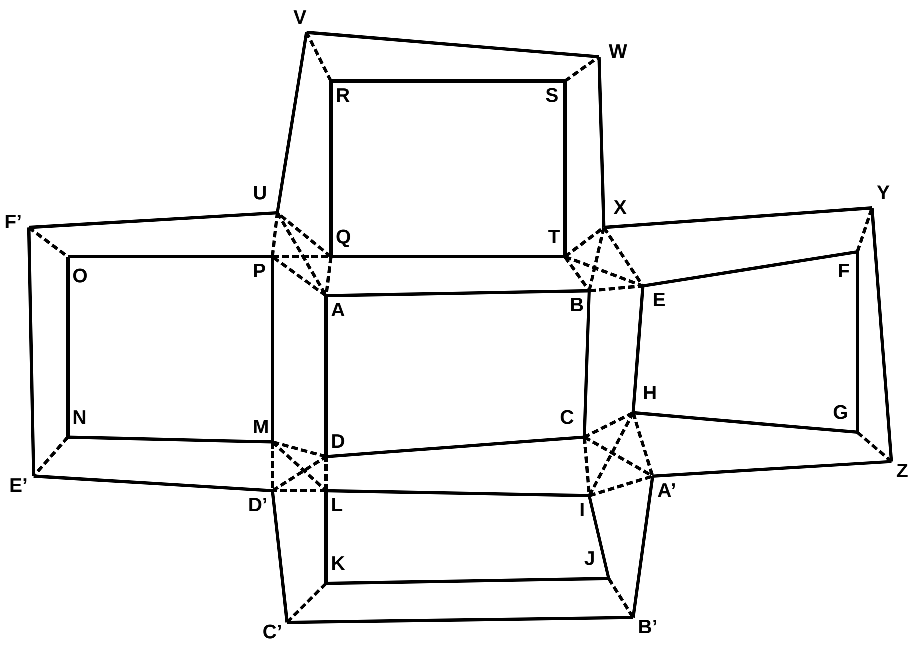


Figure 2. A more complex poché condition with bi-directional effectors in a two-dimensional representational world

In Figure 2, a more complex, multicellular poché condition is shown in a two-dimensional representational world. This figure is also not a literal diagram but represents an abstract machine in the Deleuzian sense (Deleuze and Guattari, 1987). There are five interior polygons and an exterior polygon. There are eight peripheral bi-directional effectors and there are four clusters of bi-directional effectors in the interior, each cluster forming a nexus of effectors affecting four variables. In each nexus the state of each one of the four variables needs to be resolved based on the simultaneous effects of the individual effectors. Each nexus can be modeled as a meta-effector. The four clusters that each forms a nexus can be networked as a polygon to create another meta-effector that is one step higher in a hierarchy of effectors. One role of the "nexus" meta-efffectors is conflict resolution at each nexus, to resolve conditions such as spatial overlap. A "nexus" meta-effector can also be used to maintain a minimum separation distance between the variables. Alternatively, a "nexus" meta-effector can collapse into a simple bi-directional effector, suggesting a merging of some of the cells in the multicellular configuration.

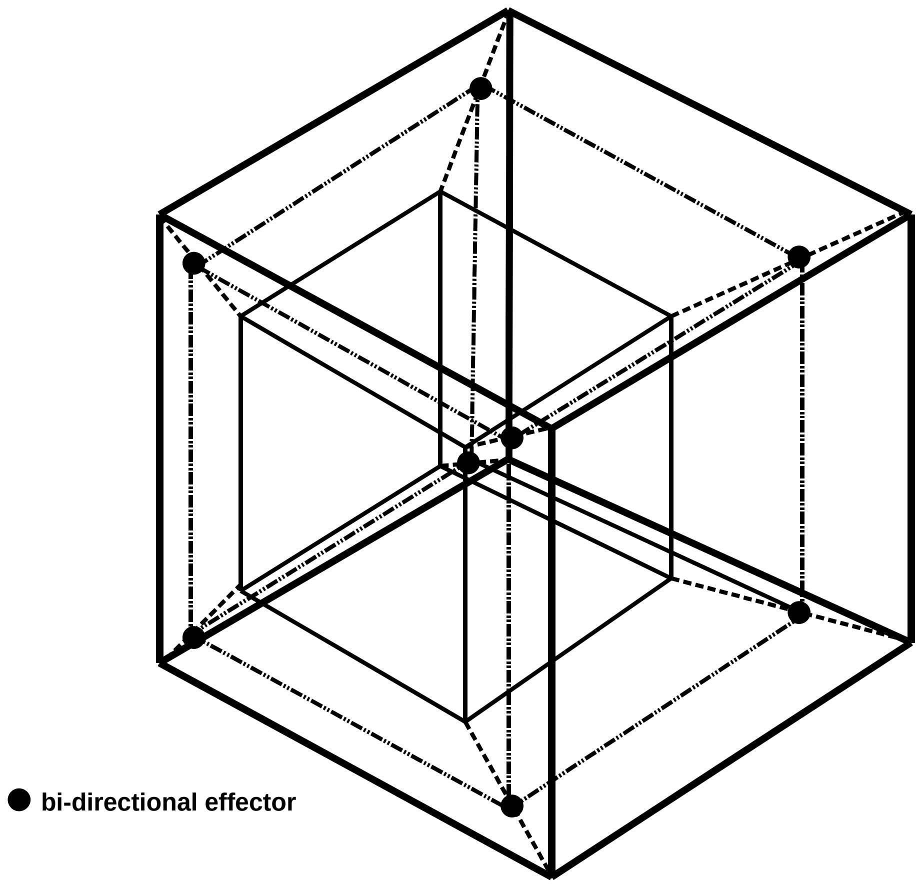


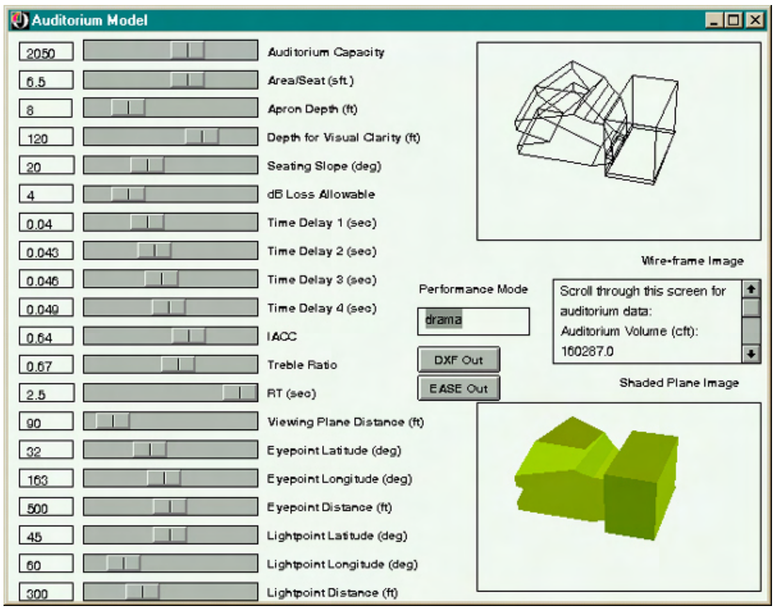
Figure 3. A simple poché condition with bi-directional effectors in a three-dimensional representational world

In Figure 3, a model consisting of three nested polyhedra is shown. If the polyhedron in the middle (shown in the dash-dot line) is a network of effectors, and the polyhedra on the inside and outside represent interior and exterior environments, then the whole system represents the condition of poche' in a three-dimensional representational world. The role of the effector changes from being an agent that acts on a polyhedron from the outside, to an agent that acts as a mediator between an interior polyhedron and an exterior polyhedron, which represent interior and exterior conditions respectively. This bi-directionality in the role of the effector allows a wide range of architectural responses to be modelled, especially simultaneous responses to interior performance criteria and external contextual conditions.

The bi-directional effector can be a mediating channel through which conflicting conditions between interior and exterior environments are resolved (mediated?). The bi-directional effector then becomes an interface in the true sense of the word.

3.1 Application of effectors in architectural design

The application of effectors in digital design processes for architecture holds a lot of potential. Since architectural entities are usually represented as polyhedra, the transformation of the polyhedra by effectors becomes a central part of design processes that shape forms and space.



2050	Auditorium Capacity
6.5	Area/Seat (sft)
8	Apron Depth (ft)
120	Depth for Visual Clarity (ft)
20	Seating Slope (deg)
4	dB Loss Allowable
0.04	Time Delay 1 (sec)
0.043	Time Delay 2 (sec)
0.046	Time Delay 3 (sec)
0.049	Time Delay 4 (sec)
0.64	IACC
0.67	Treble Ratio
2.5	RT (sec)
90	Viewing Plane Distance (ft)
32	Eyepoint Latitude (deg)
163	Eyepoint Longitude (deg)
500	Eyepoint Distance (ft)
45	Lightpoint Latitude (deg)
60	Lightpoint Longitude (deg)
300	Lightpoint Distance (ft)

Performance Mode: drama

Scroll through this screen for auditorium data:
Auditorium Volume (cft):
160287.0

Figure 4. View of auditorium design system developed by the author

The author has developed an auditorium design system (see Fig. 4) where the polyhedral form of a proscenium-type auditorium is generated based on multiple functions of acoustical, programmatic and functional parameters (Mahalingam, 1996). The functions that locate the vertices of the polyhedra that make up the auditorium can be likened to uni-directional effectors. These effectors take the role of functions. The model in the auditorium design system that was developed was inwardly oriented. The design system can now be extended with bi-directional effectors to control an exterior polyhedral form that responds to external site conditions.

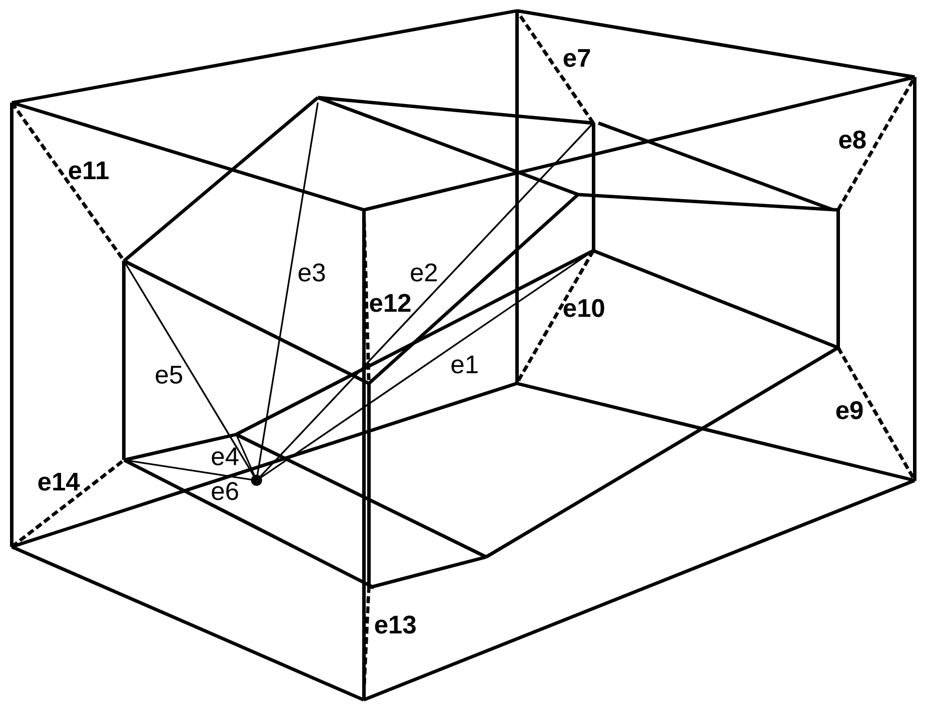


Figure 5. A conceptual diagram of the integration of the concept of effectors in the auditorium design system

In Figure 5, the spatial form of an auditorium seating enclosure is shown. The vertices of the polyhedron that make up the seating enclosure is determined by a set of twelve uni-directional effectors (e1…e6 represent one half of a symmetrical set of twelve effectors). The exterior polyhedron of the auditorium is governed by eight bi-directional effectors (e7…e14). Based on the mediating action of the bi-directional effectors, the exterior polyhedron can respond to site and other contextual conditions, while at the same time responding to the interior polyhedron that is generated by the interior performance criteria. This is just a conceptual example. The complexity of the exterior polyhedron can be increased to address the complexity that a particular contextual condition demands.

3.2 Application of effectors in architectural practice

Recently, a number of projects have been published that have used dynamic and non-linear computational processes to generate architectural designs. A recent issue of *Architectural Design* focusing on contemporary processes in architecture features two projects that can be examined for their relationship to the concept of poche'.

An outstanding project featured in the issue (Architectural Design, 2000), Embryologic Houses by Greg Lynn, uses the concept of effectors, but in a limited way. Lynn describes the underlying concept of his Embryologic Houses thus, "the variations in specific house designs are sponsored by the subsistence of a generic envelope of potential shape, alignment, adjacency and size between a fixed collection of elements." This generic envelope that is subject to mutation is composed of 2048 panels, 9 steel frames and 72 aluminium struts defining a shell. The form and space of the houses are modified within the predefined limits of the components. This is analogous to a polyhedron with a fixed set of vertices, edge, faces and constraints. All the effectors (transforming agents or control points) in Lynn's project act on the generic envelope from the outside and do not mediate between an "interior" and an "exterior" requirement. In fact, the variations in the houses are described as an adaptation to "contingencies" of lifestyle, site, climate, construction methods, materials, spatial effects, functional needs and special aesthetic effects. In the prototyping stage, six houses were developed exhibiting a unique range of domestic, spatial, functional, aesthetic and lifestyle "constraints." How these "contingencies" and "constraints" affect the generic envelope is not clearly articulated, so their role in generating the design cannot be determined. The transforming agents that mutate the generic envelope are causal agents and not mediating agents.

In another project featured in the same issue, Ali Rahim describes the operational principles in the generation of his competition winning entry for a Steel Museum in South Asia thus, "This (abstract machine) was comprised of vectors, fields, pressures and constraints in combination with inverse kinematics, particles and surfaces, embedded within the confluence of virtual matrices." Rahim's abstract machine, or machinic phylum as he prefers to call it, involves the causal transforming effect of particles and vectors (the flock of contaminants) on virtual matrices (the unactivated field) and vice versa. The interaction between the two enables the design. Though there is an exchange between the particles and the virtual matrices, again there is no mediation between an "interior" an "exterior" requirement. The actual spaces for the accommodation of program elements emerge from fluctuating intensities that indicate spatial potential. These intensities result from the indeterminate interaction between the particles and the matrices. The process is akin to crystallisation and annealing, or an act of congealing or solidification. Duration and temporal evolution determine space. There are no desiring, inhabiting forces that create spatial potential, like the need for a pleasant acoustical environment, thermal delight, or a view to the outside. There is a significant absence of interiority in the design generation processes that both these projects use, even though there is a creation of interiors in both of them. The forms and spaces created provide "opportunities" for occupancy, but there are no active occupying or dwelling forces that generate the forms and spaces. The concept of poché forces attention on this "interiority."

3.3 Future directions

With growing attention being focused on digital processes for architectural design, a well-defined mechanism, or an abstract machine that triumphs over mechanisms (Deleuze, 1988), needs to be developed to generate the process space for these design processes. The concept of effectors provides one such mechanism/machine. Effectors can be configured into various abstract machines that generate architectural designs. The concept of effectors can be the unifying concept that allows the computational modelling of all architectural entities as active agents.

In the editorial to the issue of *Architectural Design* (Architectural Design, 2000) devoted to contemporary architectural processes, an emerging field is defined that optimises the state of the "in-between" as process and "systemic delay" as a major source of creativity. The concept of "in-between", as used in some of the projects featured in the issue, is based on the concept of "tweening" used in animation systems and not on the concept of poché. The concept of poché provides a different "in-between" paradigm. Systemic delay is defined as conceptual development in the time lag between an initial idea and its material form. This can also be related to the concept of poché, if the space-time of poché (in its extreme characterisation, a poché can be a space-time continuum), is considered a systemic delay between an idea and its realisation in material form.

As such, the concept of effectors and poché can provide the means to mediate between idea and material form, between inside and outside, between performance criteria and space, in short, any condition that involves the mediation between two (or more) active principles. Though polyhedra are used in the examples in this paper, effectors can be used with completely curvilinear surfaces as well. Also, the polygons and polyhedra are not literal but represent networks of effectors that may constitute abstract machines.

4. REFERENCES

Architectural Design: Contemporary Processes in Architecture, 2000, Vol. 70, No. 3, Wiley-Academy, England, June, pp. 26-35 and pp.63-69.

Deleuze, G., 1988, *Bergsonism*, Translated by Hugh Tomlinson and Barbara Habberjam, Zone Books, New York, New York, pp. 107.

Deleuze, G. and F. Guattari, 1987, *A Thousand Plateaus: Capitalism and Schizophrenia*, Translated by Brian Massumi, University of Minnesota Press, Minneapolis, Minnesota, pp. 141 and pp. 510-514.

Mahalingam, G., 1998, "Representing architectural design using virtual computers," *Automation in Construction*, Vol. 8, Elsevier, New York, New York, pp. 25-36.

Mahalingam, G., 1996, "Object-oriented computer-aided design systems for the preliminary design of auditoria," *Journal of Architectural and Planning Research*, Vol. 13, No. 3, Locke, Chicago, Illinois, Autumn, pp. 214-229.

Venturi, Robert, 1966, *Complexity and Contradiction in Architecture*, Museum of Modern Art, New York, pp. 85.

Representing Architectural Design Using a Connections-based Paradigm

Ganapathy Mahalingam, Ph.D.

Abstract

Any *making*, including a work of architecture, is synthetic in nature and is made by making connections. To base the core of a computational representation of architectural design on *connections* is to base it on the very core of making. The articulation of the core of architecture, its architectonics, should be based on articulating its connections. This paper probes how connections can serve to represent architectural design. A paradigm consists of a core cluster of concepts that, for a time period, provides a framework to articulate the issues and problems facing a field and to generate solutions. This paper offers a connections-based paradigm to represent architectural design computationally. A number of connections-based strategies for the representation of architectural design have emerged. Modeling frameworks that have been identified include dendograms, bipartite graphs, adjacency graphs, plan graphs, planar graphs, Hasse diagrams, Boolean lattices and Bayesian networks. These modeling frameworks have enabled the representation of many aspects of architectural design. Is it possible to extract a uniform modeling framework from all these frameworks that enables the computation of architectural design in all its aspects? Using biological analogies, will an integration of these modeling frameworks provide the 'molecular' structure of a 'DNA' that makes up the architectural 'genome'? This paper will attempt to answer these questions.

1 Introduction

Resolving the computability of design has been a longstanding quest among researchers in computer-aided architectural design. One does not question whether design, as a cognitive activity, is possible, but one does question whether design is computable. Remarkable advances are being made in cognitive modeling using computer-based systems. The key to making design truly computable may lie in these advances in cognitive modeling. Researchers, who conclude that design is not computable, or is not computable in its creative aspect, invariably point out a computer-based system's inability to generate radically new forms. When you carefully examine what constitutes a radically new form, the answer that emerges is new connections at various topological levels.

Articulating the synthesis of forms or the generation of spatial organization has a long tradition spanning four decades. Beginning with Christopher Alexander's *Notes on the Synthesis of Form* (1964), the design profession has wrestled with the articulation of architecture. Lionel March and Philip Steadman's pioneering work on the geometry of the environment (March & Steadman 1971), followed by Lionel March's work on the architecture of form (March 1976) have set the precedent decades ago for what may now form the core for representing architectural design.

Articulating spatial organization found new energy in Hillier and Hanson’s work on the social logic of space (Hillier & Hanson 1984). Hillier and his team have recently expanded their research to the broader framework of space as a machine (Hillier 1996). In doing so, they have begun to make the case for a non-discursive, analytical theory of architecture based on ‘configurations.’

In his seminal work from the early 60s, Alexander attempted to get at the core process in the synthesis of forms, an issue central to architecture. In the preface to a later edition of his book on the subject (Alexander 1964), he carefully articulated his quest and its significance. He wrote:

> “In this book I presented the diagrams as the end results of a long process; I put the accent on the process, and gave the diagrams themselves only a few pages of discussion. But once the book was finished, and I began to explore the process which I had described, I found that the diagrams themselves had immense power, and that, in fact, most of the power of what I had written lay in the power of these diagrams. The idea of a diagram, or pattern, is very simple. It is an abstract pattern of physical relationships which resolves a small system of interacting and conflicting forces, and is independent of all other forces, and of all other possible diagrams. The idea that it is possible to create such abstract relationships one at a time, and to create designs which are whole by fusing these relationships—this amazingly simple idea is, for me, the most important discovery of the book.”

This book led to the widespread use of diagrams or patterns as the basis for designs, especially architectural designs. The subsequent search for diagrams and patterns extended the scale attempted by Alexander in his book, the scale of a village in India. The thought that diagrams could form the basis of designs took firm root. This paper is about an attempt to extend the tradition that began with Alexander’s work into a new realm made possible by advanced computing techniques.

2 Connections-based Representations

A connections-based representation is quite simply one that uses connections as an organizing framework for the representation. In this paper, the term “connections-based” is deliberately used rather than “connectionist,” which has its own connotations. The connectionist model is a subset of the broader category of connections-based representations.

The term “connectionist” is widely used to describe a computational technique used to model the human brain, especially its neural network. A connectionist model is made up of a network of many simple processing units that act in parallel to produce “emergent” behavior. These simple processing units have been described as intuitive, sub conceptual and sub symbolic entities that are linked in a dynamic system that does not allow a precise conceptual level description. There is a healthy and ongoing debate about the effectiveness of the “connectionist” model to represent the working of the human brain.

A connections-based representation, on the other hand, is a diagram made up of multiple nodes that are linked in various ways. The word “diagram” is based on its Greek roots *dia* and *graphein*, which mean “through” and “write” respectively. The Greek verb *diagraphein* means to “mark out by lines” from which the noun *diagramma* is derived. Originally the word diagram referred to a geometrical figure, and for a brief period even to a written list or register, which is very curious. Based on its etymology, the word “diagram” refers to an intrinsic structure, something that is drawn “through” an entity, like a skeletal framework. When coupled with the etymology of the word “understanding,” diagrams provide intrinsic knowledge of entities. One of the goals of connections-based representations is to help acquire this intrinsic knowledge of architecture. Architecture like connections must be made and is not given (Rajchman 2000). The primary architectural act can be considered as the linking of two nodes. This is the beginning of synthesis and a plurality. Starting with this primary connection-based representation (Figure 1), a hierarchy of connections-based representations can be articulated. Connections-based representations that have emerged in research can be organized based on increasing complexity. Some of the representations that have emerged include the ones described in the following sections.

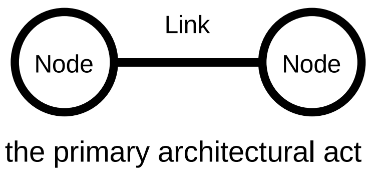


Figure 1. The architecture of connections.

2.1 Dendograms

A dendogram is a diagram that has a branch-like structure (Figure 2). Starting from a single node, branches or links lead to successive nodes. Examples of dendograms are parse trees, decision trees and binary trees. Each terminal node of a ‘tree’ representation is appropriately called a leaf. In representations such as decision trees, the leaves represent outcomes that are a result of decisions made at the nodes. Parse trees can be used to verify if a particular architectural composition has been created using a particular architectural language. Decision trees can be used to represent a design process as a hierarchical sequence of design decisions, where each design decision leads to subsequent design decisions. Because of their architecture, dendograms are useful in representing hierarchical procedures or processes. A dendogram or tree of (n) nodes has (n-1) links. Figure 3 illustrates this rule.

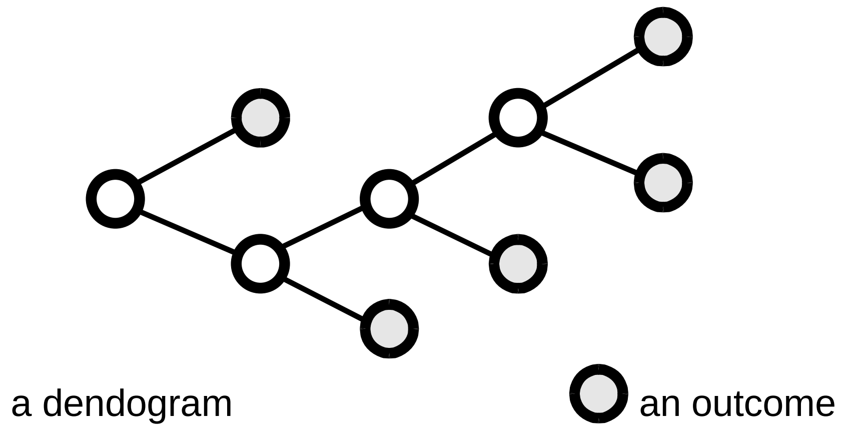


Figure 2. A dendogram.

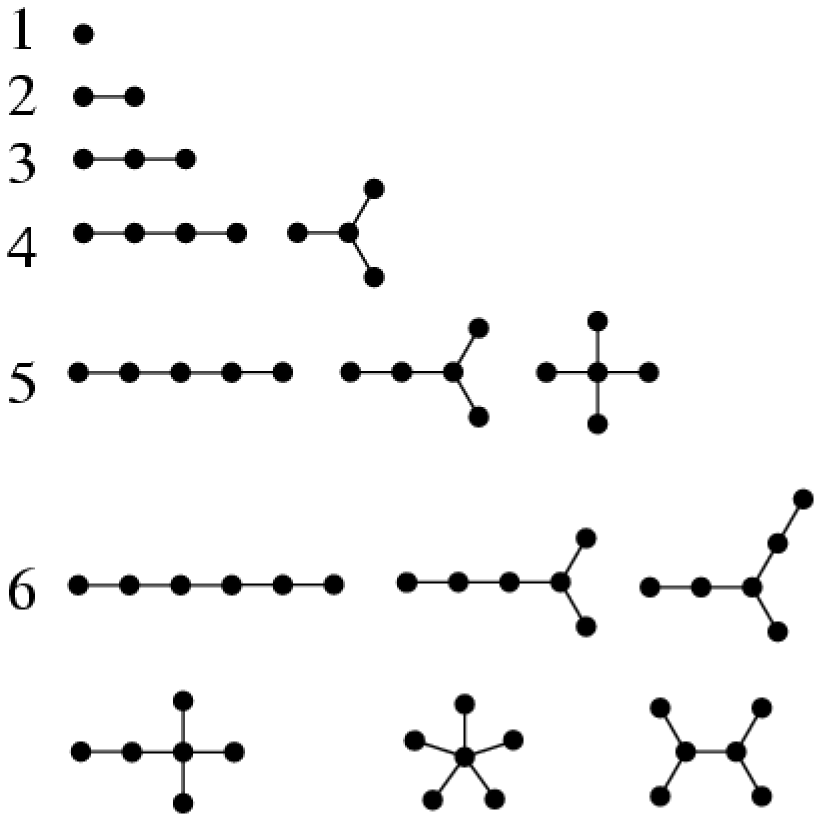


Figure 3. Dendograms of increasing order, i.e., number of nodes (image courtesy Weisstein, 1999-2003).

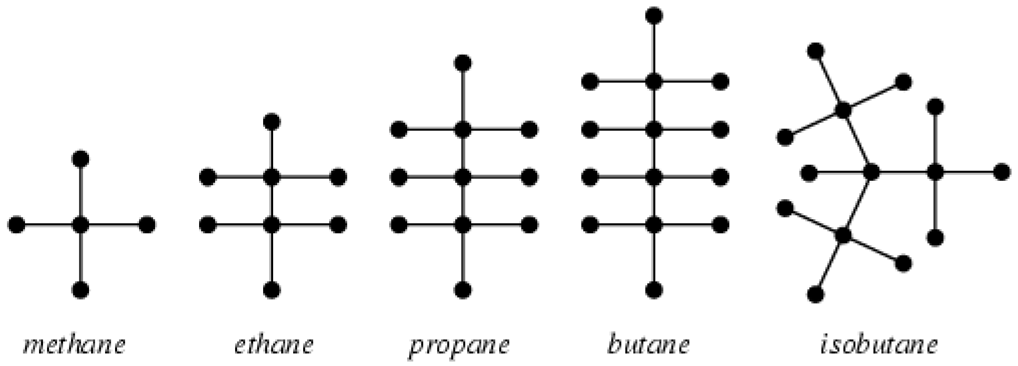


Figure 4. Molecular structures as dendograms (image courtesy Weisstein, 1999-2003).

Dendograms are especially useful in the modeling of molecular structures (Figure 4) and by analogy the relationships between spaces in a building. When the nodes of the dendogram are used as the insertion points or instantiation points of polyhedra representing spaces, dendograms can form the skeletal generating framework of the complex spatial form of a building. Dendograms can also be used to model the load-transfer action in structural assemblies. Each node represents a structural component and each link a load transfer path.

When dendograms are used to represent design processes, they become the representation of time. In this case, the dendogram is used as a state-transition graph. Each node of the dendogram represents the state of a design at a particular time. Though the state-transition graph extends in space, the spatial boundaries of the entity whose evolution is being described by the state–transition graph can be fixed. When used as a decision tree, dendograms become design decision paths in action space that are traversed in time.

A more general case of the dendogram, which is not a hierarchical tree, called a permeability map, was developed by Hillier and Hanson (1984) to represent the privacy gradient in a set of spaces.

2.2 Bipartite Graphs

A bipartite graph is a connections-based representation whose nodes can be partitioned into two sets such that no two nodes in any set are adjacent (Figure 5). In a complete bipartite graph, in addition, every node in one set is connected to every node in the other set. A tree is also a bipartite graph. Bipartite graphs are useful when the representation has two distinct set of elements that are related to each other but not amongst themselves. Bipartite graphs are also used to model a type of representation called a Petri Net, especially the channel-agency form of the Petri Net. Petri Nets are used to model hardware devices, communication protocols, parallel programs and distributed databases.

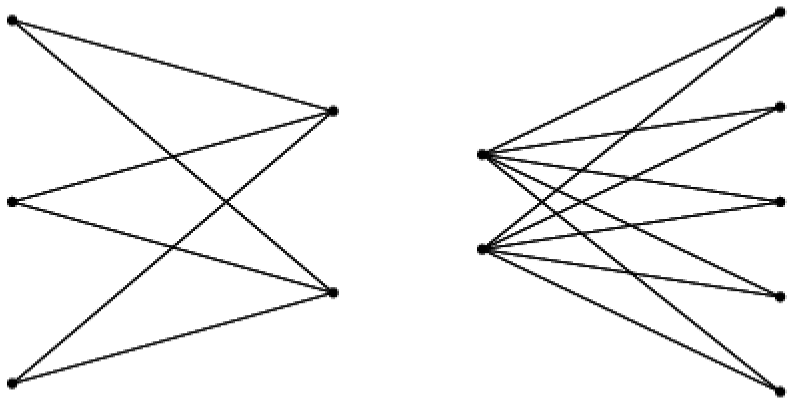


Figure 5. Bipartite graphs K $_{3,2}$ and K $_{2,5}$ (image courtesy Weisstein, 1999-2003).

The bipartite graph can be used to model a situation where a set of environmental sensors interacts with a set of architectural elements. A complete bipartite graph exhausts all the relations between the components in such as system.

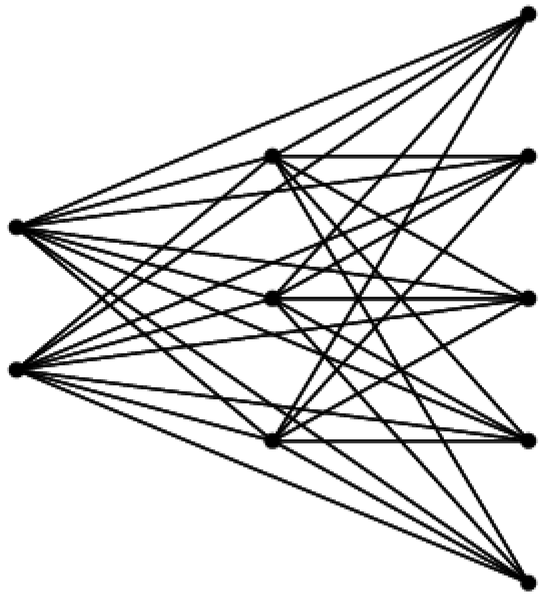


Figure 6. A 3-partite graph K $_{2,3,5}$ - an example of a k-partite graph (image courtesy Weisstein, 1999-2003).

The model of the environmental performance of an architectural space where there is a set of sources that generate environmental performance criteria, a set of receivers, that is, inhabitants who experience these environmental performance criteria, and a set of architectural elements, can be represented by a complete 3-partite graph (Figure 6) that exhausts all the relations between the various components.

Bipartite graphs can also be used as armatures for architectural designs. A notable example of the use of bipartite graphs in the arts is the use of a $K_{18,18}$ bipartite graph (Figure 7) in Umberto Eco's *Foucault's Pendulum*.

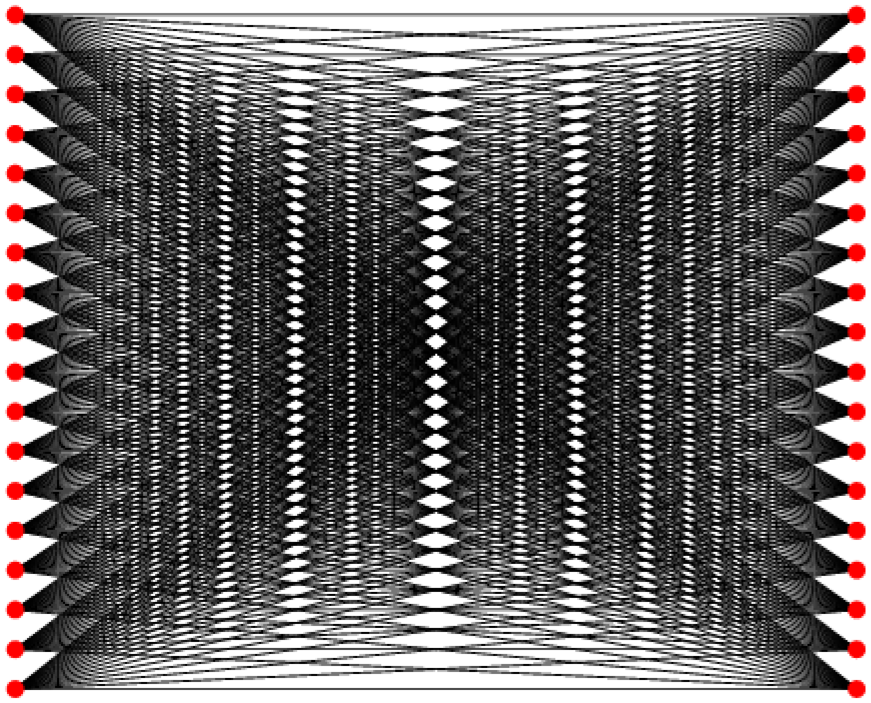


Figure 7. Umberto Eco's K $_{18,18}$ bipartite graph armature in *Foucault's Pendulum* (image courtesy Weisstein, 1999-2003).

2.3 Adjacency Graphs

In an adjacency graph, each separate space is represented as a node. Spaces that are in contact with another, that is, they are adjacent, are connected by links. In this representation, spaces that are connected only at corner points are not considered adjacent. The general exterior space is also represented as a node. All the 'interior' nodes connect to this general 'exterior' node. Adjacency graphs and their alternate form of representation, adjacency matrices, have been used in architectural design to establish proximal relations between spaces. When the duals of planar adjacency graphs are drawn, the 'wireframe' plan of a set of spaces can be generated. In a recent project (Hwang & Choi 2002), adjacency graphs were used as metadata for information retrieval in a spatial information storage system.

In an interesting analysis performed by Steadman (March & Steadman 1971), different designs by the architect Frank Lloyd Wright for different clients and sites were shown to have the same adjacency graph as the basis for the organization of spaces.

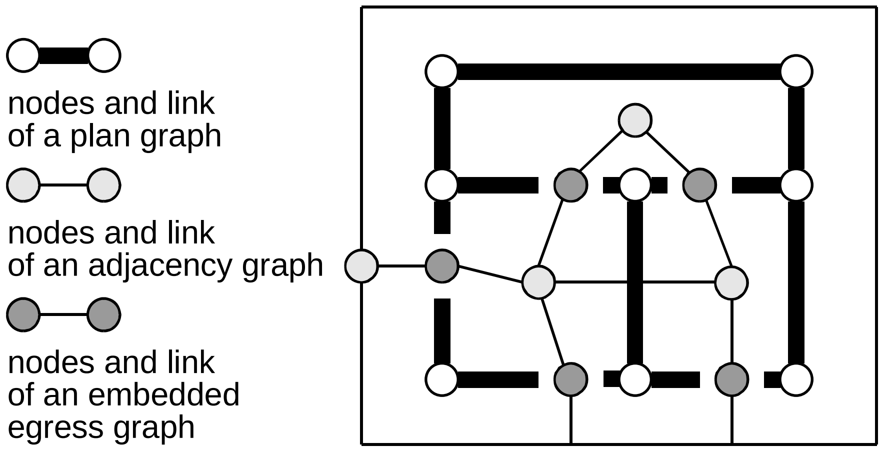


Figure 8. Plan graphs, adjacency graphs and embedded graphs.

2.4 Plan Graphs and Planar Graphs

In a plan graph (Figure 8), the junctions between walls in an architectural plan are represented as nodes and the walls themselves are represented as links. In this representation, the representation of “walls” is not restricted to physical barriers alone, but includes other divisions of space as well. A plan graph of a set of spaces is related to its adjacency graph. One is called the dual of the other.

A planar graph (Figure 8) is quite distinct from a plan graph and it is easy to be confused by the similar sounding terms. A planar graph is a graph that can be drawn on a plane without any of the links crossing each other. A completely connected planar graph (Figure 9), that is, a graph in which each node is connected to every other node, cannot have more than 4 nodes. This implies that to maintain a complete set of relations between more than four components in a connections-based representation requires three-dimensional spatial thought.

Because of the way in which a plan graph is constructed, it is always planar. Plan graphs and adjacency graphs can be integrated with other graphs, which can be embedded in them. The example shown in Figure 8 represents the modeling of an egress pattern in the floor plan of a building. Each egress element, a door or a window, is represented as a node. This node is embedded in the link between nodes that represent spaces in an adjacency graph of the plan. The egress node is also embedded in the plan graph of the floor plan. If this representation is used to simulate egress from a building during an emergency such as a fire, then traversing the graph can establish whether there is a safe egress path to the exterior of the building.

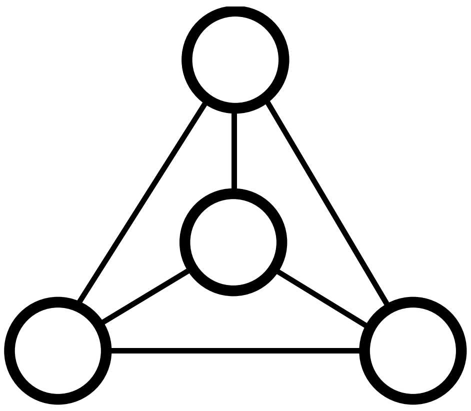

a completely connected planar graph
of the highest order (4 nodes)

Figure 9. A completely connected planar graph.

2.5 Hasse Diagrams

According to March (1976), a Hasse diagram (Figure 10) is a diagram of connected nodes such that you can move from one node to another through a set of one or more “upward” links. As such it can be used to model “directional” synthesis of any set of entities. The Hasse diagram defines a progression from a null set to a full set of entities, where each intermediate set is a cover (a mathematical relation) of the immediately preceding set or sets. The directional buildup of an architectural design or a conceptual map that defines an architectural design as it evolves incrementally can be represented by a Hasse diagram.

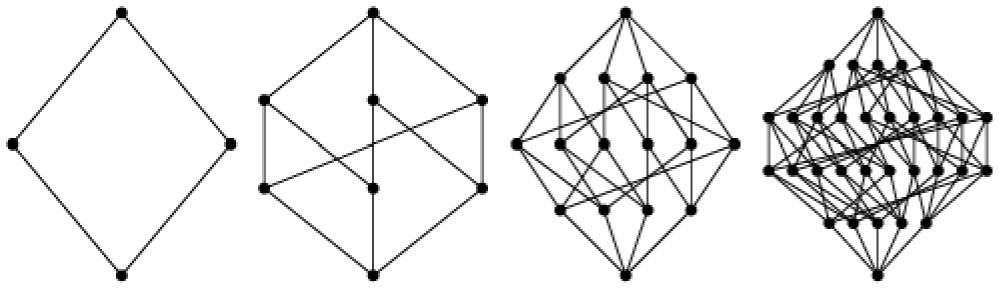


Figure 10. Hasse diagrams of sets with 2, 3, 4 and 5 entities (image courtesy Weisstein, 1999-2003).

2.6 Boolean Lattice

According to March (1976), a Boolean lattice is a representation of Boolean algebra as a complemented, distributive lattice. A Boolean algebra b(A) of a set A is the subsets of A that can be obtained by a finite number of the set theoretic operations of OR (union), AND (intersection) and NOT (complementation). Each element of b(A) is called a Boolean function. The number of Boolean functions of a set of 2 entities (say 1 and 0 as in a binary system) is 16. Computing circuitry is based on a Boolean algebra of 2 entities. The abstract structure of the Boolean algebra is isomorphic to specific algebras used in set theory, the algebra of events, symbolic logic, switching algebra and automated process control. If the sequence of design operations that generates an architectural design can be represented by Boolean lattice, then it can be automated.

2.7 Bayesian Networks

A Bayesian network (Figure 11) has been described as a "belief' network. Each node in a Bayesian network represents the probability of a variable (a Bayesian variable) in a system. Nodes are linked to each other based on conditional dependence. The network is based on a probability model and distribution. The connections are causal connections and are directional. The direction is always from cause to effect. The dependent node is called a 'child' and the influencing node is called the 'parent.' Time is also introduced into the model as a parent is the temporal antecedent of a child. In a complex Bayesian network, many cycles of dependencies can be set up. Once a Bayesian network has been set up, a variable is given a value based on observation. Calculations are then performed to find the values of all other variables based on their probability of occurring. Once all variables are established, the network defines the probable state of the system.

A Bayesian network can be used to create a predictive model of environmental performance criteria in the design of an architectural space. Such environmental performance criteria can include temperature, illumination and sound intensity.

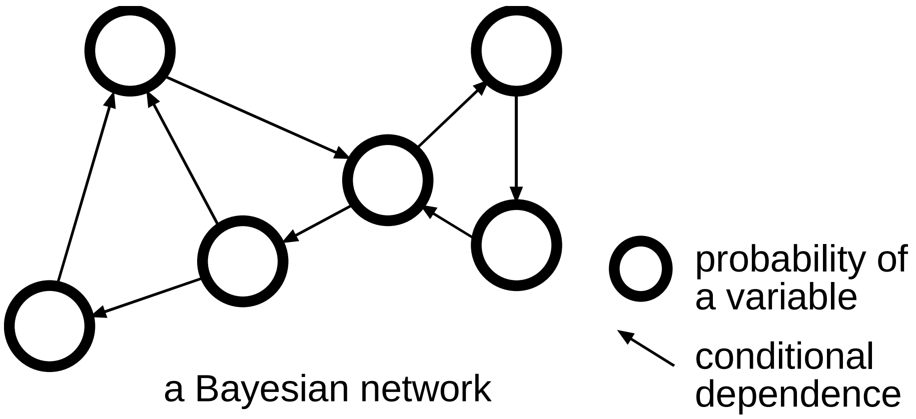


Figure 11. A Bayesian network.

Table 1: Connections-based representations and their use in the representation of

architectural design

Connections-based Representation	Architectural Representation
Dendogram	Relationship of spaces, Load-transfer in structures
Bipartite Graph	Environmental control using sensors, Modeling of environmental performance
Adjacency Graph	Relationship of spaces, Relationship of surfaces in a space
Plan/Planar Graph	Architectural floor plans, Surface distribution in spaces
Hasse Diagram	Architectural design generation using a kit of parts
Boolean Lattice	Automated architectural design generation
Bayesian Network	Prediction models for environmental performance

3 Modeling Architectural Design Using Networks

In the past, the computational representation of architecture, both as a product and as a process, has utilized a wide range of representational frameworks. Architecture has been represented as data structures, databases, procedures, algorithms and virtual computers. A connections-based representational framework now extends this range of representations to include networks (Table 1).

3.1 Elements of a Network

A network has only two basic elements, nodes and links. The complexity of networks is based on the number of nodes and their interconnections or links. When used as a representational framework, networks provide the following opportunities:

- Nodes: the representation of state (properties, variables, parameters, probability elements)
- Links: the representation of relations (constraints, semantics, physical connections, causal connections, transformations, transfer functions, dependencies)
- Networks: the representation of state, the representation of a process, the representation of probabilities of outcomes
- Network of networks: the representation of complex spatial systems

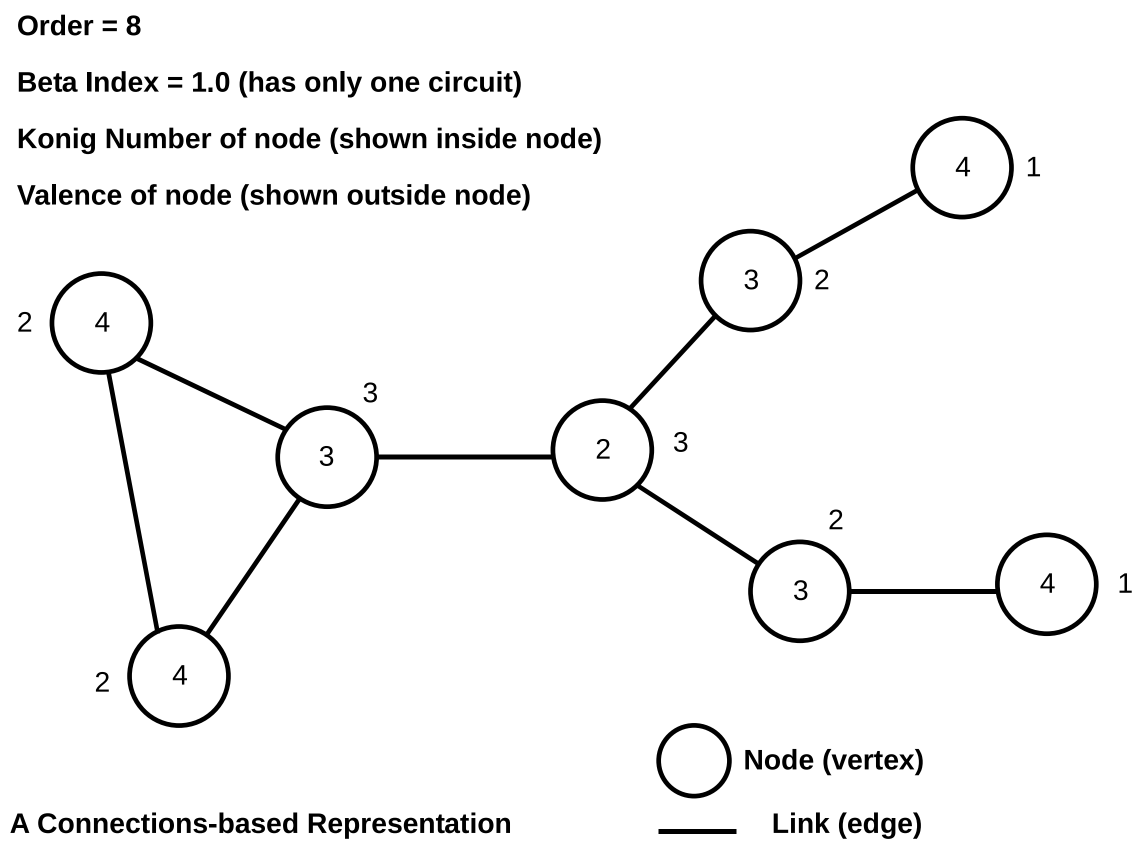

Order = 8
Beta Index = 1.0 (has only one circuit)
Konig Number of node (shown inside node)
Valence of node (shown outside node)
4
1
3
2
2
4
3
3
2
3
2
3
4
1
2
4
Node (vertex)
A Connections-based Representation
Link (edge)

Figure 12. Properties of networks.

3.2 Properties of a Network

The various properties of networks (Figure 12) lend themselves to various representations of architecture. Some of the properties include:

Hamilton Path/Hamiltoninan Cycle or Circuit: A path in a network that starts at a starting node, goes through each node only once, is not obligated to traverse each link and ends at the starting node. It describes a circuit (Figure 13).

Euler Path/Eulerian Cycle or Circuit: A path that traverses each link of the network once, with no restrictions on the number of times it goes through a node. An Euler path is possible in a network only if the network is connected and no more than two nodes have an odd valence.

Order of the Network: The order of a network is the total number of nodes in the network.

Valence: The number of separate links to a node.

Konig Number of a Node: It is the maximum number of links in the shortest path to connect a particular node in the network to any other node in the network. The Konig number is used to establish the 'centrality' of node in a network.

Beta Index: It is a measure derived by dividing the number of links in a network by the number of nodes in the network. A network with a *beta* index of less than 1.0 is a tree, a network with a *beta* index of 1.0 has only one circuit and a network with a *beta* index of greater than 1.0 is a complex network.

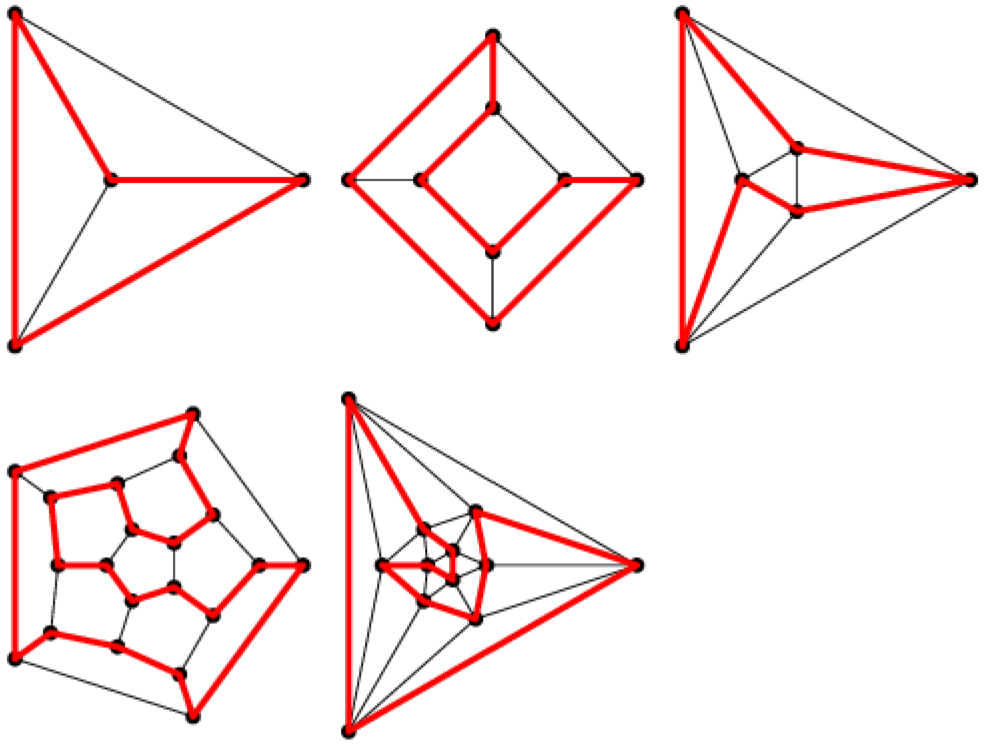


Figure 13. Hamiltonian circuits in graphs of Platonic solids (image courtesy Weisstein, 1999-2003).

4 Advantages of a Connections-based Representation of Architecture

The proposal for a non-discursive, analytical theory of architecture (Hillier 1996), may on the surface look like an erosion of a much-fought-for-and-gained political freedom in architectural expression, but in the end, may turn out to be more liberating than political freedom.

The single major advantage of using connections-based representations is the potential for distributed representation. Artificial intelligence researchers modeling the working of the brain distinguish between two kinds of representations, symbolic representations and distributed representations.

Symbolic representations use symbols such as words and numbers. These symbolic units have meanings associated with them. These units are combined into propositions in a language using the grammar(s) of that language. For example, words are combined into sentences (propositions) using the grammar of the English language. Similarly, numbers can be combined into mathematical propositions using a mathematical grammar. The main disadvantages of symbolic representations are that they are language-based and propositional; they are "brittle" and not fault-tolerant. Symbolic representations are considered "brittle" because symbolic units either exist or do not. A word is there in a sentence or it is not. Symbolic representations are not considered fault-tolerant because minor damage to a symbolic conceptual structure can cause the loss of the entire concept.

Distributed representations are ones in which meaning is not captured in a single symbolic unit, but arises from the interaction of a network of units. The common example that is given to illustrate this is that the concept "grandmother" is not stored in a single "grandmother cell" in the brain, but in a pattern or network of interacting neurons (brain cells). Distributed representations now provide the foundation for realistic computational models of human cognition related to visual, olfactory, auditory and tactile perception. Connections-based representations combine the features of symbolic representations (their structural sensitiveness) and distributed representations (their sensitiveness to statistical distributions of low-level perceptions) making them the ideal representational framework.

5 The Future of the Paradigm

The future of this paradigm lies in its ability to uncover the core of architecture, its architectonics. The architectonics of architecture may well be the architectonics of human thought. Rather than architecture being a theater of memory, through this paradigm architecture stands to be revealed as the theater of thought.

Stephen Grand OBE, a researcher from the UK and the developer of the computer game *Creatures,* has been developing an intelligent robot called Lucy. His goal is ensure that Lucy graduates from nursery school. Based on his research, Grand believes that the crucial element for intelligence is a particular circuit of neurons in the cerebral cortex of the brain that enables learning. Grand's goal is to unravel this circuit and use it to create an alternative to the digital computer that is similar to a living system. Since architecture is created by some of the most exacting neural processing known to humans, the key to this neural circuit could conceivably lie in a work of architecture. A connections-based coding, hence understanding, of this work of architecture, can lead to the discovery of this "learning" circuit.

On the other hand, Douglas Hofstadter, in his seminal book, *Gödel Escher Bach,* points out that the neural substrate of humans may pose barriers to certain processes of thought. He describes the problem encountered when someone tries to make "sense" of the Epimenides paradox thus:

> "Now my feeling is that the Tarski transformation of the Epimenides paradox teaches us to look for a substrate in the English-language version. In the arithmetical version, the upper level of meaning is supported by the lower arithmetical level. Perhaps analogously, the self-referential sentence which we perceive ("This sentence is false") is only the top level of a dual-level entity. What would be the lower level, then? Well, what is the mechanism that language rides on? The brain. Therefore one ought to look for a neural substrate to the Epimenides paradox—a lower level of physical events which clash with each other. That is, two events which by their nature cannot occur simultaneously. If this physical substrate

exists, then the reason we cannot make heads or tails of the Epimenides sentence is that our brains are trying to do an impossible task."

Hoftstadter proposes that when confronted with a situation such as making "sense" of the Epimenides paradox, the brain encodes the paradox in the neural substrate using "symbols" and processes it using "symbolic processing." He writes:

> "Now what would be the nature of the conflicting physical events? Presumably when you hear the Epimenides sentence, your brain sets up some "coding" of the sentence—an internal configuration of interacting symbols. Then it tries to classify the sentence as "true" or "false". This classifying act must involve an attempt to force several symbols to interact in a particular way. (Presumably this happens when any sentence is processed.) Now if it happens that the act of classification would physically disrupt the coding of the sentence—something which would ordinarily never happen—then one is in trouble, for it is tantamount to trying to force a record player to play its self-breaking record. We have described the conflict in physical terms, but not in neural terms. If this analysis is right so far, then presumably the rest of the discussion could be carried on when we know something about the constitution of the "symbols" in the brain out of neurons and their firings, as well as about the way that sentences become converted into "codings."

Such limits in neural processing may be why architecture is intrinsically homeostatic, that is, it does not change in its inherent structure. It may also be why all works of architecture can be created with a simple programming language. This paradigm will reveal such limitations in architecture, if they exist. Though this sounds pessimistic, there is hope for this paradigm as revealed by Alexander. In describing the creative potential of abstract diagrams, Alexander points out that these diagrams can evolve:

> "I have discovered, since, that these abstract diagrams not only allow you to create a single whole from them, by fusion, but also have other even more important powers. Because the diagrams are independent of one another, you can study them and improve them one at a time, so that their evolution can be gradual and cumulative. More important still, because they are abstract and independent, you can use them to create not just one design, but an infinite variety of designs, all of them free combinations of the same set of patterns."

Rather than use traditional genetic algorithms (Holland 1975) for the evolution of forms, which are used to transform one population of genetic characteristics (chromosomes) into another through processes of crossover (recombination), mutation and inversion, connections-based representation lend themselves to modeling based on biological analogies such as protein synthesis and morphogenesis (from developmental biology). Just as the molecular structure of DNA "instructs" protein synthesis, connections-based representations "instruct" the creation of works of architecture. A collection of connections-based representations (architectural DNA molecules – see analogy in Figure 4) then defines the "genome" of the field of architecture. Classifying this collection of connections-based representations, thereby defining the architectural genome, is a work on the scale of Durand's *Précis*. Unlike the process followed by Durand who focused on lines and delineation, mapping the architectural genome will focus on the underlying structure or architectonics of architecture. These underlying connections-based representations are distinguished by the fact that they do not belong to the measurement-based "metric" space of architectural modulation and variation but the "invariant" space of relationships.

6 Conclusions

One can conclude with a quote from Alexander that captures the spirit of a connections-based representation of design:

> "The shapes of mathematics are abstract, of course, and the shapes of architecture concrete and human. But that difference is inessential. The crucial quality of shape, no matter of what kind, lies in its organization, and when we think of it this way we call it form. Man's feeling for mathematical form was able to develop only from his feeling for the processes of proof. I believe that our feeling for architectural form can never reach a comparable order of development, until we too have first learned a comparable feeling for the process of design."

This paper offers *making connections* as a process of design which will allow that comparable order of development. The computational representation of architecture started with the representation of architectural entities as data structures and architectural design processes as procedures. It evolved into the representation of architectural entities and processes as virtual computers. The next stage in the evolution of the computational representation of architecture is the representation of architectural entities and processes as networks of virtual computers or computational entities. The examples of connections-based representations presented in this paper suggest that various aspects of architectural design from spatial synthesis to environmental performance control can be successfully represented using the techniques. Recent work by Wolfram (Wolfram, 2002) suggests that it is possible to go even further and model the evolution of networks, which can then be used as an overarching representational framework.

References

Alexander, C. (1964). *Notes on the Synthesis of Form*. Harvard University Press.

Durand, J. N. L. (1802) *Précis des leçons d'architecture*. Paris: Ecole Polytechnique.

Hillier, B. and J. Hanson. (1984). *The Social Logic of Space*. Cambridge University Press.

Hillier, B. (1996). *Space is the machine: A configurational theory of architecture*. Cambridge University Press.

Hofstadter, D. (1980). *Gödel Escher Bach.* Vintage Books Edition.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Hwang, J. and J. Choi. (2002). SpaceCore: Metadata for Retrieving Spatial Information in Architecture. In *Proceedings of the ACADIA 2002 Conference,* pp. 199 - 217.

March, L., Editor. (1976). *The architecture of form*. Cambridge University Press.

March, L. and P. Steadman. (1971). *The geometry of environment: An introduction to spatial organization in design*. RIBA Publications Ltd.

Rajchman, J. (2000). *The Deleuze Connections*. MIT Press.

Skiena, S. (1990). *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Reading, Massachusetts: Addison Wesley.

Weisstein, E. W. (1999-2003). *Eric Weisstein's World of Mathematics* (*MathWorld*™) http://mathworld.wolfram.com/

Wolfram, S. (2002). *A New Kind of Science*. Champaign, Illinois: Wolfram Media, Inc.

IMPROVING 'OBJECTIVE' DIGITAL IMAGES WITH NEURONAL PROCESSING

A Computational Approach

GANAPATHY MAHALINGAM, RAJESH G KAVASSERI
North Dakota State University, USA

Abstract. This paper describes an experiment where an image recorded with a digital camera is processed using an electro-physiological model of a neuron. The luminosity level of each pixel of the source image is treated as the stimulus for an individual neuron, and the source image is transformed into a response image based on the processing behavior of the Hodgkin-Huxley neuronal model. It is seen that transformation of the image through neuronal processing yields (i) more evenly balanced levels of luminosity compared to the image directly recorded by the digital camera and (ii) a more `subjective' rendering of the environment than what was photographed with the digital camera. The CCD (charge coupled device) - based digital camera reveals its limitation as a linear recording device that does not have a balanced dynamic range. The neuronal processing of the image adds non-linearity and a balanced range to the luminosity levels in the image, rendering it closer to a 'subjective' perception of the scene.

1. Introduction

The use of digital media by design professionals has become widespread. Design professionals such as architects and interior designers are using digital images to make design decisions about built environments at different scales. Many of these design decisions are based on the luminosity levels in the images, and the levels of contrasts between the luminosity levels. If these levels, and their differences, are based on what a CCD sensor 'sees' rather than a 'eye-brain' perceptual mechanism, then the design decisions made using objective digital representations of images may lead to unanticipated and unintended subjective experiences of the built environment. These objective digital images are generated based on computational models that are physics-based. They provide an accurate rendering of the built environment based on objective measurements of luminosity levels. In that sense, they represent 'ideal' mathematical visions of the environments they portray. These renderings do not reflect the subjective processing of the scene that occurs when the neuronal cells in the brain process the 'raw' sensory data.

A common problem that occurs when one takes a grayscale photograph (commonly referred to as a black and white photograph) with a digital camera is that the image produced by the digital camera does not reflect our perception of the scene. This is because the digital camera uses a CCD sensor that records luminosity levels in an objective manner. In keeping with the scientific method, these levels are objective measures that are recorded by the instrument without any subjective interference.

One cannot predict if one is going to get a balanced image when one takes a grayscale photograph with a digital camera by just surveying the scene. The skills of photographers like Ansel Adams lay predominantly in their ability to survey a scene and deduce that the scene would indeed produce a brilliantly balanced grayscale photograph. One of the challenges we addressed in our experiments was to see if we can account for this discrepancy between an 'objective seeing' and a 'subjective seeing,' at least in the narrow realm of luminosity levels of digital representations of images.

In our experiments we took digital images and processed those using electro-physiological models of neurons to see what emerges when a digitally encoded image is processed ‘neuronally.’ Our results revealed that the neuronal 'adjustment' to the objective luminosity levels in the source image presented a much more balanced and clearer image that was in tune with subjective perceptions.

2. Methodology

The methodology we employed is illustrated in Fig. 1. It is a multi-stage process that is implemented in the software package MATLAB.

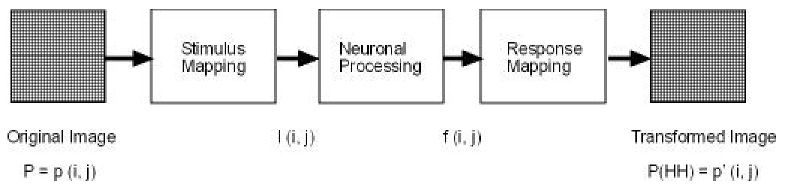


FIGURE 1. Model of method used in the experiment.

The original picture with $N \times N$ pixels is denoted as $P = p(i,j),\ i,j = 1\ldots N$ where $p(i,j)$ represents the luminosity level of the pixel (i,j). Next, we regard the luminosity of each pixel as the stimulus to a neuron so that its effect may be represented by an input current $I^{inp}(i,j)$. The current $I^{inp}(i,j)$ is then fed as a steady input current to a neuron represented by the Hodgkin-Huxley model (Hodgkin and Huxley, 1954). The neuron responds to this input current by producing an action potential with a frequency *f(i,j)* which depends on the strength of the input current. The ability of excitable neurons to encode the strength of the input in to a firing rate is used as a preliminary attempt to capture the input-output transduction process in visually responsive neurons. The firing rates *f(i,j)* are then mapped back to the range of luminosity levels contained in the original image to obtain the neuronally processed image $P^{HH} = p'(i,j)$ *,i,j = 1… N.*

2.1. NEURONAL MODEL

The biophysical model proposed by Hodgkin and Huxley (1954) has been one of the most important models in computational neuroscience. The Hodgkin-Huxley (HH) model is described by the time evolution of four variables *(v,m,n,h)* which represent membrane potential, activation of a sodium current, activation of a potassium current and inactivation of the sodium current respectively. The dynamical system for the model can be then described by:

$$C\, dv/dt = -g_l(v-v_l) - g_K n^4 (v - v_K) + g_{Na} h m^3 (v - v_{Na}) + I^{inp}$$
$$\tau_x\, dx/dt = x_\infty(v) - x$$

where $x \in \{m,n,h\}$. When the input current I^{inp} exceeds a certain threshold, the neuron is capable of displaying sustained oscillatory behavior. From a dynamical point of view, the bifurcation that determines the transition from a quiescent to oscillatory state determines the type of neural excitability in a given model (Izhikevich, 1999). Accordingly, we have two types of excitability (Rinzel and Ermentrout, 1989) namely,

Type I: neural excitability occurs when the rest potential (quiescent state) disappears after a saddle node bifurcation on a limit cycle.

Type II: neural excitability results when the quiescent state undergoes an Andronov-Hopf bifurcation.

The HH neuronal model employed here displays Type-II neural excitability where the frequency of oscillations at the onset of neural excitability is distinctly non-zero. In our case, the transition from rest state to repetitive firing is seen to occur through a supercritical Hopf bifurcation at $I^{inp} = 6.265\ mA$ corresponding to a frequency of $f = 52.5\ Hz$. As the input current is gradually increased, the frequency of oscillations increases as shown in Fig. 2. When $I^{ext} = 86.35\ mA$, the upper limit of the frequency of oscillations is reached, which is *140 Hz*. It follows that when the input current is in the range of *[6.26 - 86.35] mA*, the neuron fires with a corresponding frequency in the range of *[52.5 – 140] Hz*. The relation between firing frequency *f* and input current I^{inp} in the model employed is shown in Fig. 2. Regarding the current I^{inp} as the input and the firing frequency *f* as the output enables us to represent the input-output transduction of the neuron by the characteristics shown in Fig. 2. Thus we have attempted to capture the selectivity of the response of a neuron to different luminosity levels in a given image by the mapping process explained. Fig. 2 also shows (in dashed line) the mapping of a stimulus to a response based on the Weber-Fechner law, which maps psychophysical responses.

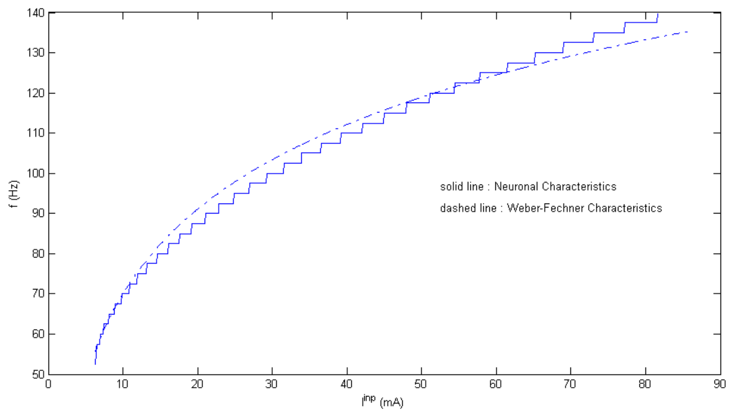


FIGURE 2. Mapping of the stimulus current to the response firing frequency based on the Hodgkin-Huxley neuronal model and the Weber-Fechner Law

3. Results

The proposed scheme was tested by processing four images acquired through a NIKON digital camera. The images were processed using the transformation model outlined in Section 1. The original and processed images are shown in Figures 3 - 10. Histograms of luminosity levels (see Figures 11, 12, 13 and 14) in the four sets of images were constructed. A summary of the mean luminosity levels for the four sets of images is shown in Table 1.

Image	Mean Luminosity of Original Image	Mean Luminosity of Processed Image	Ratio of Mean Luminosity Levels
1	93.75	142.84	1.5236
2	59.81	100.34	1.6772

3	86.98	139.73	1.6063
4	60.21	102.55	1.7032

TABLE 1. Mean luminosity levels of 4 sets of original and processed images.

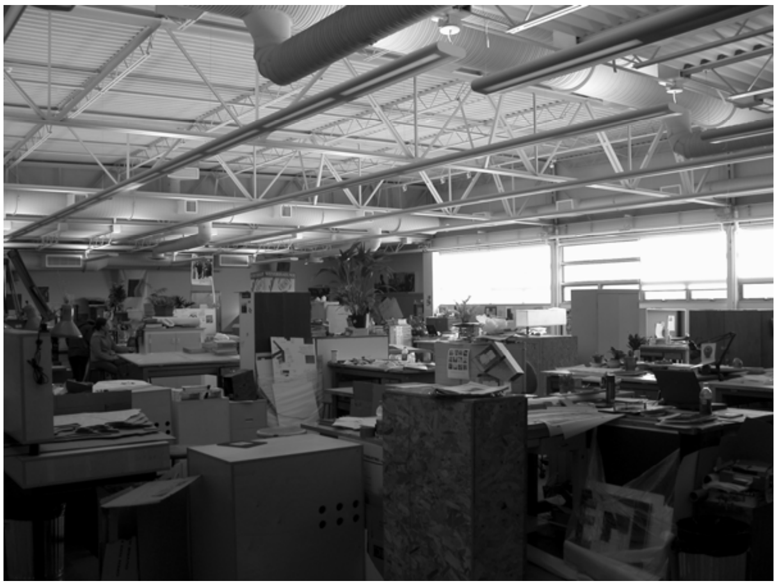


Figure 3. Photograph of a studio environment taken with a digital camera (Image 1)

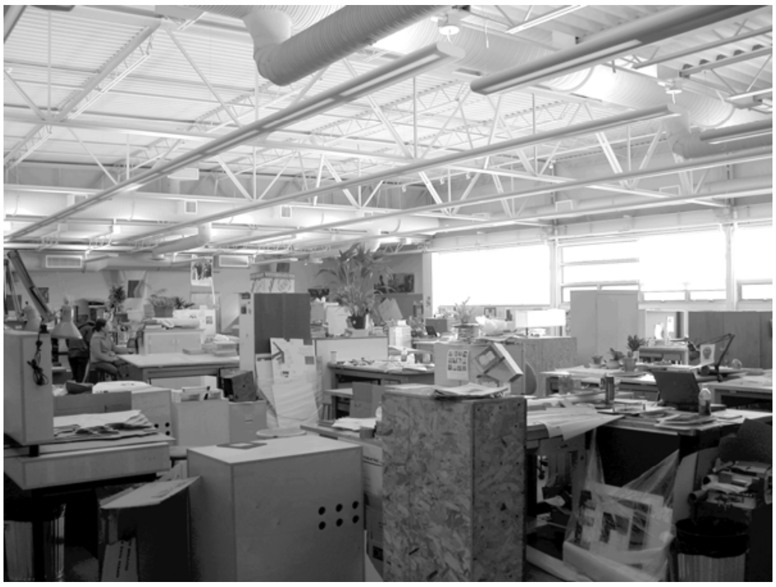


Figure 4. Image 1 that has been processed neuronally

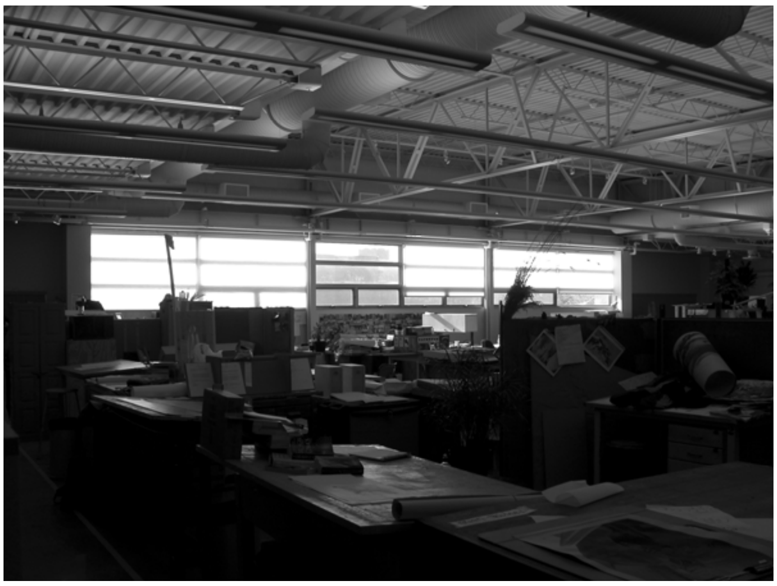


Figure 5. Another photograph of the studio environment taken with a digital camera (Image 2)

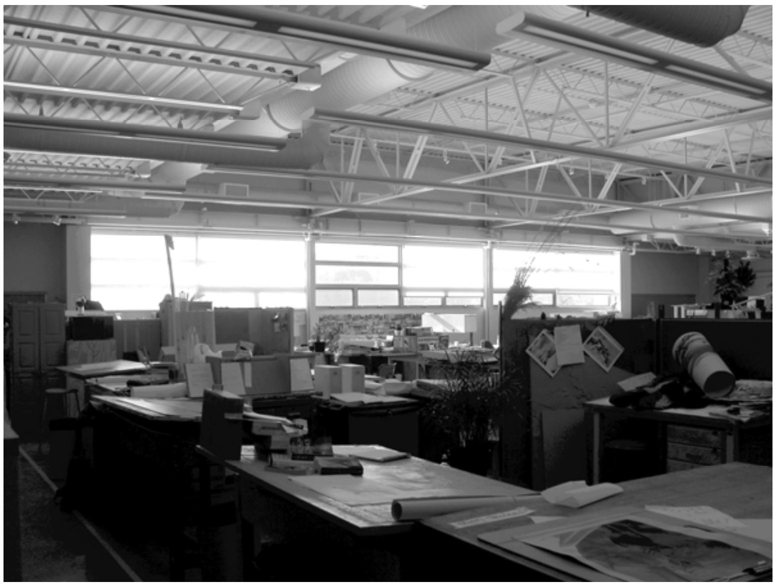


Figure 6. Image 2 that has been processed neuronally

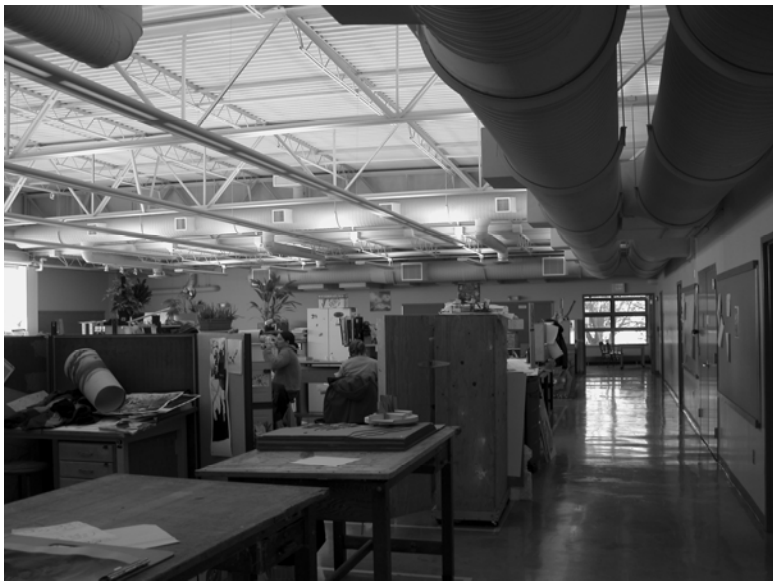


Figure 7. Another photograph of the studio environment taken with a digital camera (Image 3)

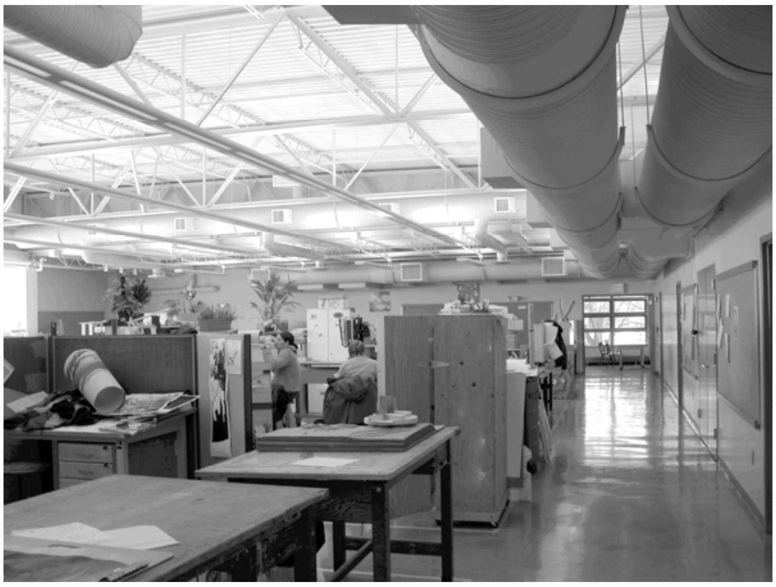


Figure 8. Image 3 that has been processed neuronally

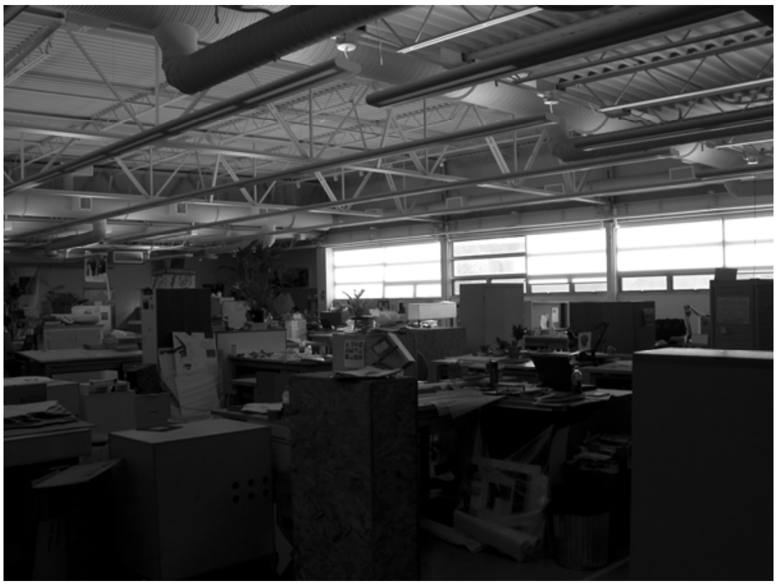


Figure 9. Another photograph of the studio environment taken with a digital camera (Image 4)

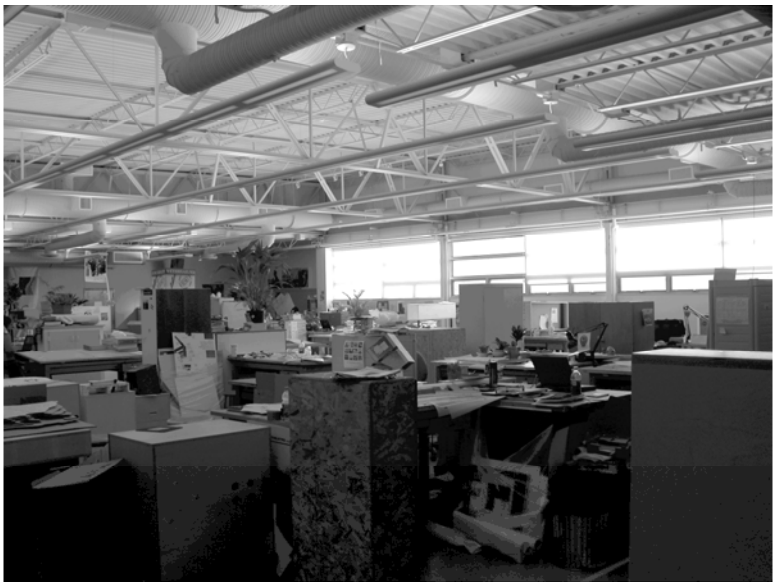
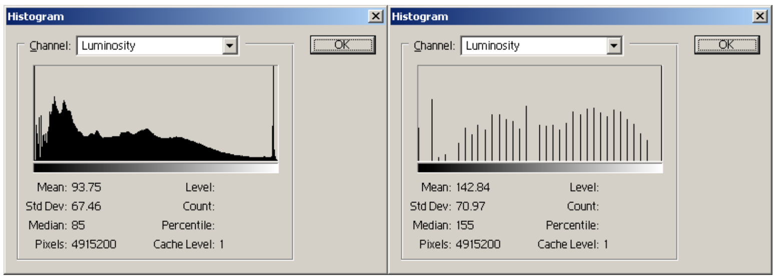


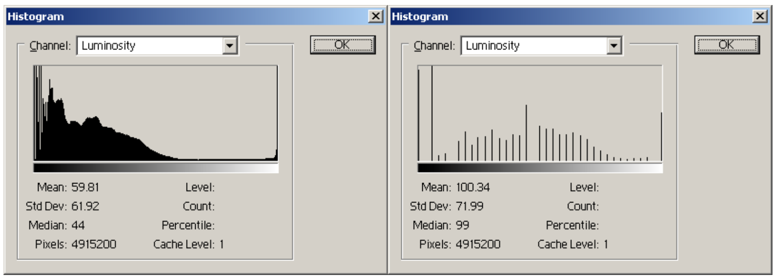
Figure 10. Image 4 that has been processed neuronally



```
Channel: Luminosity
Mean: 93.75        Level:
Std Dev: 67.46     Count:
Median: 85         Percentile:
Pixels: 4915200    Cache Level: 1

Channel: Luminosity
Mean: 142.84       Level:
Std Dev: 70.97     Count:
Median: 155        Percentile:
Pixels: 4915200    Cache Level: 1
```

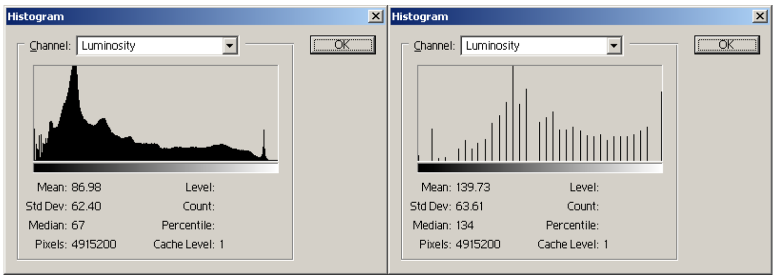
Figure 11. Histograms of luminosity levels of original and processed Image 1



```
Channel: Luminosity
Mean: 59.81        Level:
Std Dev: 61.92     Count:
Median: 44         Percentile:
Pixels: 4915200    Cache Level: 1

Channel: Luminosity
Mean: 100.34       Level:
Std Dev: 71.99     Count:
Median: 99         Percentile:
Pixels: 4915200    Cache Level: 1
```

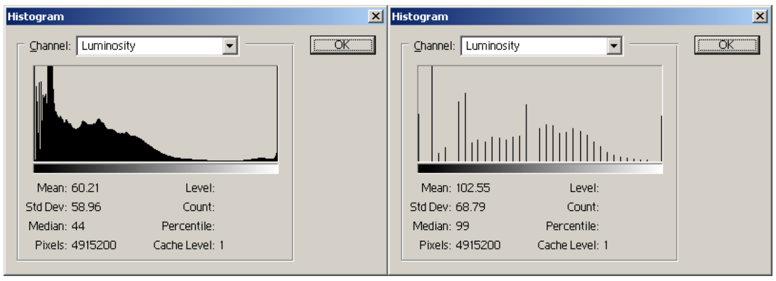
Figure 12. Histograms of luminosity levels of original and processed Image 2



```
Channel: Luminosity
Mean: 86.98        Level:
Std Dev: 62.40     Count:
Median: 67         Percentile:
Pixels: 4915200    Cache Level: 1

Channel: Luminosity
Mean: 139.73       Level:
Std Dev: 63.61     Count:
Median: 134        Percentile:
Pixels: 4915200    Cache Level: 1
```

Figure 13. Histograms of luminosity levels of original and processed Image 3



```
Channel: Luminosity
Mean: 60.21        Level:
Std Dev: 58.96     Count:
Median: 44         Percentile:
Pixels: 4915200    Cache Level: 1

Channel: Luminosity
Mean: 102.55       Level:
Std Dev: 68.79     Count:
Median: 99         Percentile:
Pixels: 4915200    Cache Level: 1
```

Figure 14. Histograms of luminosity levels of original and processed Image 4

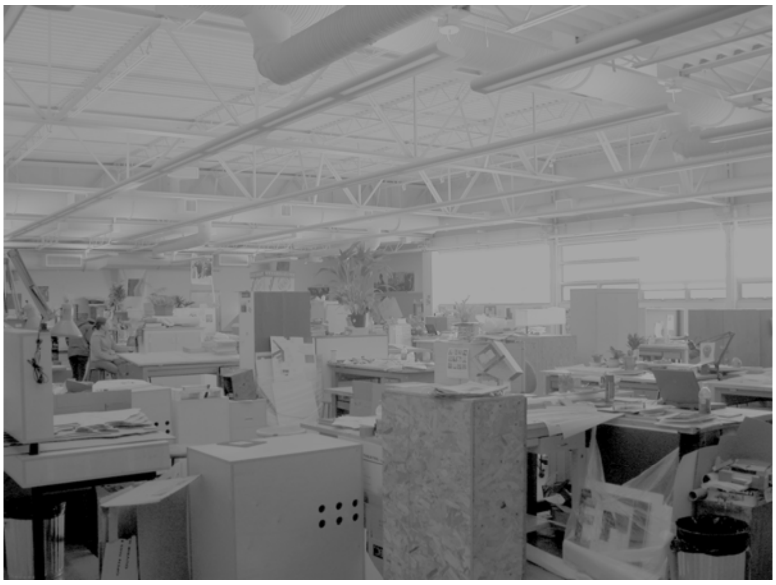
It can be noted from Figures 4, 6, 8 and 10 that the neuronal processing improves the visual appearance of the images. From Table 1, we note that the average luminosity level of the original image is increased by a minimum of 50 % upon neuronal processing. The histogram of the original Image 1 shows a concentration of luminosity levels in the “darker” ranges and a gradual waning as luminosities in the brighter range are approached. On the other hand, the processed image has a fair distribution of luminosities both in the lower and higher ranges. The histogram of the original Image 2 shows a strong concentration of luminosities from the low to midrange, and luminosities in the higher range are virtually absent. The histogram of the processed Image 2 however shows a fair concentration of luminosity levels in the entire range. A similar feature is observed from the histogram plots of the original and processed Images 3 and 4. Therefore, it is reasonable to say that the distribution of the luminosity levels is more balanced in the processed images compared to the original images. Thus, the neuronal processing model is seen to offer a marked improvement in the visual appeal of images by virtue of a balanced range of luminosities.

When we compare neuronal processing and the psychophysical response to a stimulus as predicted by the Weber-Fechner law, which is given by the relation:

$$S = klogI$$

where S = subjective response level, k is a constant and I is the stimulus intensity level, some interesting results are produced.

We see that the results produced by the neuronal processing are superior to the ones predicted by the Weber-Fechner law (Fig. 15). In order to compare the stimulus-response (input-output) mapping curve of the Weber-Fechner law with the stimulus-response mapping curve defined by the Hodgkin-Huxley neuronal model, we used a value for k = 70. Though the mapping curves seem to match rather closely (Fig. 2), the image produced by the neuronal processing is distinctly superior to the image produced by a stimulus-response mapping based on the Weber-Fechner law. This can be attributed to the difference between the continuous smooth curvature of the Weber-Fechner mapping curve and the stepped transitions in the Hodgkin-Huxley curve.



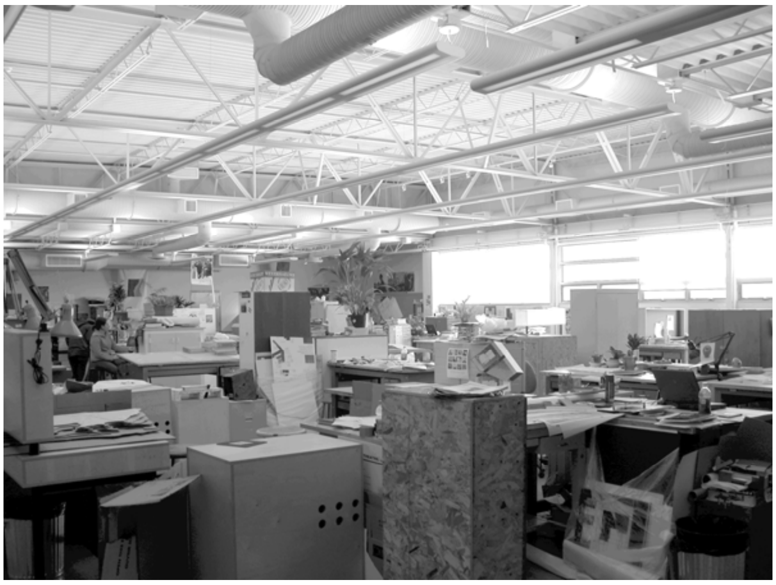


Figure 15. Comparison of processed Image 1, processed according to the Weber-Fechner law (above) and the Hodgkin-Huxley neuronal model (below)

4. Conclusions

We have shown that the neuronal processing of digital images of environments produces adjustments to the images that reflect our perception of the environments more closely. This approach would be a good post processing strategy for digital images generated by digital cameras and computer-based modeling and rendering software. In our approach, images that are 'objective,' and are generated by physics-based computational models are modified into images that are 'subjective' and generated by processing with electro-physiological neuron models. This type of processing enables design professionals, who use digital images, to make design decisions based on images that are closer to subjective perceptions. One of the assumptions that we made in this process was that the individual luminosity levels in the source images were connected in exactly the same way (as a grid) in the neuronal processing model. This need not be the case. A study of the variations in connectivity in the neuronal processing model based on the neuronal firing frequencies of individual luminosity levels or 'cells' can reveal other intricacies in the neuronal processing of the images, thereby revealing a more sophisticated 'subjective seeing' of the images.

References

Hodgkin, AL and Huxley AF: 1954, A quantitative description of membrane current and application to conduction and excitation in nerve, *Journal of Physiology*, **117**: 500-544.

Izhikevich, EM: 1999, Class 1 Neural Excitability, Conventional Synapses, Weakly Connected Networks and Mathematical Foundations of Pulse Coupled Models, *IEEE Transactions on Neural Networks*, 10 (3): 499-507.

Rinzel, J and Ermentrout, GB: 1989, Analysis of neural excitability and oscillations, *Methods in Neuronal Modeling*, MIT Press, Cambridge, Massachusetts.

DISCOVERING COMPUTATIONAL STRUCTURES IN ARCHITECTURE

An Exploration

GANAPATHY MAHALINGAM
North Dakota State University, U.S.A.

Abstract. The linkage between the worlds of Architecture, which involves the design and construction of the built environment, and Computer Technology, which involves practical applications of computation, still has a vast, as yet untapped potential. What if the implications of the linked term, 'computer-architecture,' are explored to reveal its full scope? This paper describes a unique method to analyze and code works of Architecture in a way that enables one to discover hidden computational structures in the works of Architecture. The case being made here is that the inherent structures of architecture may be computational structures as well.

1. Introduction

The term 'computer architecture' is often used in the computer industry and refers specifically to the design of computer systems, both hardware and software. Even Bill Gates, the head of Microsoft, prefers the title Chief Software Architect. This linkage between the worlds of Architecture, which involves the design and construction of the built environment, and Computer Technology, which involves practical applications of computation, still has a vast, as yet untapped potential. What if the implications of the linked term, 'computer-architecture,' are explored critically to reveal its full potential?

A work of architecture is created after an intense design process. The resultant architecture has embodied in it various formal structures (i.e., structures that articulate a particular form). The really interesting question is, are these formal structures, feasible computational structures as well? If the answer is yes, this will truly bring the world of Architecture into the world of computation! This project sets out as its main goal to discover and verify if the formal structures embodied in works of Architecture could serve as computational structures as well.

This project is the next in line of a long list of investigations completed by Mahalingam in the last decade linking the worlds of computation and architectural design. For his doctoral work Mahalingam successfully created an algorithm for the design of proscenium-type auditoriums. The algorithm was incorporated in object-oriented software for the design of proscenium-type auditoriums using the Smalltalk programming language and the VisualWorks software development environment. (Mahalingam, 1998, 2000). As a part of his doctoral investigation, Mahalingam also proposed a paradigm for the representation of architectural design entities as virtual computers (Mahalingam, 1997). This was a significant attempt to look at architectural entities as computational devices. In a subsequent investigation, a model was proposed for the parallel computational processing of load transfer in rectangular structural systems for architectural design (Mahalingam, 1999). A project was also completed where a programming language was proposed for architectural design with the complete Backus-Naur notation for the language (Mahalingam, 2000). In a more recent project, a new model was proposed for the sensor-based control of the propagation of sound in spatial enclosures based on an algorithmic model for sound propagation simulation developed earlier (Mahalingam, 1999). This project involved the modeling of the components involved as an elliptical graph called an *optimaton* (Mahalingam, 2005).

In a recent seminal paper, which has generated the main idea for this research project, a paradigm was presented for the representation of different aspects of architectural design using connections-based models (Mahalingam, 2003). The paradigm suggested a uniform representation of spatial layouts, circulatory systems, egress systems, structural systems and environmental control systems in architecture using three-dimensional networks or graphs. The argument was made that these three-dimensional networks or graphs reveal the architectonics underlying their composition, and by extension, could be the basis of computational frameworks. In this project, the author has simulated the behavior of a computational structure in the form of a virtual finite state machine (VFSM) that is based on works of physical architecture to see if the VFSM could be the basis of new computational tasks in architectural design such as the simulation of fire spread in a building, load transfer in structural systems, sound propagation in spatial enclosures, and heat transfer in buildings, to name a few.

2. Methodology

The way this was accomplished is as follows:

2.1. ANALYSIS

The first step was to analyze a work of Architecture (i.e., part of the built environment) so as to reveal its underlying systems, such as structural systems, circulation systems and arrangements of spaces. Three projects by the architect Frank Lloyd Wright from around the U.S.A. that were analyzed earlier by March and Steadman (1971) were selected by the author. Each of these works of Architecture had been analyzed to reveal the ‘invariant’ relationships in their arrangement of spaces. Other examples of some of systems that could have been included in the analysis are structural systems, circulation systems, egress systems, HVAC systems, plumbing systems, etc. These were not attempted in this initial implementation.

2.2. CODING

The next step was to code the spatial arrangement system as a diagram comprising nodes and links, i.e., as a graph. The spatial arrangement system uncovered in the analysis phase was coded as an adjacency graph comprising ‘nodes’ and ‘links.’

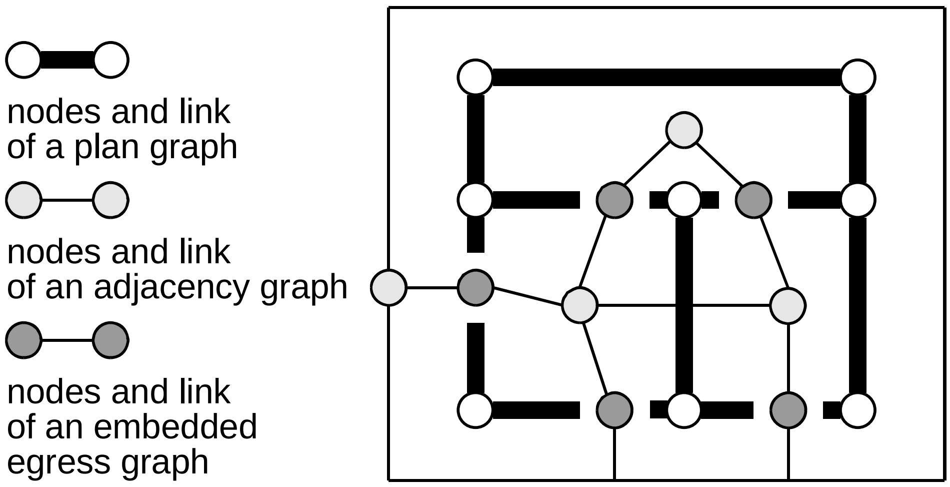


Figure 1. Encoding of an architectural plan as a graph showing how different features are embedded hierarchically.

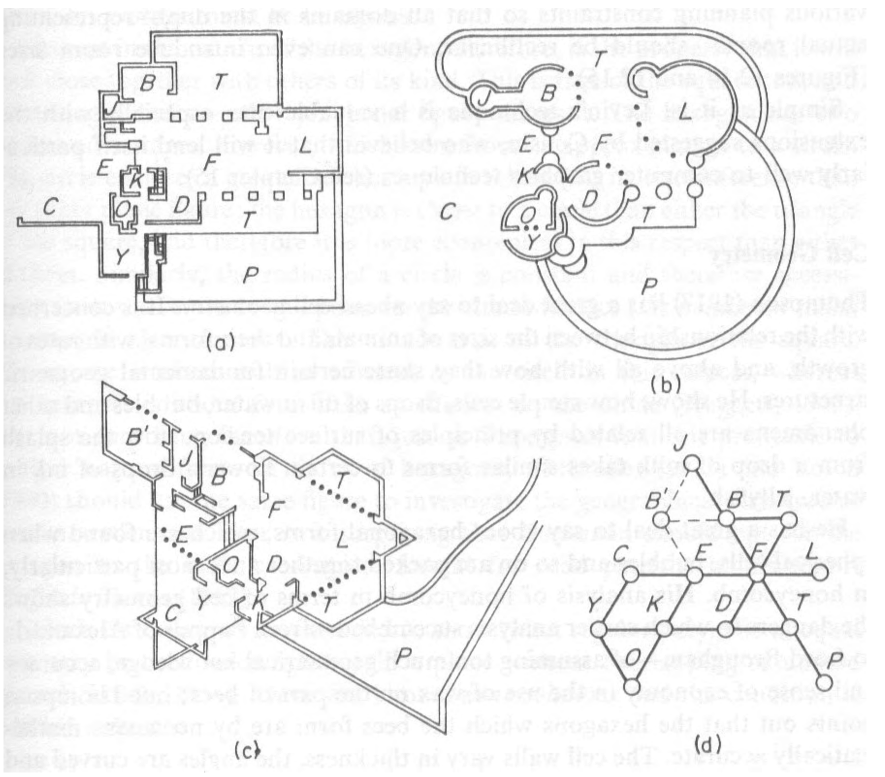


Figure 2. Three different floor plans of architectural works designed by the architect Frank Lloyd Wright showing the identical graph of space adjacencies derived from each one of them (from March and Steadman, 1971).

2.3. VFSM GENERATION

The next step was to use the graph that was uncovered in the previous step to model a virtual finite state machine (VFSM). The graph was used as a template for the generation of a VFSM using commercial software (StateWORKS for VFSM simulations).

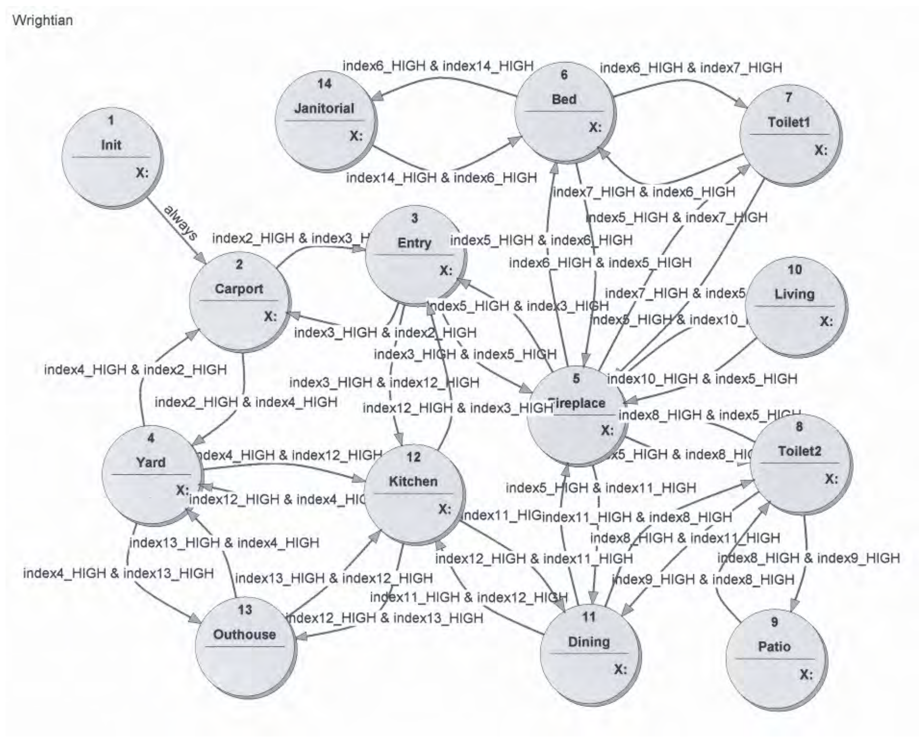

Wrightian
1 Init X:
always
2 Carport X:
3 Entry X:
4 Yard X:
5 Fireplace X:
6 Bed X:
7 Toilet1 X:
8 Toilet2 X:
9 Patio X:
10 Living X:
11 Dining X:
12 Kitchen X:
13 Outhouse
14 Janitorial X:
index6_HIGH & index14_HIGH
index14_HIGH & index6_HIGH
index6_HIGH & index7_HIGH
index7_HIGH & index6_HIGH
index5_HIGH & index7_HIGH
index5_HIGH & index6_HIGH
index6_HIGH & index5_HIGH
index2_HIGH & index3_HIGH
index3_HIGH & index2_HIGH
index5_HIGH & index3_HIGH
index3_HIGH & index5_HIGH
index4_HIGH & index2_HIGH
index2_HIGH & index4_HIGH
index3_HIGH & index12_HIGH
index12_HIGH & index3_HIGH
index10_HIGH & index5_HIGH
index8_HIGH & index5_HIGH
index5_HIGH & index8_HIGH
index4_HIGH & index12_HIGH
index12_HIGH & index4_HIGH
index5_HIGH & index11_HIGH
index11_HIGH & index8_HIGH
index8_HIGH & index11_HIGH
index12_HIGH & index11_HIGH
index11_HIGH & index12_HIGH
index13_HIGH & index4_HIGH
index4_HIGH & index13_HIGH
index13_HIGH & index12_HIGH
index12_HIGH & index13_HIGH
index9_HIGH & index8_HIGH
index8_HIGH & index9_HIGH

Figure 3. The state diagram of a virtual finite state machine (VFSM) based on the 3 architectural works by Frank Lloyd Wright that share the same adjacency graph for the spaces that they contain. The finite state machine is used to determine computationally if fire has spread from one space to another, given the occurrence of fire at the various locations.

2.1. SIMULATION

The next step was to simulate computations using the VFSM and see what computational structures could be derived from the works of Architecture. The VFSM was used to simulate the spread of fire in the buildings, a computational task in architectural design that could be mapped easily onto the VFSM.

2.1. TESTING

The last step was to test to see if the computational structures could be used to form the basis of new computer software for architectural design (i.e., the spread of fire in a building). The efficiency of the VFSM in performing the computational task attempted was demonstrated. The suitability of the VFSM for new computational tasks in traditional computation as well as other computational tasks in architectural design will be explored in the future.

3. Implementation

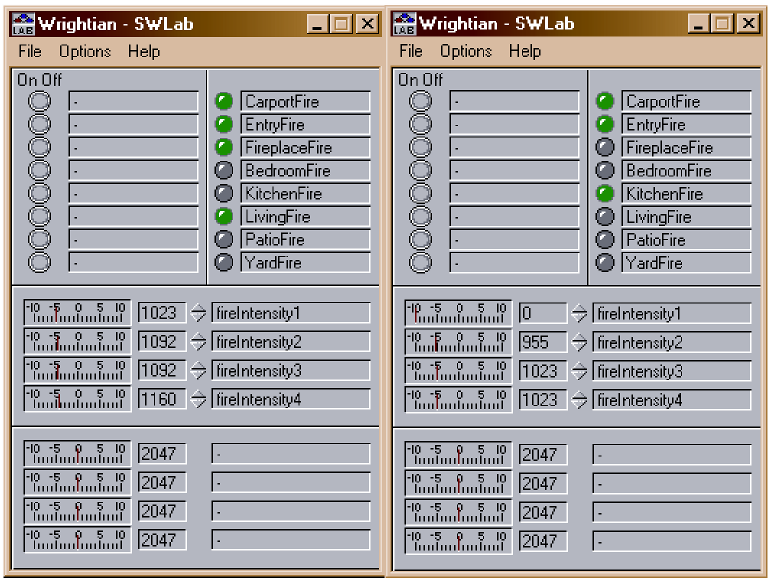
The spatial arrangement of three works of architecture by the architect Frank Lloyd Wright which were analyzed earlier and coded as adjacency graphs were used in the implementation. Incidentally all three works had the same underlying adjacency graph. The software StateWORKs (Wagner et. al., 2006) was then used to generate a virtual finite state machine (VFSM) that was based on the adjacency graph of the spatial arrangement.

A particular computational implementation was then mapped onto the VFSM. This was a computation that would determine if fire spread to a particular space given the occurrence of a fire in another space. The nodes of the graph (the spaces) were each assigned a range for a flammability value. This flammability value was modeled as a 'switchpoint' that would switch on and off based on whether the fire in that space crossed the high or low threshold value. If the intensity of a fire in that space exceeded the flammability value's high threshold then the space caught fire. Conditions were set for the fire to transmit from one space to another. This was modeled as state transition conditions in the VFSM. A system was then set up to input the intensity of a fire in each of the spaces. A simulation was then run, whereby one could input the intensity of a fire in each of the spaces using a numerical input dialog box and see if it spread to the other spaces, which was indicated in an output monitor that indicated that a fire had occurred in that space. The whole process of the spread of the fire was a computation of state transitions in the VFSM.

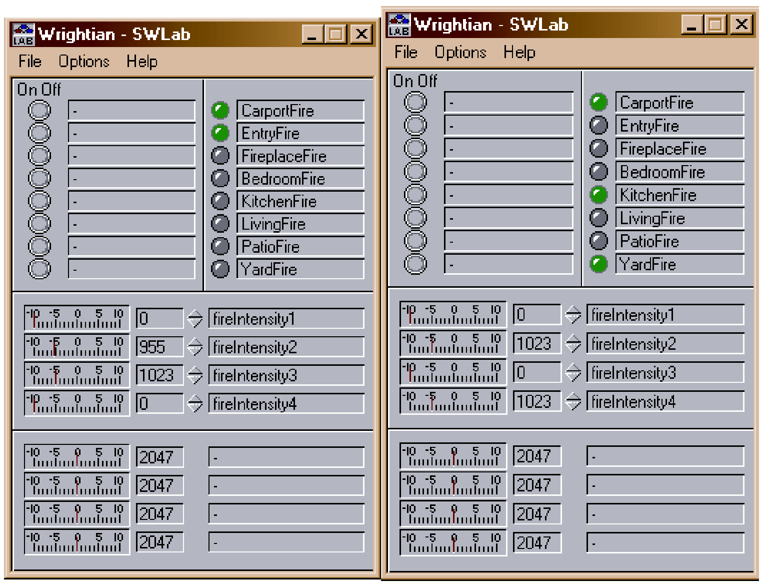
In a real world scenario, the system for the input of the intensity of the fires could be linked to a real digital input using a communication port in the computer, and the output signal that a fire had occurred could be used to activate an alarm using another communication port in the computer. This capability to link digital inputs and outputs to communication ports on the computer is inherent in the StateWORKS software system. This VFSM could effectively form the engine of a real fire alarm system in each of the buildings analyzed.

If one had to develop software for the prediction of fire spread in the architectural design by inputting flammability values for each of the spaces, starting fires of various intensities in the various spaces, and predicting where the fire would spread, then this VFSM could be used as an engine for the development of the software. The StateWORKS software system allows you to generate such software engines for runtime control systems with full control of I/O (input/output) such as WinStExec, StExec, LinuxExec and a diskless RTOS (real-time operating system) environment, which can be used for software development using other IDEs (integrated development environments). The conditional transitions from state to state in the VFSM could also be used to model systems such as Bayesian networks that are based on the VFSM. The state transition conditions could then incorporate probabilistic triggers.

Also other computational systems, such as heat transfer from space to space, could also be mapped onto the same VFSM. Instead of the ‘flow’ of fire, the ‘flow’ of heat from space to space could be computed using the same VFSM. The conditional transitions in the computational ‘flow’ from space to space could be modeled based on the heat transfer properties between the spaces.



	Left window	Right window
CarportFire	● on	● on
EntryFire	● on	● on
FireplaceFire	● on	● off
BedroomFire	● off	● off
KitchenFire	● off	● on
LivingFire	● on	● off
PatioFire	● off	● off
YardFire	● off	● off
fireIntensity1	1023	0
fireIntensity2	1092	955
fireIntensity3	1092	1023
fireIntensity4	1160	1023
-	2047	2047
-	2047	2047
-	2047	2047
-	2047	2047



-	🟩	CarportFire
-	🟩	EntryFire
-	●	FireplaceFire
-	●	BedroomFire
-	●	KitchenFire
-	●	LivingFire
-	●	PatioFire
-	●	YardFire

0	fireIntensity1
955	fireIntensity2
1023	fireIntensity3
0	fireIntensity4
2047	-
2047	-
2047	-
2047	-

-	🟩	CarportFire
-	●	EntryFire
-	●	FireplaceFire
-	●	BedroomFire
-	🟩	KitchenFire
-	●	LivingFire
-	●	PatioFire
-	🟩	YardFire

0	fireIntensity1
1023	fireIntensity2
0	fireIntensity3
1023	fireIntensity4
2047	-
2047	-
2047	-
2047	-

Figure 4. Screen shots of the VFSM runtime computation monitor in StateWORKS that monitors the Wrightian VFSM. The indicators in green show where the fires have occurred and the numerical values are the intensity of fires that have been mapped to the various spatial locations in the Wrightian houses.

4. Intellectual Merit of the Project

The intellectual merit of this project is that it makes a unique proposal to analyze and code works of Architecture in a way that enables one to discover hidden computational structures in the works of Architecture. It is hoped that the project will provide valuable insight into the architectural basis of computational structures.

During the process of architectural design, various formal structures (i.e., structures that articulate a particular form) are generated and integrated to define the design of a building. These formal structures determine the spatial layout of structural systems, circulation systems, egress systems, arrangement of spaces, HVAC systems, plumbing systems, etc. in a building. All these formal structures are integrated in the design process to create the design of a functional building. These formal structures satisfy many constraints and meet many performance criteria in different domains. As such, they are very complex design constructs. If these formal structures could be shown to be feasible computational structures as well, then the rigor and complexity of the architectural design process could be brought to bear on the design of software systems. If a particular formal structure derived from a work of Architecture is shown to be a computational structure as well, then the methodology of the architectural design process that resulted in that formal structure could be studied as a viable software design process. This will bring the whole body of design methods used in the architectural design process into the world of software design. Conversely the research process will also yield computational structures for the design of architectural entities, thereby enabling the creation of new kinds of computer-aided design systems in Architecture.

The broader impact of this research project will be to amplify the interdisciplinary relationship between Architecture and Computer Science and provide practical benefits such as the creation of new kinds of software for both traditional computational tasks and for architectural design. Though the methodology described in this project aims at discovering hidden computational structures in Architecture, it can be adapted to discover hidden computational structures in other fields such as Engineering and Biology, thereby enriching the field of computation.

5. Conclusion

The project described in this paper has successfully shown how you can take a formal structure from Architecture and convert it into a computational structure. It has also shown how this computational structure can be used as an engine to develop hardware and software systems for applications such as the monitoring of fire spread in a building. This is the proof of concept for discovering computational structures in architecture. The project still has to demonstrate that these computational structures, which are derived from works of Architecture, can be feasible computational structures for tasks in traditional computation. They hold the promise of serving as meta-computational structures for computational applications in architectural design, but have yet to be shown to enable other computational tasks such as sorting and searching, which are often considered benchmark tasks in Computer Science.

In his landmark book, Hillier presented the case that “space is the machine.” (Hillier, 1996) This book has a strong connection to this project. However, Hillier was specific in referring to his theory as a “configurational theory of architecture,” and not a “computational theory of architecture.” In a chapter devoted to the topic, he made the case for “non-discursive techniques,” that were neutral in the analysis of space and form, thereby aiming for a “universal” understanding and the development of an “internal” theory of architecture. Is Hillier’s machine a computer? If this is the case, the ‘configurations’ of architecture become viable ‘computational structures’ as well. This project reveals the intriguing possibility that this may be the case. As this project unfolds, more involved issues related to discovering computational structures in architecture are bound to emerge, which need to be thoroughly investigated.

The results of this research project are intended to be used as the foundation for an interdisciplinary Honors seminar course at our university titled, “The Architecture of Software Systems.” This course will extend this inquiry and develop it further. The Honors program at our university is based on selective admission and attracts the best and brightest students in the university who have a natural inclination for interdisciplinary studies. Courses are typically taught by a team of two or more faculty members from different disciplines. Mahalingam and a faculty member from Computer Science intend to teach the Honors course together. Their cross-disciplinary collaboration on the subject will make them effective teaching colleagues. The course will stimulate motivated students to pursue and extend research ideas in this area of inquiry further by exposing them to the state-of-the-art in this field.

Acknowledgements

The developers of the StateWORKS development tool have opened up this new avenue for research in the field of computer-aided architectural design. StateWORKS allows researchers to study computational modeling problems in architecture by building and testing tractable solutions in the form of virtual finite state machines that can be implemented in both software and hardware.

References

Hillier, B: 1996, *Space is the machine*, Cambridge University Press, Cambridge, England.

Mahalingam, G: 2005, “A Computational Model of a Sensor Network for the Optimization and Control of Acoustical Performance Criteria in Spatial Enclosures” Proceedings of CAADRIA 2005, New Delhi.

Mahalingam, G: 2003, “Representing Architectural Design Using a Connections-based Paradigm,” Proceedings of the ACADIA 2003 Conference, Indianapolis, Indiana.

Mahalingam, G: 2001, "POCHE: Polyhedral Objects Controlled by Heteromorphic Effectors," Proceedings of the CAAD Futures 2001 Conference, Eindhoven, Netherlands, July, 2001. The proceedings were also published as a book, “CAAD Futures,” by Bauke de Vries, Jos van Leeuwen and Henri Achten, Kluwer Academic Publishers, Dordrecht.

Mahalingam, G: 2000, “The Algorithmic Auditorium: Automating Auditorium Design,” Proceedings of the ACSA Technology Conference 2000, MIT, Boston, Massachusetts.

Mahalingam, G: 2000, “Computing Architectural Designs Using An Architectural Programming Language,” Proceedings of the eCAADe 2000 Conference, Weimar, Germany.

Mahalingam, G: 1999 “A Parallel Processing Model for the Analysis and Design of Rectangular Frame Structures,” Proceedings of the ACADIA 99 Conference, Snowbird, Utah, October.

Mahalingam, G: 1999, “A New Algorithm for the Simulation of Sound Propagation in Spatial Enclosures,” Proceedings of the Building Simulation ’99 Conference, Kyoto, Japan.

Mahalingam, G: 1998, “The Algorithmic Auditorium,” Proceedings of the CAADRIA 98 Conference, Osaka, Japan.

Mahalingam, G: 1997, “Representing Architectural Design Using Virtual Computers,” Proceedings of the ACADIA 97 Conference, Cincinnati, Ohio.

March, L and Steadman, P: 1971, *The geometry of environment: An introduction to spatial organization in design*, RIBA Publications Ltd.

Wagner F, R Schmuki, T Wagner and P Wolstenhilme: 2006, *Modeling Software with Finite State Machines: A Practical Approach*, Auerbach Publications, Taylor & Francis Group, New York, New York.

A CASE FOR ARCHITECTURAL COMPUTING

Computing Using Architectural Constructs

GANAPATHY MAHALINGAM
North Dakota State University, U.S.A.
Ganapathy.Mahalingam@ndsu.edu

Abstract. This paper is about the potential of architectural computing. Architectural computing is defined as computing that is done with computational structures that are based on architectural forms. An analysis of works of architecture reveals the embedded forms in the works of architecture. A uniform, connections-based representation of these architectural forms allows us to derive computational structures from them. These computational structures form the basis of architectural computing. In this paper a case is made for architectural computing, ideas are provided for how it could be done, and the benefits of architectural computing are briefly explored.

Keywords. Architectural computing: architectural programming language; intentional programming; connections-based paradigm.

1. Introduction

Researchers in the field of computer-aided architectural design have pondered the computability of design for the past 3 to 4 decades. While this inquiry may seem moot now, given that most design activities can be performed on the computer using various pieces of software, it has masked what can now be considered as a unique form of computing, architectural computing. Claude Shannon (1937), in his influential master's thesis, *A Symbolic Analysis of Relay and Switching Circuits*, literally founded modern digital computing by integrating Boolean algebra, binary arithmetic and electromechanical relays into an effective device to perform computations. What if we now recast the inherent devices of architecture as effective machines? Can we build architectural computers from them? What would these computers do and what would be their unique characteristics? What if we derive an architectural programming language from the operations of architectural design? Not too long ago, Charles Simonyi (1995) hailed the death of computer languages and the birth of intentional programming. What is the potential of architectural intentions when we consider them as effective devices? What are the kinds of 'logic gates' that we could derive from architectural constructs? Besides opening up the world of 'architectural computing,’ this inquiry would make us reconsider the world of architecture with a renewed rigor. It is time now to make the case for architectural computing. This paper is an attempt to do that effectively.

2. Architectural Computing

What is architectural computing? Architectural computing is computing that is done with a computational structure that has as its basis an architectural form. Architectural forms are embodied in works of architecture. They are essentially intrinsic. They include the forms of the building envelope, the forms of the structural system, the forms of the mechanical and plumbing systems, the forms of the circulation system, the forms of the electrical system, the forms of the life safety and communication systems, etc. These architectural forms are manifest in the finished works of architecture. They can be derived from the finished works of architecture by careful analysis. These are the manifest forms of architecture.

However, the process of creating a work of architecture has embedded in it various inherent devices as well. These are not immediately available from a cursory visual analysis. These include datums, proportional systems, ordering diagrams, etc. These inherent devices can also be represented in such a way that they can become the basis of computational devices. The challenge lies in the creative mapping of these inherent devices into computational structures.

In recent research a case has been made for the uniform representation of architecture (i.e., architectural forms) using a connections-based paradigm (Mahalingam, 2003). A case has also been made to derive computational structures from these connections-based representations of architecture (Mahalingam, 2007). Earlier a case was made for an architectural programming language (Mahalingam, 2000). These three approaches can now be integrated into a case to be made for architectural computing.

Why architectural computing? Computer scientists often talk about computer architectures, which refer to the organization of computational devices. Though the term used is architecture, these computational devices seldom approach the complexity of works of architecture in the built environment, viz. buildings. Building designs are the result of some of the most exacting neuronal processing in the brains of designers. The synthesis of building designs represents the complex structuring of our neuronal systems. It may be said that complex works of architecture reveal human neuronal underpinnings more accurately than any other cultural artifact that humans produce. Architectural computing is proposal to tap this neuronal richness that is manifest as complex architectural constructs. The first step in this process is to see if, at the heart of architectural creation, there is a programming language.

3. An Architectural Programming Language

This section of the paper is adopted from an earlier paper on the topic (Mahalingam, 2000). It is absolutely necessary to integrate it in this paper to make the case for architectural computing.

The potential success in developing a programming language for architectural design depends on a careful mapping of the fundamental operations in the creation of architectural designs onto a set of computable operations. A characterization of architectural design at a fundamental level is needed before a programming language can be defined to enable the creation of architectural designs. Architecture has been defined as the art and science of designing buildings and supervising their construction. The creation of a work of architecture is the result of a complex interaction of diverse processes. However, the complexity in the creation of an architectural design belies a set of simple, fundamental operations.

A programming language is defined by its syntax and semantics. The syntax of the language describes the rules for creating structures (programs) using the language, and the semantics of the language reveals the meaning of valid structures (programs) that can be created with the language. Of these, the syntax is formally represented. Examples of formal description systems for the syntax of a programming language are the Backus-Naur notation and syntax diagrams.

To create a programming language for architectural design, one has to define the starting symbol, terminal symbols, non-terminal symbols and production rules for the creation of architectural designs. This may seem a daunting task, but, if we realize that the fundamental entities in architecture consist of form and space, solids and voids, the definition of a language for architectural design becomes viable.

3.1 THE DEFINITION OF AN ARCHITECTURAL PROGRAMMING LANGUAGE

This section presents the definition of an architectural programming language, complete with the Backus-Naur form (BNF) for the language. The purpose of developing this language is to provide a tool to write programs that generate architectural designs when executed. A complete syntactical description of the language including its starting symbol, its non-terminal symbols, its terminal symbols, and its set of production rules is provided.

A complete syntactical description of a language is called a grammar. A grammar can be considered a tuple of the following elements:

Starting symbol (S)

Terminal symbols (T)

Non-terminal symbols (N)

Production rules (P)

The notation for a grammar is thus: G (S, T, N, P)

A language (L) based on a grammar is defined thus: L (G) = L (S, T, N, P)

The task of creating a programming language for architectural design starts with the definition of a grammar for the creation of architectural designs. Using the 4-tuple form for the definition of a grammar, G (S, T, N, P), architectural design can be mapped thus:

Starting symbol (S): Architectural form (f)

Terminal symbols (T): Solid polyhedron (p_s), Void polyhedron (p_v), Union (U), Difference (\)

Non-terminal symbols (N): Architectural form (f), architectural space (s)

Production rules (P):

$f \rightarrow p_s \mid f \cup f \mid f \setminus s$

$s \rightarrow p_v \mid s \cup s$

The union operation (U) has precedence over the difference operation (\) in the production rules. The vocabulary (V) of the grammar or language is defined as N U T, that is, the union of the non-terminal and terminal symbols. The use of the symbol * after V, N or T indicates all possible strings over the sets of V, N and T.

These production rules defined give rise to other production rules of the form:

$f \rightarrow p_s \cup p_s$

This production rule allows an architectural form to be created by unioning a solid polyhedron with another solid polyhedron.

$f \rightarrow p_s \cup f$

This production rule allows an architectural form to be created by unioning a solid polyhedron with another architectural form.

$f \rightarrow p_s \setminus p_v$

This production rule allows an architectural form to be created by differencing a void polyhedron from a solid polyhedron.

$s \rightarrow p_v \cup p_v$

This production rule allows an architectural space to be created by unioning a void polyhedron with another void polyhedron.

$s \rightarrow p_v \cup s$

This production rule allows an architectural space to be created by unioning a void polyhedron with another architectural space.

If you visualize the creation of an architectural design, an architect starts with an existing architectural form, the site of the design. The architect then synthesizes a new form by creating a solid polyhedron, combining solid polyhedra (material) or removing void polyhedra (empty space) from the solid polyhedra (material). The production rules defined to create architectural forms are both recursive and non-recursive. Since there are an infinite number of solid and void polyhedra, this grammar does not preclude any architectural form.

In this programming language, only the Boolean operators of union and difference are used. Now can we visualize an architectural design operation that creates, in essence, a different ‘logic gate’?

The grammar presented above is context-free like most programming languages. The actual grammar to create specific types of architectural forms will be a refined version of this grammar. This grammar captures the essence of a real grammar that creates an architectural form. Since polyhedra are themselves complex entities, a nested grammar can be defined to generate polyhedra. This series of nested grammars can then be used to develop a comprehensive programming language for architectural design.

3.2 THE POTENTIAL OF AN ARCHITECTURAL PROGRAMMING LANGUAGE

Kalay (1989) calls computer models of real-world phenomena "languages of representation." What if this language of representation is a programming language? Symbols sets used in computer programming languages include the binary set (1,0) or the number set (1,2,3,4,5,6,7,8,9,0) or the English alphabet set (a,b,c,d…z). Such sets allow for programs to be written in an alphanumeric language. The traditional language of architectural design is graphical. Therefore, a programming language for architectural design should probably use graphical symbols instead of alphanumeric symbols. This would make an architectural programming language a visual programming language. What if the symbol sets in an architectural programming language are graphical? Can one then draw a program instead of writing one? The equivalent of a sentence in an alphanumeric programming language would be a drawing in the visual programming language. What are the problems or benefits related to checking the validity of a program if it is drawn using graphic symbols? Actually, the problems related to checking the validity of a program written in a visual programming language should be no different than syntax checking in an alphanumeric programming language, if the graphic elements directly correspond to alphanumeric elements.

In the grammar for an architectural programming language presented in this paper, if the alphanumeric symbols are replaced with graphics representing the polyhedra, then the string of alphanumeric symbols generated by the production rules has a graphical equivalent. The architectural programming language can generate different strings based on the production rules. These strings can then be converted into graphics by substitution. Each sentence in the language will then become a spatial composition. When the substitution is made, there may be invalid forms created by some of the production rules. This is because the alphanumeric symbols are not spatial. For example a void polyhedron that is larger than a solid polyhedron cannot be differenced from it. Similarly, two solid polyhedra that do not overlap cannot be unioned to create a single architectural form. A mechanism is needed for checking spatial parameters of the polyhedra when implementing the production rules.

Drawing an architectural design may not be essentially more complex than programming an architectural design except for the visual immediacy of the drawing and the unstructured (or very complexly structured, depending on your viewpoint) nature of the drawing process. If graphical symbols are used in the architectural programming language, then programming an architectural design can become another form of drawing, a shorthand graphical notation of the design that reveals its full visual form when the program is executed. Even symbols for operators in the architectural programming language can be given graphical equivalents. *A drawing will then be a computer program.* This will be possible if the sequence of elements and operations used to create the drawing is accessible in order to map it onto a program. A finished drawing on paper using traditional media does not have a record of the sequence of graphic elements and operations used to create it, but a computer-based drawing does! Computer-based drawings can then provide a computational medium for the generation of architectural designs in a completely different sense.

With a well-defined architectural programming language, architectural designs can be generated by executing programs written as you would with a general-purpose programming language like Smalltalk. Programs can then be written (drawn?) to generate programs that generate architectural forms. This can lead to a powerful form of automation in the creation of architectural designs.

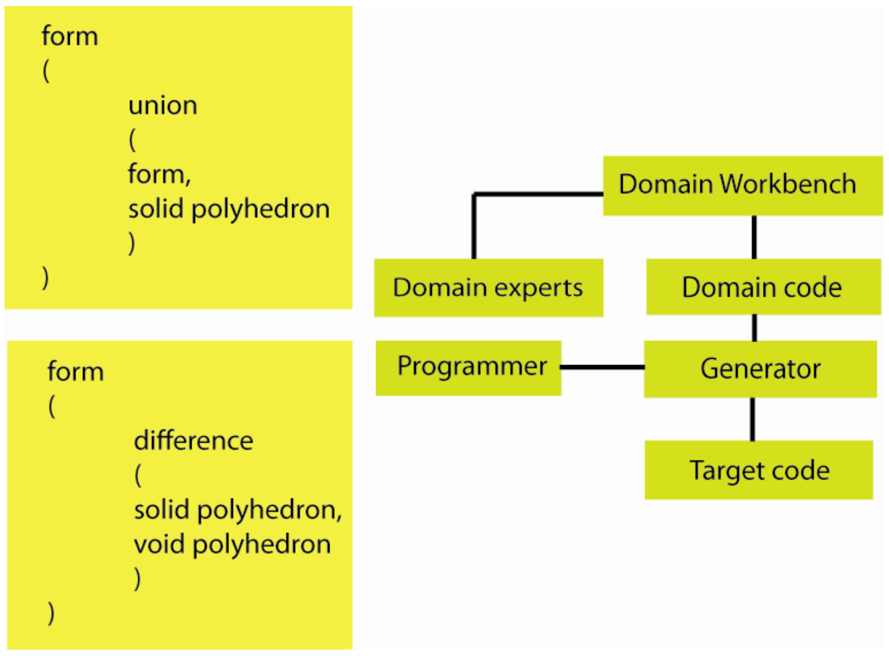
4. Intentional Programming

Charles Simonyi, who used to be one of the chief software architects at Microsoft, has been working on what he calls Intentional Software. According to his team, Intentional Software simplifies software creation by separating the software contents in terms of their various domains from the implementation of the software and by enabling automatic regeneration of the software as the contents change (Simonyi, Christerson and Clifford, 2006). In traditional software development, programmers had to take tasks that were to be completed in the domain of an application, for example, the creation of a cuboid in 3D modeling software, and represent the process in a form that the computer could understand, that is, using a general purpose programming language. The program was written to facilitate execution on the computer and not to articulate the task being performed. This disconnection between a machine executable representation and an 'intentional' representation in performing a task is the gap that is being closed by Intentional Software.

In the implementation of Intentional Software, domain experts can work in parallel with programmers in their respective areas of expertise; and the repeated intermingling can be automated. Intentional Software is supported by a Domain Workbench tool where multiple domains can be defined, created, edited, transformed and integrated during software creation (Simonyi, Christerson and Clifford, 2006).

Domain experts first define domain schema, where terms of the domain code are defined. Domain experts then define the domain code using the Domain Workbench tool and a domain specific language. This domain code takes the form an intentional tree in its parse structure. The domain code can also be converted to other forms of notation such as a finite state machine diagram (see Figure 2). A generator then processes the domain code to generate target code that is executed on the computer. The target code is the software program or application that a user needs to perform a particular task. These domain schemas and codes are defined by the domain experts, for example, the architectural programming language described in this paper would be a high-level 'intentional' domain defined by a domain expert, in this case, the architect.

Using this domain definition many programs for the creation of various architectural designs could be generated (see Figure 1 for an overview of the Intentional Software system where the domain code for some of the production rules in the architectural programming language presented in this paper is shown in yellow). This is not always easy. In complex systems, the domain vocabulary, domain relationships and domain rules may not lend themselves to be easily mapped onto a programming language to generate the target code. However, this is not as difficult in the synthesis of architectural forms as the architectural programming language shows.



```
form
(
        union
        (
        form,
        solid polyhedron
        )
)
```

```
form
(
        difference
        (
        solid polyhedron,
        void polyhedron
        )
)
```

Figure 1. Overview of the Intentional Software system

Other key features of Intentional Software include a uniform representation of multiple interrelated domains, the ability to project the domains in multiple editable notations, and simple access for a program generator (Simonyi, Christerson and Clifford, 2006). This rich environment seeks to exploit domain schema and codes from innumerable sources for diverse applications. These domain schemas and codes serve as engines for the software development process. A critical role is played by the generator in this process. The domain schemas and codes only specify the data structure; the behavior of the data in the target code is generated by the generator using a process that translates the domain code into target code.

The world of architecture is rich in domain schema and domain codes. Hitherto, the world of architecture has not been seen as a valuable source of domain schema and domain codes for software design. With the implementation of Intentional Software, the opportunity has arisen for the use of architectural schema and architectural codes in the process of software design. Consider a structural design schema in architecture that can be used to create software for the design of the structural elements involved (Mahalingam, 1999). Consider a spatial layout schema, where the spatial layout and the interconnection of the spaces is the engine for a fire spread and control software for the building (Mahalingam, 2007) (see Figure 2). Consider a spatial design synthesis schema in the manner of a master architect, for example, software that can be used to design in the manner of a Palladio or a Wright (Hersey and Freedman, 1992). Consider a circulation system schema that can be used to create a program that automatically generates spatial layouts for buildings. The possibilities are endless. Where this research avenue can be taken is as limitless as the world of intentional forms.

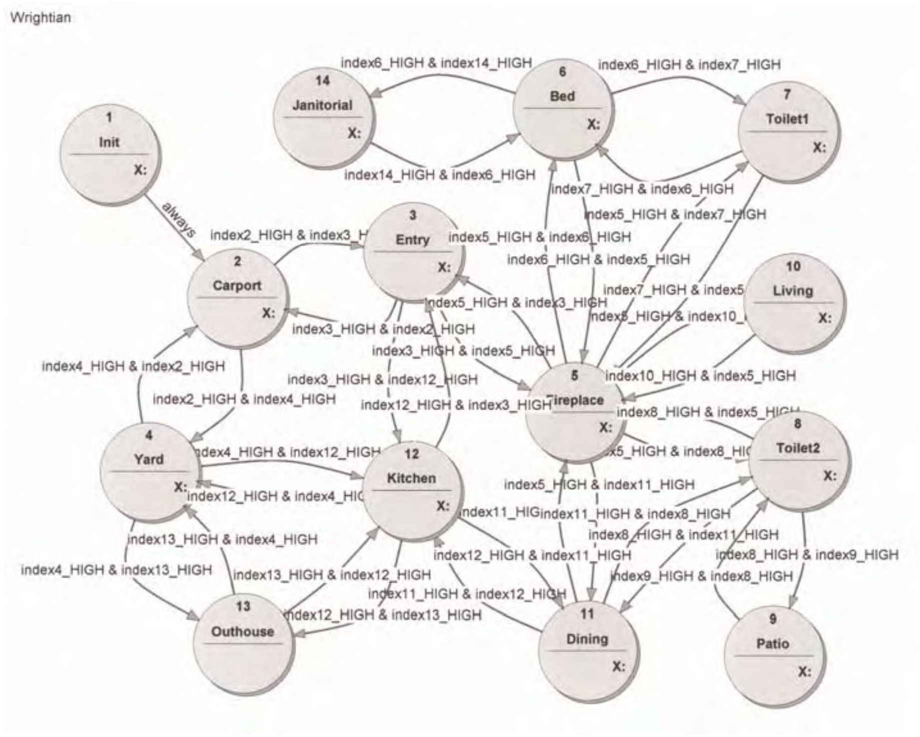

Wrightian
1 Init X:
2 Carport X:
3 Entry X:
4 Yard X:
5 Fireplace X:
6 Bed X:
7 Toilet1 X:
8 Toilet2 X:
9 Patio X:
10 Living X:
11 Dining X:
12 Kitchen X:
13 Outhouse
14 Janitorial X:
always
index6_HIGH & index14_HIGH
index14_HIGH & index6_HIGH
index6_HIGH & index7_HIGH
index7_HIGH & index6_HIGH
index5_HIGH & index7_HIGH
index2_HIGH & index3_H
index5_HIGH & index6_HIGH
index6_HIGH & index5_HIGH
index7_HIGH & index5
index5_HIGH & index3_HIGH
index5_HIGH & index10_
index3_HIGH & index2_HIGH
index3_HIGH & index5_HIGH
index4_HIGH & index2_HIGH
index10_HIGH & index5_HIGH
index3_HIGH & index12_HIGH
index2_HIGH & index4_HIGH
index12_HIGH & index3_HIGH
index8_HIGH & index5_HIGH
index4_HIGH & index12_HIGH
index5_HIGH & index8_HIG
index12_HIGH & index4_HIG
index5_HIGH & index11_HIGH
index11_HIG
index11_HIGH & index8_HIGH
index13_HIGH & index4_HIGH
index8_HIGH & index11_HIGH
index12_HIGH & index11_HIGH
index8_HIGH & index9_HIGH
index4_HIGH & index13_HIGH
index13_HIGH & index12_HIGH
index9_HIGH & index8_HIGH
index11_HIGH & index12_HIGH
index12_HIGH & index13_HIGH

Figure 2. Domain code in a finite state machine notation for a system to predict fire spread in a Wrightian style house

5. Conclusion

Architectural computing is a new frontier. There is enough structure in the process of creating architectural designs and in architectural products to allow us to derive architectural programming languages and other complex computational structures from them. These programming languages and computational structures will initially inform the process of architectural design and expand its potential. They would then migrate to other disciplines and engage worlds such as engineering and biology. Architecture is a universal phenomenon. Form is its central ingredient. Architectural computing mobilizes computing with architectural forms. The future is wide open.

References

Hersey, G. and R. Freedman, *Possible Palladian Villas*, MIT Press, Cambridge, Massachusetts, 1992.

Kalay, Y. E. *Modeling Objects and Environments.* John Wiley & Sons, New York, New York, 1989.

Mahalingam, G. *Discovering Computational Structures in Architecture*, CAAD Futures 2007 Conference, Sydney, Australia, July 2007.

Mahalingam, G. *Representing Architectural Design Using a Connections-based Paradigm*, ACADIA 2003 Conference, Indianapolis, Indiana, October, 2003.

Mahalingam, G. *Computing Architectural Designs Using An Architectural Programming Language*, eCAADe 2000 Conference, Weimar, Germany, June 2000.

Mahalingam, G. *A Parallel Processing Model for the Analysis and Design of Rectangular Frame Structures*, ACADIA 99 Conference, Snowbird, Utah, October 1999.

Shannon, C. *A Symbolic Analysis of Relay and Switching Circuits,* Master's Thesis, Massachusetts Institute of Technology, 1937.

Simonyi, C., M. Christerson and S. Clifford. *Intentional Software. OOPSLA'06* October 22-26, Portland, Oregon, USA, 2006.

Simonyi, C. *The Death of Computer Languages, The Birth of Intentional Programming,* Microsoft Technical Report, MSR-TR-95-52, Microsoft Research, 1995.

Teufel, B. *Organization of Programming Languages.* Springer-Verlag, New York, 1991.

In Formation…

The central variable and object of contention for value in Architecture is form. Architects have long relied on innumerous methods from various disciplines to derive architectural forms. This has given Architecture its incredible richness and diversity. It has also made the central process of Architecture – the creation of architectural form – an open-ended and a politically contentious process. This obsession with architectural form has been so strong that architects have often ignored the inhabitation of the forms and their sustainability as the real reasons for creating them in the first place. The profusion of architectural forms being generated currently through the use of digital media belies the persistent anxiety of architects in validating what they are creating. Reverting to divine inspiration as an explanation is the easy recourse, but validating what we create is an unavoidable human responsibility.

There is a relatively new area of focus in human inquiry that is emerging, which has the potential to give architects the foothold they have longed for in validating the creation of architectural forms. Interestingly this area of inquiry links architecture to the last component in the *quadrivium* of professions made up of architecture, engineering, law and medicine. Architects have engaged engineering and law in the creation of the built environment and this engagement has served them well. The moment has come for architects to engage the fourth profession in the *quadrivium* – medicine, and with it, the world of biology. This engagement promises to resolve the persistent anxiety in the validation of architectural form through an area of inquiry called developmental biology.

“Developmental biology is the study of the process by which organisms grow and develop. Modern developmental biology studies the genetic control of cell growth, differentiation and "morphogenesis," which is the process that gives rise to tissues, organs and anatomy. The study of morphogenesis involves an attempt to understand the processes that control the organized spatial distribution of cells that arises during the embryonic development of an organism and which give rise to the characteristic forms of tissues, organs and overall body anatomy.” (Wikipedia)

This last aspect of developmental biology, morphogenesis, the evolution of form, as the name implies, may embody the architect’s salvation.

In the early part of the 20th century D’Arcy Wentworth Thompson wrote the influential book, *On Growth and Form*, where he explored evolutionary frameworks for the growth of various kinds of biological forms. His deliberations were picked up decades later in a telling manner by Alan Turing, a pioneer in the field of computation as we have come to know it today (Saunders, 1992). Turing used mathematical models based on reaction-diffusion equations to suggest a methodology for pattern formation in biology, and in doing so may have also founded the contemporary study of pattern formation in nature. He was specifically interested in how the Fibonacci series was manifest in the growth and development of plants. Unfortunately Turing passed away before he could build significantly on Thompson’s work. Digital media enthusiasts in architecture have recently rediscovered the work of Thompson, no doubt intrigued by Turing’s connection to his work, and have sought inspiration from it to drive the creation of architectural form using digital media. This heady mixture of the early stirrings of developmental biology and contemporary digital mediation has created the potential for unprecedented knowledge in the creation of forms at the scale and complexity of nature. Developmental biologists are studying morphogenesis both *in vivo* and *in silico*. They have integrated computational tools into a field called systems biology and are now able to model aspects of morphogenesis computationally (Alon, 2006). Here we are now, at the crossroads of biology, computation, and dare we say, architecture, that has begun to shed a new light on information seen as, in formation, that is, in the particular state of being formed.

Why is this important? It may be urgently so, because understanding how genomes evolve, and morphogens (form generators) enable the creation of tissues and organs, will give architects a far more sophisticated set of form-making tools than the Cartesian grid and Euclidean geometry to take on the complex imperative of sustainability of the built environment. Our very survival may be dependent upon being able to understand and master these form-making techniques.

"Morphogens are substances governing the pattern of tissue development and, in particular, the positions of the various specialized cell types within a tissue. Morphogens spread from a localized source and form a concentration gradient across a developing tissue." (Wikipedia) This concentration gradient in turn drives the differentiation in the developing tissue required for the emergence of form. The processes of diffusion, activation and deactivation in cell structures, collectively called gastrulation, becomes the dynamic environment in which form is created. Understanding this dynamic process will enable architects to eventually attain a facility in form-making similar to ones found in nature. "Well-known morphogens include: Decapentaplegic / Transforming growth factor beta, Hedgehog / Sonic Hedgehog, Wingless / Wnt, Epidermal growth factor, and Fibroblast growth factor." (Wikipedia) Many of these morphogens were identified through the study of the fruit fly embryo. These morphogens are defined conceptually, not chemically, based on how they create or influence different forms. For example, the Sonic Hedgehog morphogen influences the creation of digits (fingers and toes) in limbs, the midline structures in the brain and the spinal cord. How these morphogens actually work in specificity is still being charted by developmental biologists.

The really intriguing set of questions is: What lessons do morphogens have for architects? What would be the equivalent of morphogens in the creation of various kinds of architectural form? For example, is there an Auditorium morphogen in architecture? What would it be, and how would it function?

The case has been made for Acoustic Sculpting and the Algorithmic Auditorium (Mahalingam, 1998). Acoustic Sculpting, identified by Professor Mahalingam (in the sense of giving it identity), is the process of creation of architectural form, specifically the spatial form of an auditorium, based on acoustical parameters such as reverberation time, the time delay gap of sound reflections and inter-aural cross correlation. Using geometric, mathematical and statistical functions, it is a method to generate architectural form that performs according to optimized criteria, acoustical criteria in this case. Through Acoustic Sculpting, sound becomes a form-giver (a morphogen?) for Architecture. The process of

Acoustical Sculpting was used in the Algorithmic Auditorium project. The Algorithmic Auditorium is the project in which an algorithm was created for the preliminary spatial design (the design of the three-dimensional spatial envelope) of a proscenium-type auditorium based on acoustical, programmatic, performance and visual criteria, and implemented as a functioning piece of software. (Mahalingam, 1995). These two concepts were a precursor to a new way of looking at architectural form-making. They also anticipated the role of morphogens in architecture by almost a decade. Evolution has now brought us to a point where practical tools such as Bentley's Generative Components (released commercially in 2007) are now making such considerations of architectural form-making viable and accessible.

Where is this leading us as architects? Is the Auditorium morphogen an evolutionary extension of the Algorithmic Auditorium? Does it point to a convergence of the fields of biology, computation and architecture? Will current inquiry resolve these questions? In a paper presented at a conference in 2003, Professor Mahalingam suggested that a uniform connections-based representation of all aspects of architecture could be used to map the architectural genome and understand the myriad architectural forms that exist in the world. (Mahalingam, 2003) What if we could map both the architectural genome and, at the same time, identify all the morphogens that differentiate architecture? Could the study of form-making in architecture run parallel to the field of study of the developmental biologists? Could architects' understanding of form-making inform the work of developmental biologists? Could the work of architects be profoundly significant, beyond the world of inhabitation and sensory experience? Could Architecture enhance Medicine? This is what is decidedly in formation…

References:

- Alon, U. (2006). *An Introduction to Systems Biology: Design Principles of Biological Circuits.* Chapman & Hall.
- Mahalingam, G. (1995). *The Application of Object-Oriented Computing in the Development of Design Systems for Auditoria.* Ph.D. Dissertation. University of Florida, Gainesville, Florida.
- Mahalingam, G. (1998). The Algorithmic Auditorium. Proceedings of The Third Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA '98), Osaka, Japan, pp. 143-152.
- Mahalingam, G. (2003). Representing Architectural Design Using a Connections-based Paradigm. Proceedings of Annual Conference of the Association for Computer-Aided Design In Architecture (ACADIA 22), Indianapolis, Indiana, pp. 269-277.
- Saunders, P.T. (1992). Editor. *Collected Works of A. M. Turing: Morphogenesis.* North Holland.
- Thompson, D. W. (1992). *On Growth and Form: The Complete Revised Edition.* Dover Publications.