SELF-ORGANIZING MAP AND CELLULAR AUTOMATA COMBINED TECHNIQUE FOR ADVANCED MESH GENERATION IN URBAN AND ARCHITECTURAL DESIGN

Álvaro Castro Castilla, Nuria Gómez Blas

Abstract: This paper presents a technique for building complex and adaptive meshes for urban and architectural design. The combination of a self-organizing map and cellular automata algorithms stands as a method for generating meshes otherwise static. This intends to be an auxiliary tool for the architect or the urban planner, improving control over large amounts of spatial information. The traditional grid employed as design aid is improved to become more general and flexible.

Keywords: self-organizing map, cellular automata, CAD, CAAD, architectural computation.

ACM Classification Keywords: J.6. Computer Applications - Computer-aided design

Introduction

Architectural and urban design brings up a kind of problems inherently different from those found in engineering or science. In contrast with generalized belief, specially from the engineering and scientific domains, the main difficulties in the development of consistent CAD tools are more related to simultaneity of constrains and complexity of their parametrization, rather than in the topics of creativity and inspiration. Currently these tools are aimed to do little more than mimic the abilities of pen and paper, with the addition of basic copy-related commands.

Different methods for coping with that complexity are to be investigated from artificial intelligence and other research areas such as L-systems or fractals. In this paper a technique for supporting the designer's work in the early stages of the process is proposed through the combination of two models: self-organizing maps and cellular automata .

A self-organizing map (SOM) is a new, effective software tool for the visualization of high-dimensional data. In its basic form it produces a similarity graph of input data. It converts the nonlinear statistical relationships between high-dimensional data into simple geometric relationships of their image points on a low-dimensional display, usually a regular two-dimensional grid of nodes [Kohonen, 2001].

Cellular automata (CA) are mathematical and computational models for systems in which the global behavior is reached through the collaboration of multiple simple parts.

Motivations for cellular automata study can be found in different fields as vehicles for studying pattern formation and complexity. CA can be treated as abstract discrete dynamical systems embodying intrinsically interesting and potentially novel behavioral features [Ilachinski, 2001].

Problem description

Urban planning deals mainly with land units and classification, taking decisions at the large scale that affect each one smaller portion. The basic atom for urban design are those pieces of ground that introduce an extensive list of parameters, such as area, use, cost per measuring unit, etc [Colonna, Di Stefano, Lombardo, Papini, Rabino, 1998]. A variable amount of them, acting together and usually seen statistically, are used by the designer to

constrain and design others. In fact, those layers of information usually come one after the other but depending on upcoming ones. That implies cyclic processes and introduces the need for induction thinking.

Land units manifest their geometrical phenotype: the land divisions map and the set of rules that should govern its physical execution. The former is commonly worked out using a mesh for spatial structuration. Of course, there several other techniques, even unique ones depending on the project. This is a matter of process decision, however we can't -and should not- avoid some sort of deviation at most stages of the urban project. We will aim for the most general approach, in this case, mesh-based design.

Meshes introduce two issues; these are the limitations we want to solve with this proposal:

Fixed topology: basic design meshes have fixed topology, as they aim for simplicity and ease of use for the designer. In contrast with the usability, the information this meshes use to be built from is inherently dynamical in its topology, as one consequence of complexity in real-life originated data. In other words, spatial relationships are not kept over time in urban reality. So our tool should be able to assume that feature and develop the framework to process such variations.

Shape constrains: refers to the land division methods and geometrical structures that conform the actual plan. Historical and cultural diversity show us several different ways to accomplish this goal. However, any taken decision at this aspect would limit the method's generality. This suggests that building a mesh where basic units are not limited by enclosures, thus conforming shapes, but turning its basic units into something shapeless would be desirable. This is quite obvious but not a common design approach, as it implies a higher level of abstraction and a bigger leap towards materialization of a design idea.

The objective is to build a new mesh that improves on the basis of these two limitations for the specific task of helping the urban planner. After that we should be able to use it in order to evaluate its first results.

Orthogonal grid approaches

Urban behavior modeling based on cellular automata has been widely studied in recent years. Many of those models are based on the typical orthogonal mesh, the archetypical grid. The CA orthogonal lattice is commonly used as the most neutral -thus general- geometrical base. It turns variables which are spatially extensive into their density-intensity equivalents and this immediately means that comparisons can be made [Batty, Xie, Sun, 1999]. The geometric configuration of the spatial units used to represent the spatial data can have a profound effect, explaining why using spatial systems which neutralize the effect of configuration remove any bias caused by convoluted or distorting geometries.

The regular grid has other advantages, as the additional ability to work in layers without additional efforts to make different ones fit into the same system, and the significant work which proves its success even with highly refined CA rules involving cultural and human factors [Portugali, Benenson, 1997].

For such reasons current models are centering their interest in *orthogonal isotropic shape-constrained meshes* (the regular grid); they are essentially analytic. Nevertheless, most of them function with a certain degree of deviation from classic CA [Zhongwei, 2003], redefining cell space, neighborhood, lattice and time concepts, which is necessary for some degree of flexibility.

Proposed technique

We are going to describe a technique for building meshes as a generalization of these grids, more flexible and adaptable entities where orthogonality, isotropy, and constant topology are specific cases. Fur such purpose, a combination of Kohonen's self-organizing map and a general CA algorithms set lead us to satisfactory results.

1. Motivations for SOM and CA combined approach

We detected interesting properties in both models that acting indepently solve different aspects of the grid (see Table 1). Basic SOM algorithms work reconfiguring their neurons' weight in a competitive basis, resulting in a problem-specific distribution while preserving topology of the map. However, CA algorithms work efficiently calculating relationships among cells, with just an initial configuration and no needed input data. On the other hand, classic CA rely on fixed-topology lattices, being highly suitable for massively parallel calculations, although they are not limited to be rectangular and uniform [Abdalla, Setoodeh, Gürdal, 2006].

Self-Organizing Maps	Cellular Automata
preserves topology of the map	relies on topologically static lattice
works on input data	works from seed and sets of rules
problem-specific adaptation (outer generation)	cell relationships (inner generation)

Table 1. SOM and CA for design.

These features suggest the possibility of dividing the design problem into two conceptually complementary parts:

- *Outer generation*: refers to the ability of the design system to assimilate external information and reconfigure itself depending on the input.
- *Inner generation*: concerning the inner properties of the system, not directly depending on the external constrains of the design problem. Although it can introduce external variables to the system, the only potential effect on the global qualities could be emergent, thus unpredictable.

2. Kohonen's self-organizing map

SOM are Artificial Neural Networks that carry out their adjustment process through the unsupervised learning paradigm, and the input data are called unlabeled data. As a simplified definition, we can say that, in a topology-preserving map, units located physically next to each other will respond to classes of input vectors that are likewise next to each other. Although it is easy to visualize this in two-dimensional array, it is not so easy in a high-dimensional space. N-dimensional input vectors are projected down on the two-dimensional map in a way that maintains the natural order of the input vectors. This dimensional reduction could allow us to visualize easily relationships among the data that otherwise might go unnoticed [Freeman, Skapura, 1991].

A basic SOM is able to reconfigure its weights distribution depending on the training data. Weight vectors correspond to the classes the algorithm is finding in the training set. If we graph the weights vectors together with the input vectors we get a similarities diagram where the neuron units are shown as representatives of a data class from that input. We can describe the learning process by the equation

$$w_i = \langle(t)(x - w_i) U (y_i)$$

Where w_i is the weight vector of the *i* unit and *x* is the input vector. The function $U(y_i)$ is zero unless $y_i > 0$ in which case $U(y_i) = 1$ so only active units will learn. The factor α(t) is a function of time to allow its modification as the learning process progresses.

As a competitive structure a winning unit is determined for each input vector based on the similarity between the input and the weight vectors. The winning unit is evaluated by

$$|| x - w_c || = \min_i \{ || x - w_i || \}$$

Finally the updated weights are those of the winning unit plus a defined neighborhood with certain decay. The cycle is repeated, adjusting the neurons' weight until a number of iterations is reached or a fixed condition is satisfied. The neurons are gradually learning as they win for a specific class of input sample. New sample vectors will fall into a previous class or excite a neuron that wasn't previously excited, tending to be a representative of a new class. Hence, the algorithm -in its simple form- is often used as a generic classifier.

The output, seen as a *weights map*, is generally used as a representation of the classes. If you graph the input vectors on top of it, you would see the relationships easily.

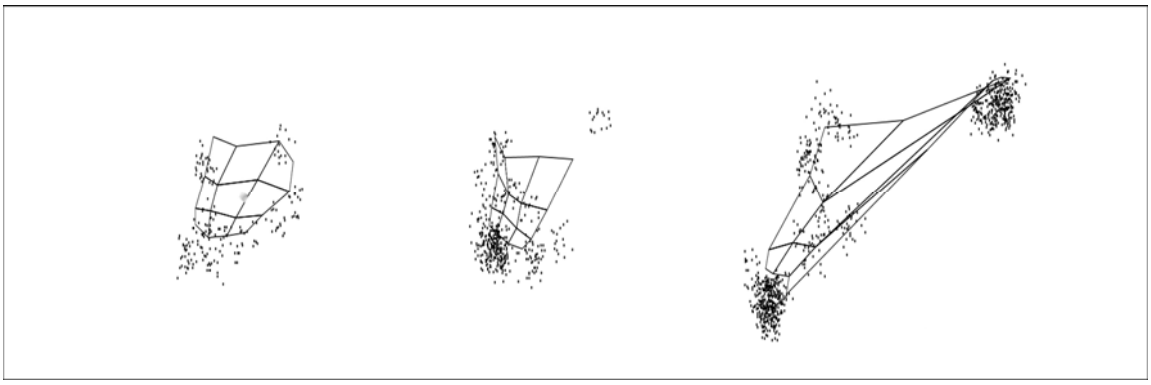

Figure 1. Graphing input vectors (point clouds) and weights (nodes of the mesh).

Nevertheless, our interest is mainly focused on that output graph. The principle we are going to follow is very simple: If we use spatial data as input, we will get spatial data output. Thus *the output graph of spatial information is spatial information itself*; and so the graph turns into a visualized mesh of points.

As information required by the designer is visual, the evaluation of the results could be done entirely with this mesh, until the next step (CA) is applied. Moreover, the mesh is adaptable to the new samples, which means that the designer could learn how to introduce new inputs an re-evaluate the results on the fly.

The SOM was built according to the following features:

- 2-dimensional input vectors
- Variable map resolution
- A Gaussian neighborhood.
- 2 learning phases: At the beginning the learning factor is tuned for exploration of the space and after 100 cycles it optimizes weights. The second phase stops after 10000 iterations or the variation is small enough.
- The weights are graphed in a 2-dimensional map of points, connected by lines representing the immediate neighborhood

It was implemented in Java and ran as an independent application, isolated from the rest of the experiment. Networks of 50x50, 100x100, 150x50 neurons were used, all of them running at a reasonable speed for its manipulation. The option of exporting the neurons' weights as a list of point coordinates was implemented in order to extract the desired configurations out of the program.

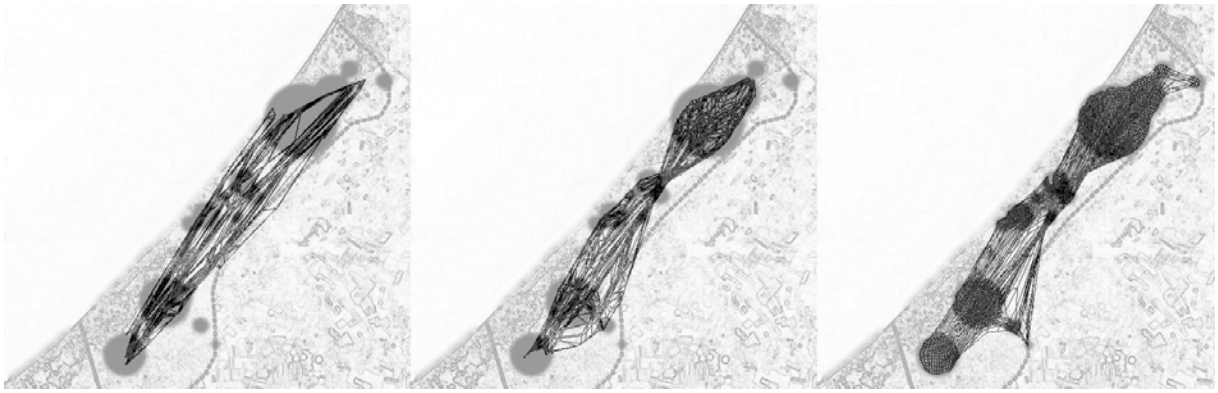

Figure 2. Evolution of the SOM algorithm applied on spatial data, in a vast land region.

3. Cellular Automata: topology reconfiguration

Cellular Automata are a very widespread method for modeling complex systems, such as urban growth. There have already been much work using this models in architecture and urbanism and even specialized ones have been developed [Torrens, 2000]. The discussion of these goes beyond the scope of this paper.

We have seen an algorithm that yields and adaptive mesh, continuously changing over time until we decide to halt it and extract the output. This mesh is an ordered set of points, which interpretation depends on the supplied data.

On the other hand, CA use a lattice of regularly spaced cells with no other geometrical information than the relationship among them. They are built from individual cells that contain all the information needed to update the state. Furthermore, the only external information to the cell comes directly from the adjacent cells, which along with it forms this neighborhood.

The proposal is to run a CA algorithm on top of the previously generated mesh, matching each neuron to a CA cell. The only prerequisite is that the CA dimension and the lattice resolution match exactly to the mesh.

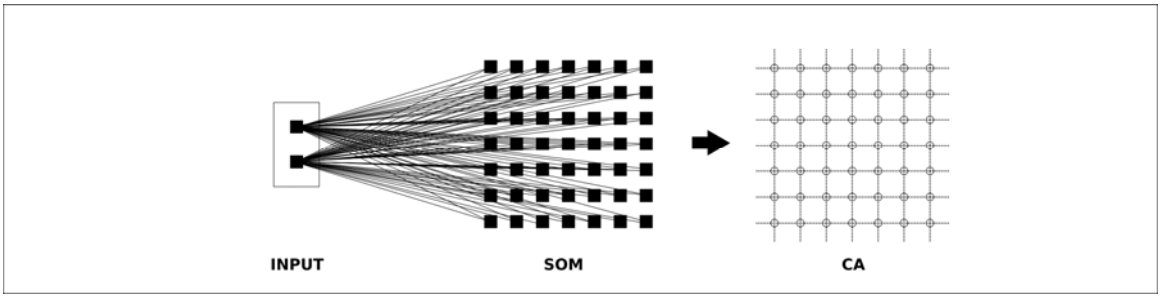

Figure 3. SOM-CA mapping

The CA task is to activate or deactivate a neuron for the next weights adjustment (modifying the neighborhood) or just for the output. By this way we obtain two different possibilities:

- *Filtered flow:* selecting the output units, a linear approach.

- *Feedback flow:* reconfiguring the neighborhood through the suppression of some units, a cyclic approach.

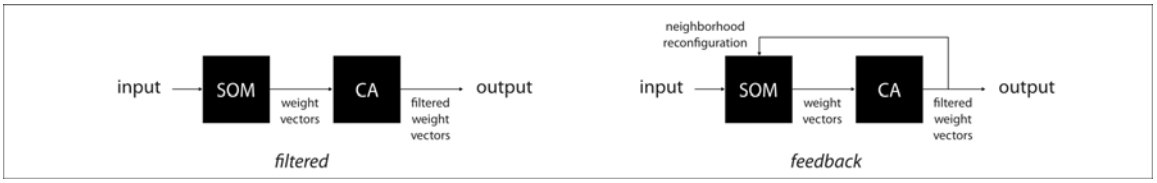

Figure 4. SOM-CA data flow methods.

Both of them result in the *topology reconfiguration* of the mesh, modifying the neurons space. This neurons space acts as the lattice of the CA, which itself has a constant cartesian topology (coming out from the SOM). That is necessary for a regular study of patterns formation [Marr, Hütt, 2005]. These patterns are then used for the reconfiguration of the SOM. It implies that there is always a virtual topology where the CA lives, and an actual topology that is being constantly regenerated by the cellular automata. The use of the CA paradigm for addressing the topology design has recently been demonstrated successful with several approaches [Abdalla, Setoodeh, Gürdal, 2006]. This activate/deactivate behavior could also be extended with other parameters from more complex, even n-dimensional models [Castro, Cabañero, 2007].

The tests were made using the simplest option (filtering), so the separated programs approach was reasonable. A more detailed study should be dedicated exclusively to the feedback configuration.

This second program was also implemented in Java providing the following features:

- Time control: start/stop
- Direct interaction with the CA cells: manually add/erase
- A set of implemented Classic CA, plus traffic-pedestrian simulation and other interesting rules to test
- The ability of changing rules on the fly
- A stack of rule processes, allowing to apply more than one rule in the same run, in a linear and ordered manner

As the previous program, the mesh generator, it is able to export the data at a specific time and extract it from the program.

Merging the two algorithms: results

Both programs were implemented with the ability to export their results, so we can use the frozen data externally. In this last section it is described briefly how this information has been introduced into the designer's workflow, the last step of the technique. The most important thing that has to be performed is to process that information into a 3-dimensional geometry inside a design tool. The chosen platform was the open-source 3d modeling software called Blender, running an embedded Python interpreter.

The python algorithm is straightforward. These steps are to be followed:

1. Read the SOM mesh data points
2. Read the CA cells status
3. Match the SOM neurons with the CA cells, activating or deactivating the neurons
4. Project the SOM in the 3d space, on the XY plane

5. Optional use of an interpretative/generative algorithm to create volumes or additional point properties. It is an interesting step, in which relies the final aspect of the urban plan as it introduces the actual phenotype directly. It is important to notice that although it will introduce great differences in the system, the underlying structure will remain dependent on the previous steps. If this step is avoided, the designer will work directly with the imported 2-dimensional mesh.

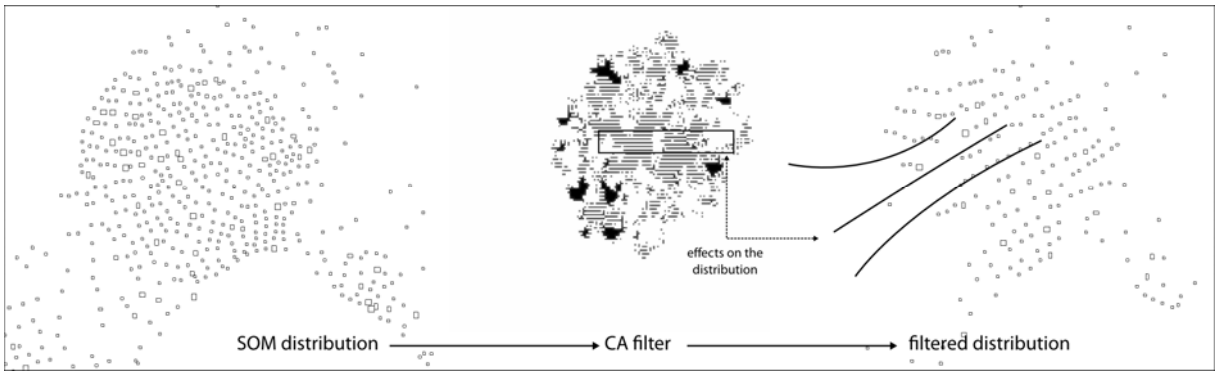

Figure 5. Example application: effects CA-filtered SOM.

Conclusion

The degree of success this technique could have in the design process is a matter of discussion both in the practical and the theoretical domains. However, different applications have already shown that it can be used as an aid tool with more general ambit than the regular grid. Becoming more than an aid tool, capable of synthesizing part of a design is a promising but ambitious objective.

Future work will develop this issues:

- Input data preprocessing
- Feedback flow as a more powerful and complex mechanism
- Multidimensionality of the input data
- Creation of layered or 3-dimensional meshes

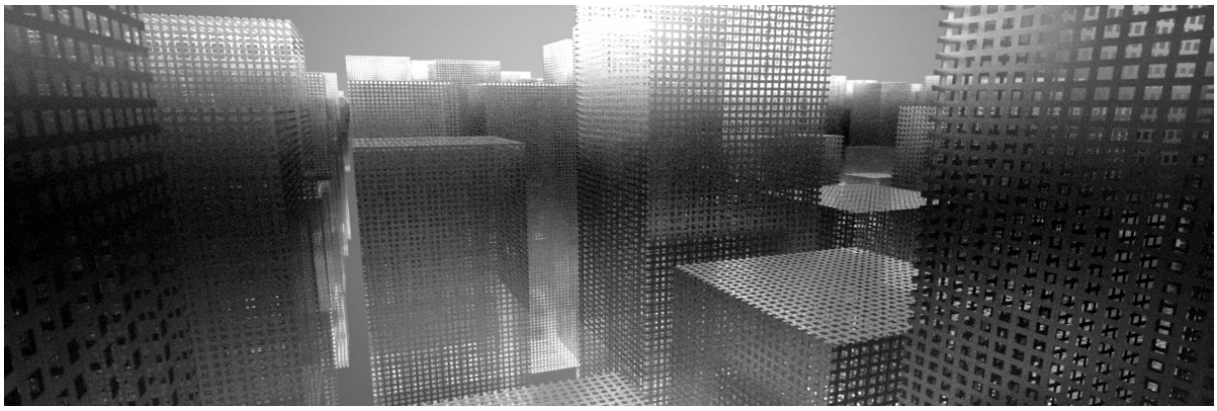

Figure 6. View of a prototype of synthesized city generated with this system.

Bibliography

[Ilachinsky, 2001] Andrew Ilachinsky. Cellular automata - A Discrete Universe. World Scientific Publishing Co. Pte. Ltd., Singapore, 2001.

[Kohonen, 2001] Teuvo Kohonen. Self-Organizing Maps. 3rd Edition. Springer-Verlag, Berlin, 2001.

[Colonna, Di Stefano, Lombardo, Papini, Rabino, 1998] Colonna, Di Stefano, Lombardo, Papini and Rabino. Learning Cellular Automata: Modelling Urban Modelling. In: Proc. of the 3rd International Conference on GeoComputation, University of Bristol, 1998.

[Batty, Xie, Sun, 1999] M. Batty, Y. Xie, Z. Sun. Modeling urban dynamics through GIS-based CA. In: Computers, Environment and Urban Systems. Elsevier, 1999.

[Portugali, Benenson, 1997] J. Portugali, I. Benenson. Individuals' cultural code and residential self-organization in the city space. In: Proceedings of GeoComputation '97 & SIRC '97. University of Otago, New Zealand, 1997.

[Zhongwei, 2003] Sun Zhongwei. Simulating urban growth using cellular automata. Intl. Inst. for Geo-information science and Earth observation. The Netherlands, 2003.

[Abdalla, Setoodeh, Gürdal, 2006] M. M. Abdalla, S. Setoodeh, Z. Gürdal. Cellular automata paradigm for topology optimisation. In: IUTAM Symposium on Topological Design Optimization of Structures, Machines and Materials: Status and Perspectives. Springer, Netherlands, 2006.

[Freeman, Skapura, 1991] J. A. Freeman, D.M. Skapura. Neural Networks. Algorithms, Applications and Programming Techniques. Addison-Wesley Publishing, USA, 1991.

[Torrens, 2000] P. M. Torrens, How Cellular Models of Urban Systems Work. CASA Working paper series, paper 28. CASA, London, 2000.

[Marr, Hütt, 2005] C. Marr, M. Hütt. Topology regulates pattern formation capacity of binary cellular automata on graphs. Physica A 354 (2005). Elsevier, 2005.

[Castro, Cabañero, 2007] Álvaro Castro, Carlos Cabañero. IC_ emergent processes particles projector. In: AMinima, Barcelona, 2007.

Authors' Information

Álvaro Castro Castilla - Universidad Politécnica de Madrid, Escuela Técnica Superior de Arquitectura, Madrid, Spain; e-mail: alvaro@endosymbionts.com

Nuria Gómez Blas - Dept. Organización y Estructura de la Información, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain; e-mail: ngomez@dalum.eui.upm.es