Self-Organizing Map and Axial Spatial Arrangement
Topological Mapping of Alternative Designs

Yoshihiro Kobayashi, Ph.D.
Meenakshi Sharma
School of Architecture and Landscape Architecture
Arizona State University

Abstract
This research attempts to formulate a computational framework for exploring spatial arrangements in the early phases of design. In the physical world, this could be compared to exploring spatial arrangements using cardboard cut-outs or simply a grid of spaces on paper. This research demonstrates the framework by means of a generative design system that introduces axial order in a plan parti made up of discrete 3D objects. The tool is designed to organize the 3D objects along an Axis specified by the user and also rearrange them following user-defined mathematical expressions. The numerical parameters (the dimensions and physical properties of the individual objects) are linked through the mathematical expressions to vary the spatial arrangement of objects. Implementation of the tool involves the Self Organizing Maps (SOMs) as the Graphical User Interface (GUI) in generative systems. This allows the user to select and dynamically view spatial arrangements that have been organized on a map based on their similarity. The application is implemented, tested, and its results are demonstrated using buildings designed by Louis I. Kahn, Frank Lloyd Wright and Mies van der Rohe.

Introduction
Design is, always has been, and always will be concerned at its central core with the manipulation of form, with composition, understood as the putting together of twodimensional and three-dimensional components, either spaces or material elements, in arrangements or configurations (Steadman 1983). This study attempts to develop a framework for producing a generative system which would aid the architect in the exploration of form, space and order in the early phases of design.

This study introduces a new approach to exploring Spatial Arrangements computationally. This approach is demonstrated by means of an application that can arrange and redistribute 3D objects along an Axis based on their individual physical properties. It forms the constraint that introduces order in the arrangement. The design approach supported by such a generative system would be ‘whole-to-part’. Rules and constraints would be laid upon the whole, i.e. the plan parti, in a manner to affect the parts and their relationships. Consequently, the same rules apply to all the parts, yet they would respond uniquely according to their own physical properties. The physical
properties of the objects considered are their position with respect to the Axis, area, and volume.

For the computer tool to be able to meet with its objective of supporting the design process, the design of the Graphical User Interface (GUI) is crucial. Most of the Computer-Aided Design (CAD) applications (i.e. AutoCAD, Maya, formZ, 3D Studio Max, Revit) have basic transform functions such as scaling, translating, rotating, etc. for 3-Dimensional Modeling. More advanced applications have more complex transform functions, in which the user needs to specify many parameters. However, the existing approaches restrict the space of design alternatives, make it difficult to compare alternatives, and require tedious user intervention. One of the identified difficulties of the generative systems is for the user to understand and benefit from the thousands of possible solutions generated by the system. Therefore, any help the system provides in sorting the generated solutions based on a criteria can help the architect use the solutions better in design decision making. Considering these needs of the designer and what the current applications offer, this study proposes to use Self-Organizing Map (SOM) as a fresh approach to the GUI in this field. SOM has an advantage of being able to organize the alternative designs graphically on the basis of the similarity, making it easier for the user to select and display the appropriate arrangement by mouse actions. The proposed tool produces generative diagrams that are schematic statements with which one begins conceptualizing a project.

Related Work

The research addresses three study areas as listed below. The related work in these fields is discussed here.

Transformation Functions in CAD Applications

Parametric and Constraint-Based transformations form an important tool for analysis as well as invention of architectural form (Burry 2003). As discussed above, the existing approaches of most CAD and CG applications restrict the space of design alternatives and require a tedious user intervention. First, the selected objects undergo the same transformation, irrespective of their individual physical quantities like position, area, or mass. Second, the users need to specify many parameters before applying any complex transform functions. In the existing approaches, the parametric transformations are done by a variety of means through interaction with the keyboard, mouse and monitor in combination, or by direct access to the database that contains information that can be updated externally while a model is being constructed (Burry 2003). For instance, applications such as Maya by Alias|Wavefront and 3D Studio Max by Discreet have an intimidating number of menus, modes, and widgets for object transformations and scene manipulation (Smith, Salzman & Stuerzlinger 2001).

User Interaction

In order to manipulate the constraint-based spatial arrangement for 2D and 3D arrangements, a number of researchers...
have introduced both, 2D and 3D input devices. Some of the successful relevant studies involving 2D input devices are the Multi Interface virtual Environment (MIVE) Interface (Smith et. al. 2001) and ‘Object Associations’ (Bukowski and Sequin 1995; Goesele and Stuerzlinger 1999). Both these examples also demonstrate use of constraints, using the mouse as the input device.

In MIVE, the mouse is used to move and rotate objects using two buttons. Upon moving and re-locating the object, the constraint is tested. The disadvantages are, one, that the user spends a considerable amount of time in selecting the objects and, two, defining constraints for new objects is a complex task. In ‘Object Associations’, the 2D mouse motion is mapped to the vertical and horizontal transformations of an object’s position. The disadvantages in this case are the time lag between actions and also that only one object can be selected at a time to perform the transformation (Smith et. al. 2001).

**Self-Organizing Maps**

Self-Organizing Maps (SOMs) are a data visualization technique invented by Professor Tuuvo Kohonen in 1982 (Hodju and Halme 1999). SOMs are self-adaptive topological maps inspired by the perception systems found in the mammalian brain (Hodju and Halme 1999). In the brain, the different sensations are represented topologically. The sensations from closely related organs are processed by cells that are topologically closer to each other on the surface of the cortex in the brain. For instance, senses of seeing and smelling are processed by neighboring cells in the cortex.

Most neural network applications are trained by feeding it with a lot of sample data that contains the inputs and corresponding outputs that you would normally expect in your process. In contrast to this, SOM is based on unsupervised learning. The SOM can be visualized as a sheet-like neural-network array, the map units (or neurons) of which become specifically tuned to various input signal patterns in an orderly fashion (Honkela 1997). SOMs are capable of managing high dimensional data that is beyond the human capacity. For example, colors in the RGB model have three-dimensional data for red, green and blue values. To arrange colors manually according to their similarities would be manually time intensive and inaccurate. The SOM is used to map a multi-dimensional dataset onto a one- or two-dimensional surface which plots the similarities of the data by building clusters of similar data items (Germano 1999). The location of each map unit or neuron carries semantic information, preserving the topology of the data points. The SOM serves as a clustering tool for high-dimensional data, making visualization easier by using a typical two-dimensional shape (Hodju and Halme 1999).

The SOM is widely used as a data mining and visualization method for complex data sets. Application areas include speech recognition, character recognition, image processing, economical analysis, and diagnostics in industry and in medicine (Honkela 1997).

In this study, SOM is used to automatically organize and map the mathematical functions topologically according to the resulting locations of 3D
objects. This implies that the arrangements similar to one another would be assigned cells or nodes closer in the SOM, when compared to the not-so-similar ones. By clicking on a node of the SOM, the user is able to view the corresponding arrangement.

**Methodology**

**The Mathematical Model**

From architectural philosophy, the proximity of the 3D objects to each other, and to the Axis, define the axially of the plan parti. This implies that to get differential spatial arrangements, the framework must involve a method of manipulating the distance of objects from the Axis using parameters. These spatial arrangements can be compared according to their dispersion along the Axis, termed as the 'relative axial dispersion factor', which is discussed later. Arrangements are similar to one another when their relative axial dispersion factor is small in value.

In this study, a mathematical model has been formulated using concepts of parametric modeling to build linear and non-linear equations in order to define the position of each part in an architectural whole. The parameters considered as inputs for these equations are the distance from the Axis, area, and volume. As a result, the mathematical model builds the transform operations applied to the architectural whole.

Applying the mathematical model to the axial systems, axially or the distance of each object from the Axis would be a function of original distance, area, and volume of each element in the plan parti. The linear equations were built manually by applying weights to distance, area, and volume. These were tested on sample cases and seeing the results, the decision was made whether to include these as executable functions with valid results.

Figure 1 shows a complete listing of linear and non-linear functions that are used for the application with Function ID 0 as the original distance. The above listed functions form the 100 sample datasets used to demonstrate the application. For instance, the function f(3) (refer to Figure 1) is applied to a set of 3D objects to get the new distance of objects from the Axis. The input parameter in this case is old distance. For example, \{0.5, -0.8, 0.3\} is an input dataset for an arrangement of three objects which are distanced from the Axis 0.5 units in positive direction for first object, 0.8 units in negative direction for second object, and 0.3 in positive direction for third object. Applying function f(3) to this arrangement, the resulting dataset would be \{0.25, -0.4, 0.15\}. This function applies the transform operation to move each object perpendicular to the Axis (the user defines) in varying amounts. Similarly, functions can be developed to move objects parallel to the Axis defined by the user and to rotate, mirror and scale them. This is currently beyond the scope of the study.

**Mapping Functions onto the Self-Organizing Map**

The planar SOM consists of a regular grid of neurons as the processing units. For the purpose of this application, a 2-dimensional SOM is used. To map the solution sets of 100 functions listed in Figure 1, we use a SOM of 10 \times 10 array
<table>
<thead>
<tr>
<th>ID</th>
<th>New Distance to Axis</th>
<th>ID</th>
<th>New Distance to Axis</th>
<th>ID</th>
<th>New Distance to Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>dis</td>
<td>34</td>
<td>0.0025*volume</td>
<td>67</td>
<td>1.5<em>dis+0.005</em>area+0.001*volume</td>
</tr>
<tr>
<td>1</td>
<td>0.1*dis</td>
<td>35</td>
<td>0.00075*volume</td>
<td>68</td>
<td>1.5<em>dis+0.005</em>area+0.001*volume</td>
</tr>
<tr>
<td>2</td>
<td>0.25*dis</td>
<td>36</td>
<td>-0.001*volume</td>
<td>69</td>
<td>4.0<em>dis+0.005</em>area+0.001*volume</td>
</tr>
<tr>
<td>3</td>
<td>0.5*dis</td>
<td>37</td>
<td>-0.0001*volume</td>
<td>70</td>
<td>2.0<em>dis+0.005</em>area+0.001*volume</td>
</tr>
<tr>
<td>4</td>
<td>0.66*dis</td>
<td>38</td>
<td>-0.00025*volume</td>
<td>71</td>
<td>2.0<em>dis+0.05</em>area+0.0001*volume</td>
</tr>
<tr>
<td>5</td>
<td>0.75*dis</td>
<td>39</td>
<td>-0.0005*volume</td>
<td>72</td>
<td>-1<em>dis+0.05</em>area+0.001*volume</td>
</tr>
<tr>
<td>6</td>
<td>1.2*dis</td>
<td>40</td>
<td>-0.00075*volume</td>
<td>73</td>
<td>dis+0.05<em>area+0.001</em>volume</td>
</tr>
<tr>
<td>7</td>
<td>1.5*dis</td>
<td>41</td>
<td>dis+0.02*area</td>
<td>74</td>
<td>dis+0.02<em>area+0.001</em>volume</td>
</tr>
<tr>
<td>8</td>
<td>1.75*dis</td>
<td>42</td>
<td>dis+0.02*area</td>
<td>75</td>
<td>(\sqrt{\frac{\text{area}}{\text{dis}}(\text{volume})^{1/3}})</td>
</tr>
<tr>
<td>9</td>
<td>2.0*dis</td>
<td>43</td>
<td>dis+0.025*area</td>
<td>76</td>
<td>(\frac{\text{area}}{\text{dis}})</td>
</tr>
<tr>
<td>10</td>
<td>2.5*dis</td>
<td>44</td>
<td>dis+0.05*area</td>
<td>77</td>
<td>(\frac{\sqrt{\text{area}}}{\text{dis}})</td>
</tr>
<tr>
<td>11</td>
<td>-0.1*dis</td>
<td>45</td>
<td>dis+0.075*area</td>
<td>78</td>
<td>(\frac{(\text{volume})^{1/3}}{\text{dis}})</td>
</tr>
<tr>
<td>12</td>
<td>-0.25*dis</td>
<td>46</td>
<td>dis-0.01*area</td>
<td>79</td>
<td>(\frac{\sqrt{\text{area}}}{\text{dis}})</td>
</tr>
<tr>
<td>13</td>
<td>-0.5*dis</td>
<td>47</td>
<td>dis-0.02*area</td>
<td>80</td>
<td>(\frac{(\text{volume})^{1/3}}{\text{dis}})</td>
</tr>
<tr>
<td>14</td>
<td>-0.66*dis</td>
<td>48</td>
<td>dis-0.025*area</td>
<td>81</td>
<td>(\sqrt{\text{area}})</td>
</tr>
<tr>
<td>15</td>
<td>-0.75*dis</td>
<td>49</td>
<td>dis-0.005*area</td>
<td>82</td>
<td>((\text{volume})^{1/3})</td>
</tr>
<tr>
<td>16</td>
<td>-1.2*dis</td>
<td>50</td>
<td>dis-0.075*area</td>
<td>83</td>
<td>(\sqrt{\text{area}})</td>
</tr>
<tr>
<td>17</td>
<td>-1.5*dis</td>
<td>51</td>
<td>dis+0.001*volume</td>
<td>84</td>
<td>(\sqrt{\text{area}})</td>
</tr>
<tr>
<td>18</td>
<td>-1.75*dis</td>
<td>52</td>
<td>dis+0.0001*volume</td>
<td>85</td>
<td>(\sqrt{\text{area}})</td>
</tr>
<tr>
<td>19</td>
<td>-2.0*dis</td>
<td>53</td>
<td>dis+0.00025*volume</td>
<td>86</td>
<td>(\sqrt{\text{area}})</td>
</tr>
<tr>
<td>20</td>
<td>-2.5*dis</td>
<td>54</td>
<td>dis+0.0005*volume</td>
<td>87</td>
<td>(\sqrt{\text{area}}) (\text{+(volume)^{1/3}})</td>
</tr>
<tr>
<td>21</td>
<td>0.01*area</td>
<td>55</td>
<td>dis+0.00075*volume</td>
<td>88</td>
<td>(\sqrt{\text{area}}) (\text{+(volume)^{1/3}})</td>
</tr>
<tr>
<td>22</td>
<td>0.02*area</td>
<td>56</td>
<td>dis+0.0001*volume</td>
<td>89</td>
<td>(\sqrt{\text{area}}) (\text{-(volume)^{1/3}})</td>
</tr>
<tr>
<td>23</td>
<td>0.025*area</td>
<td>57</td>
<td>dis+0.00025*volume</td>
<td>90</td>
<td>(\sqrt{\text{area}}) (\text{-(volume)^{1/3}})</td>
</tr>
<tr>
<td>24</td>
<td>0.05*area</td>
<td>58</td>
<td>dis+0.00025*volume</td>
<td>91</td>
<td>sin(dis)(0.01*area)</td>
</tr>
<tr>
<td>25</td>
<td>0.075*area</td>
<td>59</td>
<td>dis+0.00025*volume</td>
<td>92</td>
<td>sin(dis)(0.001*area)</td>
</tr>
<tr>
<td>26</td>
<td>0.01*area</td>
<td>60</td>
<td>dis+0.00075*volume</td>
<td>93</td>
<td>sin(dis)(0.03*area)</td>
</tr>
<tr>
<td>27</td>
<td>-0.02*area</td>
<td>61</td>
<td>0.2<em>dis+0.01</em>area-0.001*volume</td>
<td>94</td>
<td>sin(dis)(0.001*area)</td>
</tr>
<tr>
<td>28</td>
<td>-0.025*area</td>
<td>62</td>
<td>0.2<em>dis+0.01</em>area-0.0005*volume</td>
<td>95</td>
<td>cos(dis)(0.01*area)</td>
</tr>
<tr>
<td>29</td>
<td>-0.05*area</td>
<td>63</td>
<td>0.2<em>dis+0.01</em>area-0.0001*volume</td>
<td>96</td>
<td>cos(dis)(0.001*area)</td>
</tr>
<tr>
<td>30</td>
<td>-0.075*area</td>
<td>64</td>
<td>-0.2<em>dis+0.01</em>area+0.001*volume</td>
<td>97</td>
<td>cos(dis)(0.01*area)</td>
</tr>
<tr>
<td>31</td>
<td>0.001*volume</td>
<td>65</td>
<td>-0.2<em>dis+0.01</em>area+0.0005*volume</td>
<td>98</td>
<td>cos(dis)(0.001*volume)</td>
</tr>
<tr>
<td>32</td>
<td>0.00025*volume</td>
<td>66</td>
<td>-0.2<em>dis+0.01</em>area+0.0001*volume</td>
<td>99</td>
<td>area<em>0.01</em>cos(volume/100.0)</td>
</tr>
</tbody>
</table>

Figure 1. A complete listing of linear and non-linear functions that are used for the application.
of nodes. Each node would contain n-dimensional data for a plan parti made up of n objects. The typical sample data for each node would be \{x_1, x_2, x_3, x_4, ..., x_n\} for n objects. The sample data is randomly calculated and assigned to each of the 100 nodes of the SOM. This is done because the input data is unknown and varies for each input set. This could be compared to a jigsaw puzzle, where not knowing where to start, the user randomly throws the pieces and then begins to find similar color or pattern pieces and puts them together.

The input vectors are formed by the solution sets resulting from applying the 100 functions listed in Figure 1. The solution sets represent the distance of each object of the plan parti from the Axis. This implies that for ‘n’ objects, there will be ‘n’ dimensions. This way, the nodes on the SOM represent the mathematical functions that produce results similar to that of the sample data of the neuron. And the functions would be arranged topologically, such that the ones with similar solution sets lie closer to each other compared to dissimilar ones. In this application, the ID of each function (Figure 1) represents the solution set of that function on the SOM visually (see Figures 5, 6, 7).

A detailed discussion of the SOM algorithm is beyond the scope of this paper. Included in this section is the basic working of the SOM algorithm as used in this study:

Step 1: Initializing the neurons—The neurons of the SOM are initialized and randomly assigned values from the dataset.

Step 2: Selecting an input vector—An input vector is randomly selected from the other input vectors calculated from the 100 defined functions.

Step 3: Finding the winning neuron—The distance between each of the nodes and the selected input vector is calculated. This is done by using the Euclidean distance. For instance, the distance between the input vector \{0.5, 0.25, -0.3\} and neuron \{0.3, 0.2, 0.0\} for a set of three 3D objects would have distance calculated as distance:

\[
5.0(\sqrt{(0.5 - 0.3)^2 + (0.25 - 0.2)^2 + (-0.3 - 0.0)^2})
\]

The distance here forms the Relative = Axial Dispersion Factor (see Figure 2).

This is calculated for all 100 neurons. The neuron with the least distance is the winning neuron. This forms the test for similarity (i.e., the less the Relative Axial Dispersion Factor, the closer the solutions are to each other).

In Figure 2, the input vector is compared to two neurons of the SOM to calculate the Relative Axial Dispersion Factor.
(x(t) - Wv(t)), where t = current iteration
Wv = current weight vector x = selected
input vector _(t) = neighborhood function
_(t) = learning constant at iteration t.
In the application, _(0) = 0.99 for the first
iteration. Over each iteration, the learning
constant is reduced by using the function:

_(t+1) = _(t) * 0.99

This way the values are: _(1) = 0.98,
_(2) = 0.96 and _(3) = 0.92 until the
iterations reach very insignificant values
for t number of iterations. The application
is tested for 500 iterations. Figure 3 shows
the neighborhood of the winning neuron
darkest shade of blue) in the application.
 Decreasing intensity of color indicates
increasing distance from the winning
neuron. In the application, each neuron has
a neighborhood of 18 neurons.

The neighborhood function, _, depends
on the learning constant and the distance
of neuron from the winning neuron. In the
application, the neighborhood function is
defined as:

_(t) = _(t) * (0.5)^distance

, where the distance
values are assigned as 0, 1, \(\sqrt{2}\), 2 and 2

moving from the winning neuron to the
farthest neuron in its neighborhood. Using
these functions, the sample data for the
neurons is updated to be closer to the
input vector.

Step 5: Steps 2 through 5 are repeated
for each input vector, for t number
of iterations. This trains the SOM neurons,
which display clustering of similar
data items. The input vectors that are
associated with mathematical functions are
mapped onto the ordered SOM.

When the user selects a node from
the SOM, the function (represented by
the function ID) associated with that node
is applied to the 3D objects in the spatial
arrangement which impacts the position
of each of the objects with respect to the
Axis.

System Framework

The tool is programmed as a Java
application with Swing UI components.
The plan parti for case study buildings are
included in the programming code. Each
3D object is defined as an array of points
(coordinates of the base polygon) with the
height. The mathematical expressions are
defined as functions of distance from Axis,
area, and volume parameters.

The interface is divided into two
screens (Figure 4). The first one—the

Figure 3. The neighborhood of the winning neuron.

Figure 4. Layout of User Interface in the implemented
tool.
‘drawing panel’ appears when the application is started. It consists of the 3D Modeler that displays the plan parti made up of 3D objects. It includes the drawing toolbar allowing the user to draw the Axis and includes the viewing actions of zoom, pan, 3D Orbit, etc. The pulldown menu ‘Case Studies’ allows the user to select one from the case study buildings. The three-dimensional counterpart of the plan parti of the building appears on the ‘drawing panel’. After using the viewing options to get an optimal view of the plan parti, the user would select the pencil tool to select the start point and end point of the Axis. Once the line drawn is selected and ‘SOM’ is selected from the pull-down menu, a new screen appears—the ‘SOM panel’. This panel is further divided into two panels, one that displays the SOM itself with the functions mapped on it. The second panel displays the transformed plan parti. When the user clicks on any cell of the SOM populated by function IDs, the corresponding function is applied to the arrangement and the new arrangement is displayed in the ‘SOM panel’. When the user clicks on any empty cell, the original arrangement is achieved. When the user clicks on the ‘Update’ button, the arrangement in the ‘drawing panel’ is then updated as the new one and, in turn, the SOM also gets updated. These steps, when used in combination can achieve a wide array of alternative arrangements.

**Testing and Discussion of Results**

Case studies were used to demonstrate the above described application. The testing of the application involves two approaches:

- **Axial Arrangements of 3D objects placed in space.**
- **Redistribution of axially arranged 3D objects along a newly defined Axis.**

Three popular examples in architecture that follow the ordering principle of Axis are considered for the testing as case studies. In the developed application, for each of the buildings, an Axis was defined in the plan parti by the user. The application generates the SOM by mapping the mathematical functions. The Axis in Case Study 1 (Figure 5) was defined against the Axis of the original arrangement, while Case Studies 2 and 3 had the Axis defined along the original arrangement (Figures 6, 7). Once the SOM was generated, the author saved all the resulting arrangements as images and manually mapped them to a blank SOM to study better how the definition of the Axis and the plan parti affects the arrangements and their corresponding location on the SOM. This was done to facilitate the discussion of results. Figures 5, 6, 7 show the original arrangement with the userdefined Axis, the resulting SOM and all the spatial arrangements corresponding to each populated node in the SOM for each Building.

- **Case Study 1:** Richards Medical Center (1957 to 1961) designed by Louis I. Kahn. The circulation within the building follows the axial movement, while not strictly.
- **Case Study 2:** Unity Temple (1906) designed by Frank Lloyd Wright. The plan is closely related to the centralized churches of the Renaissance period. This study
demonstrated the use of Axis comparable to that in classical architecture.

Case Study 3: Berlin Building Exposition (1931) designed by Mies Van Der Rohe.

The results of these case studies are closely studied to draw inferences on the impact of the parameters, the weights applied to them, and the mathematical expressions built for the application. The results are judged by the integrity of the axially generated arrangements. It was observed:

- SOM clearly organizes the resulting arrangements to form clusters of similar arrangements. Some similar arrangements may occupy different regions on the SOM, but are still grouped with other similar arrangements. To achieve a clear segregation, the number of iterations that the application goes through should be of higher value.
- More variations in the spatial arrangement are achieved by applying non-linear expressions.
- The direction of the Axis is noticeable in most arrangements. The results of the arrangements vary broadly from making the Axis dominant to making it almost unidentifiable.
- Each arrangement can be further explored when it replaces the original one.
- In Figure 5, the axially along the original Axis of the arrangement is easily perceivable when weights (within a certain range) are applied to the distance parameter only. When weights are applied to area and volume, the new Axis is more prominent than the original one. In this case, the SOM demonstrates a strong segregation of the arrangements with the new Axis being prominent (left side of the SOM) vs. the original Axis (right side of the SOM). In Figures 6 and 7, the defined Axis coincides with the original Axis of the arrangement, the functions manipulating only the distance give arrangements that maintain the relationship of objects to one another, as if being pulled near or further away using strings. This scenario does not lend as wide a range of arrangements as seen in Figure 5.
- Area and volume are significantly larger numerical parameters and are not affected by the location of the original Axis in the current arrangement of the plan parti. When weights are applied to area and volume, the arrangements show more variation than with distance. To generate an identifiable axis, small weights are applied to the area parameter and even smaller ones to the volume. In Figure 7, functions $f(9)$ and $f(70)$ (refer to Figure 1) occupy the same node on the SOM. This is because, when very small weights are applied to the already small magnitude of area and volume in objects of Case Study 3, the resulting values are negligible. The same functions $f(9)$ and $f(70)$ occupy different regions of the SOM in Case Study 1 (Figure 5) and Case Study 2 (Figure 6),
Figure 5. The SOM and resulting arrangements for Case Study 1 building.
Figure 6. The SOM and resulting arrangements for Case Study 2 building.
Figure 7. The SOM and resulting arrangements for Case Study 3 building.
• The digital support for creating spatial configurations toward the early phases of design.
• A mathematical model with easy and flexible manipulations.
• A graphical means of arranging and displaying similarity of results through the SOM.

Currently the application is designed for planar arrangements of 3D objects and limited to Axis-based transformations. The framework accommodates for easy manipulations of existing functions and more user-specific functions to be added to the application, adhering to the efforts toward permitting the users to formulate their own generative style. By introducing additional parameters to build functions, the user would have more control over spatial arrangements and a larger search space for optimal design solutions. Also, taking into account the topological relationships and neighborliness among the entities of the plan parti, the designer would be able to define the constraints better.

The results vary broadly from making the Axis very dominating in the arrangement to making it almost unidentifiable. This is when the user benefits by seeing a gamut of arrangements, some expected and others unexpected. Each arrangement can be further explored when it replaces the original one. The designer truly plays the role of evaluating, interpreting, and further exploring the results that the computer generates with the appropriate GUI.

Finally, more studies in different fields could benefit from this research. The approach can be applied to the relatively

**Conclusion**

The application successfully presents and demonstrates a user-friendly GUI that helps the designer explore axial arrangements dynamically with the help of mouse movements.

The three important contributions of the study to this research field are:

- The digital support for creating spatial configurations toward the early phases of design.
- A mathematical model with easy and flexible manipulations.
- A graphical means of arranging and displaying similarity of results through the SOM.

Currently the application is designed for planar arrangements of 3D objects and limited to Axis-based transformations. The framework accommodates for easy manipulations of existing functions and more user-specific functions to be added to the application, adhering to the efforts toward permitting the users to formulate their own generative style. By introducing additional parameters to build functions, the user would have more control over spatial arrangements and a larger search space for optimal design solutions. Also, taking into account the topological relationships and neighborliness among the entities of the plan parti, the designer would be able to define the constraints better.

The results vary broadly from making the Axis very dominating in the arrangement to making it almost unidentifiable. This is when the user benefits by seeing a gamut of arrangements, some expected and others unexpected. Each arrangement can be further explored when it replaces the original one. The designer truly plays the role of evaluating, interpreting, and further exploring the results that the computer generates with the appropriate GUI.

Finally, more studies in different fields could benefit from this research. The approach can be applied to the relatively
new field of generative systems, not just in architecture or design, but engineering, product design, and related fields.

The framework of the study does not emulate the way designers work while addressing a design problem. But, understanding the difference in human capacity and the computational one, the study is an effort toward integrating the two by taking advantage of what each offers to facilitate the exploratory phase of design. In conclusion, the application demonstrates itself as a useful tool that would be able to find acceptance in the design exploration by students and professionals alike.

References


