Performative Design and Fabrication of a Parametric Wall Screen for Tropical Climates

A Modular Approach

Thomas Wortmann¹, Bige Tuncer²
¹,²Singapore University of Technology and Design
¹thomas_wortmann@mymail.sutd.edu.sg ²bige_tuncer@sutd.edu.sg

We have developed a modular approach to the parametric design of a patterned façade for tropical climates, using a "lighter" data set and model that integrates a number of performance considerations. This modular approach separates the façade’s design into form, façade pattern, façade density requirements, and façade components, and reintegrates these aspects via a triangular mesh, represented as a fast and convenient data structure based on half-edges. Through this separation, the design team can simultaneously work on the architectural appearance of the design, its performance, and its fabrication, while retaining holistic control through the constant reintegration of design changes via the half-edge data structure. In this way, we retain the advantages of a parametrically driven design process, such as automatic design generation and the integration of performance aspects, while permitting more flexible and non-sequential design explorations by different members of the design team.

Keywords: Generative Design, Parametric Design, Performance-based Design, Computer-aided Design Tools, Modular Design

INTRODUCTION

Parametric design promises great opportunities to develop "performative" designs that respond to challenges such as global warming and diminishing natural resources (Kolarevic 2005; Hensel 2013). However, environments for the development of such parametric designs rely on topologically sorted graphs (Woodbury 2010), in other words, on linear flows of data from inputs to outputs, suggesting relatively linear design processes. In other words, a designer influences a design at the initial definition of the parametric system, and then only by the manipulation of its parameters. For example, Datta et al. present a parametric system that automatically generates a façade based on randomized parameters (2014). In performance-based design, performance values derived from simulations often take the role of design parameters (e.g., Oxman 2008), raising the question of how designers can integrate performance values into a parametric system while retaining control over the design's final appearance. Modularization approaches have been applied in order to integrate performance considerations into parametric design (Turrisi et al. 2012). However, these approaches often
create a set of parametric models that perform explorations in a rather linear manner.

Contrastingly, the literature on architectural design processes often emphasizes the non-linear, heuristic natures of architectural design processes (Rowe 1991; Lawson 2006). The increasing numbers of consultants and stakeholders involved in architectural design compound these difficulties (Loukissas 2012).

A powerful approach to improve the performance of architectural designs is BIM-modelling (Eastman 2011). However, BIM-modelling, although widely adopted for design documentation and analysis, appears unsuitable for early design stages due the difficulty of changing complex, information-rich BIM models. Such models consist of hundreds, and sometimes thousands, of "intelligent" objects, each with their own properties, behaviours, and relationships. Accordingly, Holzer (2007) identifies a need for "lighter data-sets and models", which, compared to BIM-models, are easier to change.

The modular approach to the parametric design of a patterned façade for tropical climates is an example of such a "lighter" data set and model that integrates a number of performance considerations. Our modular approach separates the façade's design into the four aspects of form, façade pattern, façade density requirements, and façade components, and reintegrates these aspects via a triangular mesh, represented as a fast and convenient data structure based on half-edges. Through this separation, the design team could simultaneously work on the architectural appearance of the design, its performance, and its fabrication, while retaining holistic control through the constant reintegration of design changes via the half-edge data structure. In this way, we retained the advantages of a parametrically driven design process - such as automatic design generation and the integration of performance aspects - while permitting more flexible design explorations by different members of the design team.

**CASE STUDY: A MODULAR APPROACH TO PERFORMATIVE DESIGN**

Specifically, we present the design process of a panelized façade for a free form, multi-functional pavilion in Singapore. The pavilion's façade combines structural requirements with shading and natural ventilation through a pattern of variable density, derived from a limited set of parametric, triangular tiles. To develop this façade, the design team created and manipulated digital models, analyzed structural and daylight performance, and digitally fabricated physical models and prototypes.

**Representing Form with the Half-edge Data Structure**

The design team modelled the pavilion's shape as a triangular mesh, i.e. a collection of triangles (faces), each of which was defined by three corner points (vertices). Specifically, every mesh triangle served as the base geometry of one façade component. (For this base geometry, the thickness of the façade components is not taken into account.) In such a mesh, the sides of the triangles, i.e. the connections between vertices, are called edges.

To implement the integration described above in an efficient manner, we needed to represent the triangular mesh with a data structure that would quickly find the vertices and edges of a face, the one or two faces incident to an edge, and the edges and faces incident to a vertex (the so-called one-ring neighborhood). Of the four types of data structures for polygonal meshes listed by Botsch et al. (2010) - Face-Based, Edge-Based, Half-Edge-Based, and Directed Edge - only the half-edge data structure supports the required queries without additional computations (p. 25-27), which is why we chose it to represent the pavilion's architectural form and façade components. However, the 3D-modelling environment employed for the design (Rhinoceros) only provided face-based data that we had to convert into a half-edge data structure. (See figure 1 for a visual comparison of the face-based and half-edge-based data structures.)
**Face-based Mesh Representation.** The face-based data structure is the smallest and simplest, and consists of an array of vertices and an array of faces, with each face defined as an array of indices into the array of vertices (ibid., p. 22-24). This data structure underlies common file formats for the exchange of three-dimensional information (e.g. STL and OBJ), and is available in scripting languages (e.g. RhinoScript and Maya Embedded Language) and visual programming interfaces (e.g. Grasshopper) for 3D-modelling environments. For example, Aish et al. (2014) employ this representation to represent a triangulated façade in DesignScript. However, although topological information such as the incident faces to an edge or the one-ring neighbourhood of faces around a vertex is implicitly available in the vertex and face arrays of the face-based data structure, the retrieval of such information is slow and difficult to implement (Botsch et al. 2010). Specifically, finding the edges incident to an edge or vertex requires scanning all faces. This scan takes linear time or, in asymptotic notation, $O(n)$, where $n$ is the number of vertices.

**Half-edge-based Mesh Representation.** Half-Edge-based data structures address this problem by splitting every edge into two half-edges pointing in two opposite directions, and by storing mesh vertices, half-edges, and faces in three interrelated arrays (ibid., p. 25-26). The vertex array stores the location of every vertex, as well the index of one incident half-edge. The half-edge array stores the indices of the start and end vertices for every half-edge, as well as the previous, next, and opposite half-edges. (The opposite half-edge is referred to as the twin.) The face array simply stores the index of one incident half-edge for every face. One can now enumerate the vertices of a face by repeatedly querying for the next half-edge. Similarly, one can enumerate the half-edges and faces incident to a vertex by repeatedly querying for the next half-edge of the opposite, twinned, half-edge. Despite their usefulness, half-edge data structures are less available in scripting and visual programming interfaces than face-based representations, possibly because of their relative complexity. (An exception is Plankton, a free third party plug-in for Grasshopper that provides an implementation of the half-edge data structure.)

**Converting a Face-based Mesh Representation into a Half-edge-based Representation.** Converting face-based data into half-edge data takes two straightforward steps:

1. First, one traverses the sides of every face (ac-
According to the arrays of indices in the face-based face array, creating one half-edge per side and assigning previous and next half-edges according to the traversal of each face. The traversal and assignment ensure that the half-edges for each face form a closed ring, and that corresponding, i.e. opposite, half-edges are pointing in opposite directions. (Assuming that the normal directions of the original mesh are uniform, which is usually the case in 3D-modelling applications, the order of vertices defining a face in a face-based data structure is either uniformly clockwise or counter clockwise.) Simultaneously, one creates a new face array by storing the first half-edge of each face. The number of operations for this step increases linearly with the number of vertices, or $O(n)$, since it requires one traversal per face.

(2) The second step pairs half-edges with their twins. (As explained above, twinned pairs have opposite directions as long as the original mesh displays unified normals, but one still needs to find them.) Finding these pairs with brute force by comparing each half-edge against every other half-edges takes quadratic time, or $O(n^2)$. Sorting the half-edges according to the coordinates of their mid points in the x, y, and z directions improves the required number of operations to linearithmic time, or $O(n \log n)$. Due to the sorting, twinned half-edges adjoin in the half-edge array, and thus can be referenced to each other in a single pass, i.e. in linear time, while the three sorts in x, y, and z take linearithmic time.

The above algorithm allowed us to convert the face-based representation of the triangle mesh provided by the 3D-modelling environment into a half-edge data structure in linearithmic time, which in practice did not take more than a number of seconds. The resulting half-edge data structure processes topological queries about neighbouring elements in constant time, or $O(1)$. One can also compute vertex normals, edge normals, and face normals in constant time by taking the cross product of two adjacent edges to compute face normals, and by averaging the face normals of adjacent faces for edge and vertex normals. Consequently, the calculation of all edge normals takes linear time with the half-edge data structure, opposed to the exponential time required with a face-based data structure. Since the design team modelled the pavilion in Rhinoceros, we implemented the half-edge data structure and conversion algorithm with RhinoScript and IronPython. The next section discusses the pattern we applied to the façade.

**Density-modulating Pattern**

Our aim was to design a pattern that could achieve a gradient of densities, and thus address structural, shading, and programmatic requirements, from a limited number of tiles. Furthermore, the pattern should make the individual tiles less apparent visually. We could then assign a pattern tile to every façade component (i.e., to every mesh triangle), resulting in a cohesive pattern of variable density for the pavilion’s façade overall.

The pattern consists of thirteen parametric tiles, as depicted in figure 2. Defining each tile are its three corner points and the three density values associated with them, with the density values spanning a range from 0 to 4. The five tiles in the middle row represent tiles of homogenous density, progressively densifying from left to right, while the tiles in the top row represent tiles where one corner is less dense than the others, and the tiles in the bottom row represent tiles where one corner is denser than the others. Note that the design team intended the lines associated with densities 0 and 1 as structural elements, and the lines associated with densities 2, 3, and 4 as non-structural shading devices. Since every mesh triangle of the pavilion’s form corresponds to a pattern tile, and every mesh triangle to three vertices, one can define a pattern for the whole mesh by assigning density values from 0 to 4 to the vertices of the mesh. The next section describes this assignment of density values.
Integrating Performance Data

We determined the density of the pavilion’s façade based on three considerations: Structure, solar irradiation, and programmatic requirements for daylight and views. These considerations led to quantitative density requirements for the façade, which we represented as a labelled point cloud and integrated into the design process by matching these labels to the vertices of the mesh defining the pavilion’s form. Specifically, we simulated the pavilion’s structural displacement and solar irradiance, and determined relative daylight and view requirements according to the pavilion’s program. (The resulting density requirements were sufficient for the preliminary design of the pavilion’s façade. However, the approach presented here can also accommodate more precise simulation results and requirements.) The following two sections describe how we obtained the values for the three considerations of structure, solar irradiation, and program, how we translated these values into density requirements for the pavilion’s façade, and how we integrated these values with the half-edge data structure described above.

Determining Structural Displacement, Solar Irradiation, and Programmatic Requirements. For the displacement analysis, we assumed a preliminary dimension for the structural lines of the pavilion (i.e., the edges of the mesh triangles representing the pavilion’s form) of 200x60mm, and wood as a material. (The face-based data structure simplified the creation of this structural model through the direct accessibility of edges.) Simulating this structure under dead load with Karamba, a third-party structural analysis plugin, resulted in a maximal displacement of 12.5mm.

We conducted the solar irradiation analysis in DIVA, a daylight simulation plugin for Grasshopper. Using the vertices of the defining mesh as sensors, the simulation predicted a maximum annual solar irradiation of 1663kWh/m², reflecting the large amounts of sunlight typical for Singapore.

The pavilion’s program consisted of an exhibition area, a small library, and a performance area. Instead of defining numeric brightness requirements in terms of lux or UDI, we opted for a sketch approach of directly defining relative densities for the pavilion’s façade based on programmatic brightness requirements. To indicate these relative densities, the design team prepared a painted mesh. According to the values of the painted mesh (expressed as relative percentage from 0% to 100%), the exhibition areas should receive a relatively large amount of daylight and thus have a relatively open roof (~50%), with the library receiving less light (~75%), and the stage of the performance area being maximally closed (100%). The design team required the sides of the pavilion to be less dense than the roof (~25%), allowing views of the surroundings areas. The least dense areas of the façade (0%) half-enclosed a courtyard, allowing views into and across the courtyard.

Labelling the Half-Edge Data Structure with Density Requirements. Note that the structural, solar irradiation and programmatic analyses described in
the preceding section resulted in three values for every vertex \( n \) of the mesh defining the pavilion’s architectural form: The length of the displacement vector \( \delta_n \) expressed in mm, the annual solar irradiation amount \( \sigma_n \) expressed in kWh/m², and a programmatic requirement \( \phi_n \) expressed as a percentage. Further, note that the pattern discussed previously allows five different degrees of density \( x \), labelled from least to most dense as 0, 1, 2, 3, and 4. (See figure 3 for visualizations of the displacement, solar irradiation, and programmatic density requirements, and their combination.)

Instead of optimizing a difficult, time-consuming multi-objective problem to reconcile the three factors of structure, solar irradiation, and program, we directly calculated combined density requirements and integrated them into the half-edge data structure according to the four steps described below:

1. As the first step, we normalized the values for displacement \( \delta_n \) and annual solar irradiation \( \sigma_n \) according to their maximal values.
2. We then mapped the displacement values \( \delta_n \) to the range from 0 to 0.6, and the solar irradiation \( \sigma_n \) and programmatic values \( \phi_n \) to the range from 0 to 0.5.
3. We combined these values into a single density value per vertex according to following formula:
   
   \[
   \begin{align*}
   \text{if } \delta_n > \sigma_n + \phi_n & \Rightarrow \ x_n = [\delta_n \cdot 4] \\
   \text{if } \delta_n < \sigma_n + \phi_n & \Rightarrow \ x_n = [(\sigma_n + \phi_n) \cdot 4]
   \end{align*}
   \]
4. Finally, we labeled every mesh vertex with its corresponding density value by first sorting the density values according to their \( x \), \( y \) and \( z \) coordinates. (Analogous to the matching of half-edges, this method achieves the matching of density values in linearithmic time).

In other words, in cases where the sum of the values for solar irradiation and programmatic requirement were smaller than the displacement value, the density was determined based on the displacement value alone, resulting in a maximal density of degree 2. In cases where the sum of the solar irradiation value and the programmatic value was larger than the displacement value, this sum determined the density, resulting in a maximal density of degree 4. (This distinction between two cases reflects that the design team only intended the pattern elements up to density degree 2 to be structural, while the team intended the remaining density degrees 3 and 4 only for the modulation of daylight and views.) Due to the prodigious availability of sunlight in Singapore, the structural requirements influenced the combined density only to small degree, since the density required for shading was usually larger than the structural requirement.

**Generating Façade Components for Visual Preview and Digital Fabrication**

Above, we have discussed how three density values associated with the corner points of each pattern tile define that tile’s density and appearance. The previous section has explained how the design team derived these density values according to structural, shading, and programmatic considerations, and how the half-edge data structure could integrate these density values. We could thus generate the geometry of the pavilion’s façade by mapping the different parametric tiles of the pattern to the corner points of every façade component and their corresponding density requirements.

Since we separately defined the pattern tiles, the...
form of the triangular mesh, and the façade's density requirements, we could quickly generate different versions of the pavilion's geometry for different purposes. Specifically, we employed this parametric model to generate geometries for visual preview, for the 3D printing of scale models, and for the creation of full-scale prototypes. (We implemented these geometries with an objected-oriented programming paradigm; specifically, as a single class for the creation of the three types of façade components, with the pattern tiles defined in a second class.)

Due to the linearithmic matching of labelled points enabled by the mesh-based representation, and the constant time topological queries enabled specifically by the half-edge data structure, the designers could quickly recreate a visual preview (composed of simplified, two-dimensional components) from the underlying information of the mesh and its density values. Depending on the size of the mesh and the type of the required geometry, regenerating the model can take from several seconds to a couple of minutes. In this way, designers can interact with the form of the façade by manipulating the geometry of the mesh and the density of its pattern by manipulating the associated density values and derive visual feedback to their design changes (see figure 5 for a diagram of this process). One can also use the resulting model to validate the design with more accurate simulations of the daylight distribution inside the pavilion, allowing the simultaneous exploration of visual and appearance and performance (although these simulations take additional time.).

Beyond visual representation, we employed the parametric model for digital fabrication. To create an intersection-free solid geometry for the 3D printing of a scale model (see figure 4), we offset the vertices of every mesh triangle up and down according to their vertex normals, and defined the façade components according to this new set of corner points. (This technique avoids intersections between components, but results in slightly twisted sides for these components. See Aish et al. (2014) for a more extended discussion of this geometrical problem.)

The design team also created several full-scale prototypes of the pavilion's facade from folded sheet metal. (Figure 5 depicts one of these prototypes.) For these prototypes, the design team reinterpreted the façade components as two interlocking sheet metal elements forming a triangular box (see figure 6). The sheet metal elements consisted of thin aluminium and were cut by a large-scale CNC laser-cutting machine. The parametric model assisted in the creation of these elements by generating not only the geometry of the sheet metal components, but also by unfolding these elements into fabrication ready cut-sheets. To avoid the twisting of the components' sides described in the previous paragraph, we oriented the sides of each sheet metal component according to the normal of the corresponding edges of the underlying mesh triangle. (This orientation according to the edge normals leads to small misalignments at the nodes, i.e. the mesh vertices. We circumvented this problem by only connecting the sides of the façade components and by leaving the nodes devoid of material.)

Since the automatically generated cut sheets took the edge alignment of the completed components into account, the prototypes could be assembled without precisely checking the folding angle of each component's side. Instead, the components adjusted their shape during the assembly process, because the pre-defined boltholes only permitted the assembly of the intended form. (This process of self-adjustment was aided by the softness of the thin material.)
MODULARIZATION AND RE-INTEGRATION

The half-edge data structure encodes topological relationships such as adjacency, and thus allows constant time searches for queries such as the corner vertices of a faces, the faces surrounding a mesh vertex (necessary for calculating the vertex normal), and the two faces adjacent to a mesh edge (necessary for calculating the edge normal). As we have seen in the preceding section, the availability of such geometric information not only allowed the quick regeneration of the model for visual previews, but also aided in the generation of fabrication-ready geometries. Organizing the pavilion’s design around the information encoded by the half-edge data structure thus allowed the separation of four critical aspects of the design. These aspects were: (1) The architectural form of the pavilion, represented as a triangular mesh and encoded as a half-edge data structure, (2) the parametric tiles of the density-modulating pattern for the pavilion’s façade, (3) the desired density for the façade, based on considerations such a structure, solar irradiation, and program, and (4) the different geometries for representing and fabricating the pavilion’s façade components.

This modularity allowed the designers to explore different implementations of each aspect separate from the other three, and thus enabled collaboration between different members of the design team by allowing individual contributions to the design at different levels. For example, we implemented the façade components in terms of a simple, two-dimensional geometry for visual preview, as a solid geometry from 3D printing, and as folded sheet metal components suitable for the production of large-scale prototypes. This development of the façade components could proceed independently from the determination of density values for the pavilions façade, which the design team based on performance values.

Interactive visual previews, enabled by the near real-time time integration of the four aspects of architectural form, pattern, components, and density values, allowed the design team to judge the emerging appearance of the final design and, where necessary, amend the density values to achieve a better aesthetic result. The speed and convenience of the half-edge data structure, and the ease of matching labelled points to it, allowed this modularization
of the design process and the fast re-integration of the different modules. In this way, we combined the possibility of simultaneously exploring different aspects of the design with an ability for quick previews of the combined result, ensuring holistic control of the different aspects of the design. (See figure 6 for a diagram of the different modules and their relationships.)

CONCLUSION
In conclusion, we have identified three types of parametric models: (1) Linear flow models that create a set of outputs from a set of inputs, (2) BIM models that most commonly are collections of parametric components, and (3) centralized network models that organize parametric components according to a lightweight guiding geometry that integrates other inputs, such as performance data in a non-linear exploration structure.

We believe that the approach of modularizing an architectural design, and re-integrating it via a lightweight, topological data structure, mitigates the linearity of parametric design environments while adopting "intelligent" objects as building components as one of the key advantages of BIM-modelling.

Although the case study presented here relies heavily on customized programs, we believe that one can integrate the advantages of the third approach into existing CAAD packages by adopting modularized workflows to parametric design and perhaps by employing specialized plug-ins. The non-manifold meshes proposed as building representations by Aish and Pratap (2012) could extend this approach from planar to three-dimensional topologies. We see special relevance for centralized network models for aspects of a building design that lend themselves to a representation as components that are organized by an underlying, lightweight geometry and other, perhaps performative or construction-related, information such as building facades and structures.

As our case study demonstrates, such a centralized network - with the half-edge data structure at its centre - can generate different types of outcomes for both visualization and fabrication. While this study has considered only visual feedback to design changes, future work should aim to include automatic performance updates as well, while preserving flexibility for design exploration and intervention. Ultimately, we believe that approaches such as the one presented here might lead to design environments that allow designers to effortlessly integrate performance and fabrication aspects into their designs without unduly restricting their heuristic explorations.

Acknowledgement. We thank the anonymous reviewers for their insightful comments, which helped to strengthen and clarify the ideas in this paper. The pavilion presented as a case study is a design by the Advanced Architecture Laboratory (AAL) at Singapore University of Technology and Design.

REFERENCES
Aish, R and Aparajit, P 2012 "Spatial information modeling of buildings using non-manifold topology with ASM and DesignScript", Autodesk
Aish, R, Verboon, E and Fagerström, G 2014 'Topo-facade: envelope design and fabrication planning using topological mesh representations', Fabricate: Negotiating Design and Making, Zurich, CH

Botsch, M, Kobbelt, L, Pauly, M, Alliez, P and Levy, B 2010, Polygon Mesh Processing, A K Peters, Natick, MA


Hensel, M 2013, Performance-Oriented Architecture: Re-thinking Architectural Design and the Built Environment, Wiley, Chichester, UK


Lawson, B 2006, How designers think: the design process demystified, Elsevier/Architectural, Oxford; Burlington, MA

Loukissas, YA 2012, Co-designers: cultures of computer simulation in architecture, Routledge, Abingdon, UK


