With the introduction of physics-based algorithms and modeling environments, design processes have been shifting from the representation of materiality to the simulation of approximate material descriptions. Such computational processes are based upon enacting physical and material behavior, such as gravity, drag, tension, bending, and inflation, within a generative modeling environment. What is often lacking from this strategy is an overall understanding of computational design; that information of increasing value and precision is generated through the development and iterative execution of specific principles and integrative mechanisms. The value of a physics-based modeling method as an information engine is often overlooked, though, as they are primarily utilized for developing representational diagrams or static geometry – inevitably translated to function outside of the physical bounds and parameters defined with the modeling process. The definition of computational design provides a link between process and a larger approach towards architecture – an integrative behavior-based process which develops dynamic specific architectural systems interrelated in their material, spatial, and environmental nature. This paper, focusing on material integration, describes the relation of a computational design approach and the technical framework for a behavior-based integrative process. The application is in the development of complex tension-active architectural systems. The material behavior of tensile meshes and surfaces is integrated and algorithmically calibrated to allow for complex geometries to be materialized as physical systems. Ultimately, this research proposes a computational structure by which material and other sorts of spatial or structural behaviors can be activated within a generative design environment.
1 Introduction

Architectural design processes have long relied upon means of representation to produce design solutions. In contemporary methods, the media for representation is primarily geometry – depicted in a discrete number of elements (point, lines, surfaces, etc.), their finite positions, and their associativity. The relation to materiality, manufacturing, assembly, and operation is often applied – a flexible definition to which varying these properties does not have direct implications on the geometry formation. Subsequent simulation models serve to assign specificity, verify, rationalize or simply nullify material, structural, or environmental systems implied by such design geometries. The trend in integrative constraint-based design processes is to address the inclusion of specifications within the design geometry through geometric and mathematical rules; allowing critical properties to not be exclusive only to the simulation model (Mark, Gross, and Goldschmidt 2008). While this provides a framework for iterative geometry generation, it does not address the inclusion of processes which simulate integrated behaviors which are often too complex for simple mathematical and associative rules.

A simulation model, by definition, is not of static conditions, but rather an encapsulation of processes (Weinstock and Stathopolous 2006). In detail, the ‘model’ defines the complete static and dynamic characteristics which drive the internal capacities and external conditions influencing the local and global arrangements. Simply stating the properties and classification of the elements within the design geometry does not account for how the elements influence each other, and most importantly in manners that are not hierarchically predictable. Associativity, in terms of simulation, is not a discrete relationship; rather it is a process, in itself, which defines how elements behave individually and collectively. Form is akin to the ‘model’ – the information and repercussions of behavior through its relation to context.

In a behavior-based approach, the definitions of the elements themselves include the processes, in instances of manufacturing constraints, dynamics in assembly, and influences of operation. Form results from the resolution of the various behaviors – in geometric position, and association, as well as in the information which defines a particular state. The inclusion of information as a result of a simulating process relates a behavior-based approach more intimately to the simulation model. In architecture, this may be most evident in Frei Otto’s approach towards design through the simulation of material behavior in scaled form-finding models. Otto’s soap-film experiments are scaled material simulations utilized to mimic and test full-scale system behavior (Otto 1990). The most immediate translation of this is in the use of Finite Element Analysis. While clearly not an efficient design tool, the logic of FEA, though, has been taken into the design realm through physics-based computational modeling programs such as Axel Kilian’s CAdenary software, also referred to as CatenaryCAD (Chak et al. nd). The design environment, through activating springs as informed material force elements, provides rapid simulation for the form-finding of hanging chain models.

Resulting from this process of design simulation is information beyond mere geometric position which makes a distinct connection to further computationally intense and numerically accurate simulation models. The design information also falls more acutely into being classified as a design model, following the truer definition of ‘model’: the dataset which houses processes and elements indicating the internal systems which organize the design and external influences that define the entirety of the architecture. In the alignment of process, materialization and functioning, behavior-based design envelops a conceptual, computational, and practical understanding of architecture as form, which can be projected from a material system to system of systems – a material, spatial, and operational gestalt.

2 Systems and Computation

The term system, in the context of architectural form and practice, serves as a synthesizing concept by which the architectural system (form) can be seen as an integrated, spatial and material device, where its design is generated (in practice) by a computational system of algorithmic mechanisms operating to resolve internal and external performative criteria. Computationally, the methods need not mimic...
the dynamics of the entities and environments that are being produced. One is a
generating, evaluative, feedback device; the other, a dynamic operational material and
spatial form (Alexander 1968). Critical to an integrative view of the architectural system
is understanding that the computational systems which simulate integrative behavior
are information generating devices whereby information is continually expanded and
exacted. This is the only way in which integration (behavior) can be resolved, through
the evolution of increasing specificity, rather than in the sorting of pre-determined data
(knowledge) or on the foundations of generalized assumptions.

In most architectural projects, the early design stage is decisive in determining a
building’s eventual performance (Kimpian et al. 2009). However, few design tools
and methods, purposed for early stage design, have been developed to address the
complexity of interactions amongst material elements and predict relationships to their
environment. The task is evermore grand when environment is defined as not only a
series of climatic conditions but as differentiated situations of occupation, and spatial
quality. Two primary modes have emerged as processes for generating such architectural
systems. The first is a method of “divide and conquer” where systems are segregated,
optimized and dealt with in relative isolation, a combination of top-down organization
of the whole, and bottom-up in the formation of the individual systems (Liggett et al.
1992). The second is a systems behavior-based approach where the form is realized
in the process of interrelating material and spatial conditions (Weinstock 2004). This
computational process utilizes fundamental principles of physical properties, allowing
basic initial specifications (stiffness, elasticity) to be relayed into behaviors of how
multiple elements interact (under tension, compression, pressure, gravity, etc.).
Iterative development, at first, establishes the principles of various system aspects.
Subsequently, iteration serves as a part of a feedback function to allow for increasingly
particular specification and evaluation. The distinction and distance between the initial
form generation and the fully specified system in operation is minimized.

Examining material and computing its reactions to interrelated conditions, physical,
climatic and sociological, posits the design process as an iterative effort of gathering,
coordinating, comparing, re-generating and most importantly specifying information.
Gathering considers defined and adjustable levels of computational material prediction,
investigated through physical and computational experiments. This information can be
packaged within the constants of the computational process, though in themselves
acting as deterministic processes not invariant values or equations. Coordination
means the synchronization of information with methods of measure. Comparing and
re-generating activates iterative feedback. Specifying means a search for increasing
accuracy of information from initial abstract, simple principles, logics, and mathematical
parameters. The “divide and conquer” method tends to rely upon precedence, or a
knowledge-based approach, where previous studies provide specification up front in
the design process. In this case, information is recompiled rather than formulated.
This is an awkward imposition on the design process in that such specificity may not
register the uniqueness of a particular context or material resources.

This brings up a fundamental distinction in the approach towards computer-
based design in architecture. Terzidis states this clearly in the distinction between
computerization and computation. The former, implying a translation to the computer,
assimilates analog processes of the gathering and organization of well-formed
information into automated procedures to derive a result. The latter, implying a making
of and from computing processes, executes relational means which inductively and
deductively inform specific solutions (Terzidis 2003). The most basic understanding of
a functioning system is one which operates through levels of development, feedback
and redevelopment (Bertalanffy 1969) in search for a particular homeostasis. In this
sense, a computational approach connects the complex nature of architectural design
more closely to the realization and operation of architecture as the manifest of a
behavior-based generative system for realizing highly specific systems.

3 Computational Behavior

Addressing a behavior-based computational approach towards the design of force-
active material structures (often categorized as lightweight structures) is a logical
application. This is primarily because there are no simple mathematical equations which can describe the entirety of a surface defined by the equilibrium of applied force – be it tensile, compressive, or pressurized. The well-known examples of physical form-finding by Antoni Gaudi and Frei Otto depict a process by which physical simulations produced forms representative of pre-stressed structures. In Otto’s soap film experiments, the material generated surfaces with equalized force distribution in tension – a geometry which could be matched at full scale with tensioned textiles. Gaudi’s hanging chain models replicated mass, gravity, and topology to generate a tensile structure representative of a geometry that could be inverted to work as a compression only structure. With either of these methods comes a tricky translational step. There is no directness in the simulation between study model and physical structure. As mentioned previously, the most literal bridge between form-found geometry and simulated structure is through the use of Finite Element Analysis. Prior to computational means, force-active structures utilized exhaustive scaled simulations intended to be exact in their material characteristics and structural behavior, followed by hand calculations to analyze additional loading (Moncrieff 2005). Finite Element Analysis alleviates this need through allocation and simulation of a well-defined model, a clear example of a knowledge-based process. As such, FEA sits outside of a computational design approach, not only, obviously, in its demand for specified geometry, material, and contextual definitions, but in the fact that it has a singular goal in resolving the, albeit complex, structural behavior of the pre-stressed and loaded system.

Axel Kilian most notably introduced a computational approach for form-finding and model simulation with the research and development of his CADenary software. While still a form of representation – a “hanging” pure tension model which could be inverted to function as a compression-only structure – Kilian defined a methodology using Particle Systems which allowed for the approximate simulation of a physical hanging chain model (Kilian and Ochsendorf 2005). Within a particle system, a particle defines a point mass, while a spring acts to apply tension or compression forces between connected particles. Gravity is activated to create the hanging chain form with a series of interconnected particles and springs (with the springs acting only in tension). Beyond the possibility of iterative and cumulative methods of design (always a significant challenge in any physical form-finding method), the modeling environment based upon fundamental principles of behavior allows for the physical and integrated nature of a compression-only design to be investigated.

CADenary was developed using the programming environment Processing. This fact should be considered as inclusive to the computational design process: the development of a unique yet specialized modeling environment. The computational system is not completely universal – simply capturing the fundamentals of the hanging chain model, yet allowing for complex investigations to take place. Other capacities within the design environment offered possibilities to expand the structural investigation of the system, activating gravity in alternative vectors to test the stability of a given form
and programming constraints for fabrication (Kilian 2004). While Kilian packaged these concepts, techniques and technologies as software, the entirety of the development process, more compellingly, evokes a framework for the design of the computational process as well as its activation: development of process principles, studied through generated materialization, and accumulated into a specific computational modeling environment – all executed to discover new principles in the targeted compression-only structural-material system. This satisfies the conditions of a computational approach as one differentiated from an approach based on pre-packaged processes to one which focuses on a ‘meta-design’ of fundamental behaviors and principles (Coons 1963). The inter-looping process coordinates fundamentals discerned from physical and computational experiments, activated within an iterative generation and feedback process, ultimately augmented with precise model simulation (Figure 1).

4 System Complexity

Looking primarily at force-active tensile structures, there is a particular challenge in recognizing the distinction between the behavior of tensioned meshes (cable nets) and tensioned surfaces (textiles or membranes) while registering the differences within the computational framework. A fundamental distinction lies in the materialization between a mesh and a surface for tension-active systems. A mesh is comprised of a series of individual elements connected together with fixed nodes (Figure 2). A surface is defined by fibers which are essentially overlapping in two primary directions, defined as the warp and weft. In performing against a tensile load, a surface will always look to distribute the forces as evenly and equally as possible. Force is resolved across the entire surface, though with influence of variables between the warp and weft of the textile construction. For a mesh, force is resolved individually with each node. This means that a mesh can realize a more varied set of morphologies (form, topology, and tension behavior), but possibly resulting in high-degrees of variation in the force intensities across the whole system. This is not always a problem, but can result in varied performance against additional loading such as wind, or varied degradation of material (Lewis 2003). These are critical conditions to consider in tension-active systems, but can be resolved in parameters of topology and feedback from structural simulation.

What is necessary to note in the process of design generation, when qualifying the material distinctions stated above, is the degree of variation in tension distribution across the spring topology. As stated, tensioned meshes can be accomplished with essentially any topological arrangement of springs. This can be registered numerically within the relative values that the tensioned springs return within the Particle System environment. It can also be measured by the degree of curvature. An ideally tensioned geometry is anticlastic (doubly-curved) with a Gaussian curvature of zero. The degree

**Figure 2.** Materialization and variable computational topology for fixed mesh

**Figure 3.** Materialization and strict computational topology for woven textile

**Figure 4.** Various aspects of tension-active system behavior

**Figure 5.** Translation from geometry and force description to un-tensioned 2D template for fabrication
of double curvature in the geometry will indicate the degree of tension, where less curvature means a more intense moment of tension (Knippers et al. 2010).

For tensioned surfaces, the topology of springs is more constrained. The goal is to simulate the bi-directional nature of a textile construction to where forces will always tend to distribute across the entire system of connected elements. Representation of the overlapping fibrous behavior of a woven material would be incredibly computationally intense. With a fixed topology of springs that registers the warp and weft direction of a textile (like the U and V directions of a surface), the bi-directional nature of how tension distributes can be approximated (Figure 3). There are of course plenty of variables within this to investigate complex arrangements of interconnected tensioned surfaces. Double curvature, again, is a characteristic to measure. Though, unlike the ability to manipulate every element individually within the topology of a tensioned mesh, curvature within a tensioned surface can only be controlled by varying parameters between the sets of U and V springs, and altering the boundary conditions.

What becomes evident in developing and executing a behavior-based design process is the accumulation of information beyond that of static geometry. While Particle Systems, outside of the work described previously by Axel Kilian, have been utilized as form-finding engines, the wealth of information that can be resourced is often overlooked – geometry being the primary and singular motive. Within a Particle System, a spring is calculated following Hooke’s Law of Elasticity. This states that a force can be determined by a spring’s degree of displacement from its rest length (rest length is the moment at which a spring exerts no force) (Gordon 2006). While this does not account for non-linear behavior, it does provide an approximate and relative description of forces as they move through a particular tensile topology. This can be useful for understanding, as mentioned before, variations in force distribution. Ultimately, it provides an intimate opportunity and feedback for investigating the inextricable relationship between topology, force, and form. The accumulative values of force at the boundary conditions allow for force intensities and force vectors to be visualized at the points where the system anchors to secondary structures (Figure 4). Generating form is as much about manipulating and evaluating force information as it is about negotiating topology and geometry.

A pivotal point in evaluating the system behavior as an information model is the necessity for coordinated information of force, topology, and geometry for materialization. In this case, materialization should be considered in terms of generating verification models (for the system principles) and mock-ups. By no means does the ‘systems behavior’ portion of the process generate thorough and accurate enough information to produce components for full scale construction. Such is the process described including ‘system simulation’ as the logical step towards production and full-scale assembly. In developing scaled models from the Particle System information, the key aspect is understanding elasticity. With the use of elastic materials for modeling, double curvature can be accomplished. At architectural scale, elasticity is minimized to the point that each element is broken down into singly curved geometries (Linhard 2009). With a highly elastic material double curvature in tension can be accomplished from a flat un-tensioned sheet.Translating from a doubly-curved geometry to this flat sheet cannot be done by a simple geometric ‘unrolling’ algorithm. But, by including the information of force and the geometric development (Figure 5), a particular spring topology can be simultaneously unstressed and unrolled to determine its flat 2D template (Figure 6).

In execution, this behavior-based computational process continually produces information which reverberates between physical simulations, computational generation of material behaviors, and simulation of specific parameters and material definitions. It offers a distinct definition for the meta-framework and individual processes for computational design. In practice, this has enabled the examination of tension-active systems as integrated cellular mesh and surface systems. Working first with physical form-finding studies to exact basic cell definitions, the computational strategy is constructed to express the cells as spring topologies (Figure 7). From both a design and computational perspective, inevitable complexity can be managed...
Through an effort where design variation occurs primarily through topology variation. As the springs form-find force equilibrium, the description of geometry is not an explicit design task. Geometry is part of the information that results from the manipulation of topology and resolving of force distribution. The primary topology of an individual component in this research is that of a cylinder. Such a fundamental organization can be easily transformed to simpler topologies such as a plane or more complex ones such as interconnected cylinders or toroidal arrangements (Figure 8). Through iterative steps of model generation and translation to physical mock-ups, a continual expansion of the intricacy, topology and geometry complex is pursued (Figure 9).

5 Conclusion

Computation has broken into the realm of offering the possibility to mimic methods of design simulation as opposed to design representation. Form-finding, in its most literal (and historical) definition is the scalar determination of form through the replication of real world contextual and material conditions. The familiar models of Antoni Gaudi, Frei Otto and others are the foundation for computational design simulation processes. Axel Kilian, with this CADenary software, and the various physics-based engines, such as Kangaroo, Nucleus for Maya, and Traer Physics, offer an adaption of these previous physical model-based processes. But they do not offer a comparable framework when design computation delves into the realms of considering multiple design factors, and the necessity for generative design iteration to determine effectiveness between various criteria.

This research has focused on initiating principles of material and structural behavior to enact an intense but iterative computational design process. This encompasses a vastly integrative view of design process which includes the synchronous development in forming the meta-framework, the programmed behavior-based processes, and the iterative triggering of the design generating system. The systems based approach begins to register a direct, not representational, connection towards a view of architecture as dynamic in its response to material resources, behaviors, and contextual input. While the studies from this process focus on the development of complex tension-active systems, and primarily their potential for complex morphological description, the framework can be expanded to envision an even more integrative view of architecture. Where complex behaviors can be computed from simple principles, the effort of integration becomes an expansion of the ‘process constants’ – the engines which define individual behavior and produce information to resolve collective behavior. Computational design relies upon the specificity and iterative coordination of principle behaviors; a significant shift from design processes which function upon collective generalizations and organization of preconceived formal structures and information.

References


